NEXT

**CCIE Professional Development Routing TCP/IP, Volume I, Second Edition**

By Jeff Doyle - CCIE No. 1919, Jennifer Carroll - CCIE No. 1402

.....

Publisher: Cisco Press
Pub Date: October 19, 2005
ISBN: 1-58705-202-4
Pages: 936

Table of Contents | Index

## Overview

A detailed examination of interior routing protocols -- completely updated in a new edition

- A complete revision of the best-selling first edition--widely considered a premier text on TCP/IP routing protocols
- A core textbook for CCIE preparation and a practical reference for network designers, administrators, and engineers
- Includes configuration and troubleshooting lessons that would cost thousands to learn in a classroom and numerous real-world examples and case studies

Praised in its first edition for its approachable style and wealth of information, this new edition provides readers a deep understanding of IP routing protocols, teaches how to implement these protocols using Cisco routers, and brings readers up to date protocol and implementation enhancements. *Routing TCP/IP, Volume 1*, Second Edition, includes protocol changes and Cisco features that enhance routing integrity, secure routers from attacks initiated through routing protocols, and provide greater control over the propagation of routing information for all the IP interior routing protocols. *Routing TCP/IP, Volume 1*, Second Edition, provides a detailed analysis of each of the IP interior gateway protocols (IGPs). Its structure remains the same as the best-selling first edition, though information within each section is enhanced and modified to include the new developments in routing protocols and Cisco implementations. What's New In This Edition? The first edition covers routing protocols as they existed in 1998. The new book updates all covered routing protocols and discusses new features integrated in the latest version of Cisco IOS Software. IPv6, its use with interior routing protocols, and its interoperability and integration with IPv4 are also integrated into this book. Approximately 200 pages of new information are added to the main text, with some old text removed. Additional exercise and solutions are also included.

NEXT

**CCIE Professional Development Routing TCP/IP, Volume I, Second Edition**

By Jeff Doyle - CCIE No. 1919, Jennifer Carroll - CCIE No. 1402

.....
Publisher: Cisco Press
Pub Date: October 19, 2005
ISBN: 1-58705-202-4
Pages: 936



- Copyright
- About the Authors
  - About the Technical Reviewers
- Acknowledgments
- This Book Is Safari Enabled
- Icons Used in This Book
- Command Syntax Conventions
- Foreword
- Introduction
  - Objectives
  - Audience
  - Changes from First Edition
  - Organization
  - Book Features
- Part I: Routing Basics
  - Chapter 1. TCP/IP Review
    - TCP/IP Protocol Layers
    - IP Packet Header
    - IPv4 Addresses
    - Address Resolution Protocol (ARP)
    - Internet Control Message Protocol (ICMP)
    - Host-to-Host Layer
    - Looking Ahead
    - Summary Table: Chapter 1 Command Review
    - Recommended Reading
    - Review Questions
    - Configuration Exercises
    - Troubleshooting Exercises
  - Chapter 2. IPv6 Overview
    - IPv6 Addresses
    - IPv6 Packet Header Format
    - Extension Headers
    - ICMPv6
    - Neighbor Discovery Protocol
    - Looking Ahead
    - Review Questions
  - Chapter 3. Static Routing
    - Route Table
    - Configuring Static Routes
    - Troubleshooting Static Routes
    - Looking Ahead

- Summary Table: Chapter 3 Command Review
- Review Questions
- Configuration Exercises
- Troubleshooting Exercises
- Chapter 4. Dynamic Routing Protocols
  - Routing Protocol Basics
  - Distance Vector Routing Protocols
  - Link State Routing Protocols
  - Interior and Exterior Gateway Protocols
  - Static or Dynamic Routing?
  - Looking Ahead
  - Recommended Reading
  - Review Questions
- Part II: Interior Routing Protocols
  - Chapter 5. Routing Information Protocol (RIP)
    - Operation of RIP
    - Configuring RIP
    - Troubleshooting RIP
    - Looking Ahead
    - Summary Table: Chapter 5 Command Review
    - Recommended Reading
    - Review Questions
    - Configuration Exercises
    - Troubleshooting Exercises
  - Chapter 6. RIPv2, RIPng, and Classless Routing
    - Operation of RIPv2
    - Operation of RIPng
    - Configuring RIPv2
    - Configuring RIPng
    - Troubleshooting RIPv2 and RIPng
    - Looking Ahead
    - Summary Table: Chapter 6 Command Review
    - Recommended Reading
    - Review Questions
    - Configuration Exercises
    - Troubleshooting Exercises
  - Chapter 7. Enhanced Interior Gateway Routing Protocol (EIGRP)
    - The Roots of EIGRP: An Overview of IGRP
    - From IGRP to EIGRP
    - Operation of EIGRP
    - Configuring EIGRP
    - Troubleshooting EIGRP
    - Looking Ahead
    - Summary Table: Chapter 7 Command Review
    - Review Questions
    - Configuration Exercises
    - Troubleshooting Exercises
  - Chapter 8. OSPFv2
    - Operation of OSPF
    - Configuring OSPF
    - Troubleshooting OSPF
    - Looking Ahead
    - Summary Table: Chapter 8 Command Review
    - Recommended Reading

- Review Questions
- Configuration Exercises
- Troubleshooting Exercises
- Chapter 9. OSPFv3
  - Operation of OSPFv3
  - Configuring OSPFv3
  - Troubleshooting OSPFv3
  - Looking Ahead
  - Summary Table: Chapter 9 Command Review
  - Recommended Reading
  - Review Questions
  - Configuration Exercises
- Chapter 10. Integrated IS-IS
  - Operation of Integrated IS-IS
  - Configuring Integrated IS-IS
  - Troubleshooting Integrated IS-IS
  - Looking Ahead
  - Summary Table: Chapter 10 Command Review
  - Review Questions
  - Configuration Exercises
  - Troubleshooting Exercises
- Part III: Route Control and Interoperability
  - Chapter 11. Route Redistribution
    - Principles of Redistribution
    - Configuring Redistribution
    - Looking Ahead
    - Summary Table: Chapter 11 Command Review
    - Review Questions
    - Configuration Exercises
    - Troubleshooting Exercises
  - Chapter 12. Default Routes and On-Demand Routing
    - Fundamentals of Default Routes
    - Fundamentals of On-Demand Routing
    - Configuring Default Routes and ODR
    - Looking Ahead
    - Summary Table: Chapter 12 Command Review
    - Review Questions
  - Chapter 13. Route Filtering
    - Configuring Route Filters
    - Looking Ahead
    - Summary Table: Chapter 13 Command Review
    - Configuration Exercises
    - Troubleshooting Exercises
  - Chapter 14. Route Maps
    - Basic Uses of Route Maps
    - Configuring Route Maps
    - Looking Ahead
    - Summary Table: Chapter 14 Command Review
    - Review Questions
    - Configuration Exercises
    - Troubleshooting Exercise
- Part IV: Appendixes
  - Appendix A. Tutorial: Working with Binary and Hex
    - Working with Binary Numbers

- Working with Hexadecimal Numbers
- Appendix B. Tutorial: Access Lists
  - Access List Basics
  - Standard IP Access Lists
  - Extended IP Access Lists
  - Calling the Access List
  - Reflexive Access Lists
  - Keyword Alternatives
  - Named Access Lists
  - Prefix Lists
  - Filter Placement Considerations
  - Access List Monitoring and Accounting
- Appendix C. CCIE Preparation Tips
  - Laying the Foundations
  - Following the Certification Path
  - Hands-On Experience
  - Intensifying the Study
  - The Final Six Months
  - Exam Day
- Appendix D. Answers to Review Questions
  - Chapter 1
  - Chapter 2
  - Chapter 3
  - Chapter 4
  - Chapter 5
  - Chapter 6
  - Chapter 7
  - Chapter 8
  - Chapter 9
  - Chapter 10
  - Chapter 11
  - Chapter 12
  - Chapter 14
- Appendix E. Solutions to Configuration Exercises
  - Chapter 1
  - Chapter 3
  - Chapter 5
  - Chapter 6
  - Chapter 7
  - Chapter 8
  - Chapter 9
  - Chapter 10
  - Chapter 11
  - Chapter 13
  - Chapter 14
- Appendix F. Solutions to Troubleshooting Exercises
  - Chapter 1
  - Chapter 3
  - Chapter 5
  - Chapter 6
  - Chapter 7
  - Chapter 8
  - Chapter 10
  - Chapter 11

Chapter 13
Chapter 14
Index

# Copyright

## CCIE Professional Development Routing TCP/IP Volume I Second Edition

Jeff Doyle, CCIE No. 1919, Jennifer Carroll, CCIE No. 1402

Copyright © 2006 Cisco Systems, Inc.

Published by:

Cisco Press

800 East 96th Street

Indianapolis, IN 46240 USA

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

Printed in the United States of America 1 2 3 4 5 6 7 8 9 0

First Printing October 2005

Library of Congress Cataloging-in-Publication Number: 2004104363

## Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc. cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

## Warning and Disclaimer

This book is designed to provide information about routing TCP/IP. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an "as is" basis. The authors, Cisco Press, and Cisco Systems, Inc. shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

## Corporate and Government Sales

Cisco Press offers excellent discounts on this book when ordered in quantity for bulk purchases

or special sales.

For more information please contact: U.S. Corporate and Government Sales 1-800-382-3419 corpsales@pearsontechgroup.com

For sales outside the U.S. please contact: International Sales international@pearsoned.com

# Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through e-mail at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

| | |
|---|---|
| Publisher | John Wait |
| Editor-in-Chief | John Kane |
| Executive Editor | Brett Bartow |
| Cisco Representative | Anthony Wolfenden |
| Cisco Press Program Manager | Jeff Brady |
| Production Manager | Patrick Kanouse |
| Development Editor | Andrew Cupp |
| Senior Project Editor | San Dee Phillips |
| Copy Editor | Interactive Composition Corporation |
| Technical Editors | Frank Knox, Steven Edward Moore, Rena Yang |
| Editorial Assistant | Tammi Barnett |
| Book and Cover Designer | Louisa Adair |
| Composition | Interactive Composition Corporation |
| Indexer | Tim Wright |

CISCO SYSTEMS

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: 31 0 20 357 1000
Fax: 31 0 20 357 1100

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
Capital Tower
168 Robinson Road
#22-01 to #29-01
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on the Cisco.com Web site at www.cisco.com/go/offices.

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China PRC • Colombia • Costa Rica • Croatia • Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong SAR • Hungary • India • Indonesia • Ireland • Israel • Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal • Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia • Slovenia • South Africa • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

Copyright © 2003 Cisco Systems, Inc. All rights reserved. CCIP, CCSP, the Cisco Arrow logo, the *Cisco Powered* Network mark, the Cisco Systems Verified logo, Cisco Unity, Follow Me Browsing, FormShare, iQ Net Readiness Scorecard, Networking Academy, and ScriptShare are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, The Fastest Way to Increase Your Internet Quotient, and iQuick Study are service marks of Cisco Systems, Inc.; and Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, the Cisco IOS logo, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherSwitch, Fast Step, GigaStack, Internet Quotient, IOS, IP/TV, iQ Expertise, the iQ logo, LightStream, MGX, MICA, the Networkers logo, Network Registrar, *Packet*, PIX, Post-Routing, Pre-Routing, RateMUX, Registrar, SlideCast, SMARTnet, StrataView Plus, Stratm, SwitchProbe, TeleRouter, TransPath, and VCO are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries.

All other trademarks mentioned in this document or Web site are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0303R)

Printed in the USA

# Dedications

I would like to dedicate this book to my wife, Sara, and my children, Anna, Carol, James, and Katherine.

Jeff

I would like to dedicate this book to my husband, Mike, and sons, Mitchell and Jonathan. Their patience and support helped me complete this book.

Jennifer

PREV NEXT

# About the Authors

Jeff Doyle (CCIE No. 1919) specializes in IP routing protocols, MPLS, and IPv6. He has designed or assisted in the design of large-scale IP service provider networks throughout North America, Europe, Japan, Korea, and the People's Republic of China. Jeff has presented numerous corporate seminars, and has also spoken at NANOG, JANOG, APRICOT, and at IPv6 Forum conferences. Jeff holds a BA from Memphis State University, and studied Electrical Engineering at the University of New Mexico. Jeff lives in Denver, Colorado.

Jennifer Carroll (CCIE No. 1402) is an independent network consultant in Redmond, Washington. She has designed, implemented, and optimized many TCP/IP networks, and has developed and taught a variety of networking and internetworking courses on routing protocols and Cisco routers over the past 15 years. Jennifer can be contacted at jennifer.carroll@ieee.org.

# About the Technical Reviewers

Frank Knox, Chief Technical Officer, has been with Skyline Computer for a little over six years. He is a dual CCIE (CCIE No. 3698: SNA/IP and Routing/Switching) as well as a CCSI. In addition to his CTO responsibilities, Frank teaches several advanced Cisco-related courses, including a one-week CCIE Lab Preparation Workshop. He is considered to be an expert in mainframe attached router technologies and in the technologies and issues associated with integrated networking (for example, SNA/IP and Voice/Data). He has more than 37 years of networking experience with IBM, GTE (Verizon) Directories, and Skyline Computer Corp. This experience includes field service, field support, product planning, management, and all facets of networking education. In addition, he developed and taught several courses for the University of Dallas Telecommunications MBA program. Frank also has an MS degree in Telecommunications from Pace University (4.0 GPA).

After working in various roles as an engineer within Cisco for the past 6.5 years, Steven Edward Moore transitioned to the IP Routing Protocol Scalability Team. There, his focus encompasses all aspects of extending network and protocol scalability: considering new features and optimizations to the protocol architecture, designing tests to measure current protocol scalability, working with customers to implement scaling functionality in their network, and participating in events such as Networkers to educate others on how to enhance their network's performance and scalability from the routing perspective.

Rena Yang is a software engineer at Cisco Systems. She has more than six years of experience implementing code in Cisco IOS. She currently works on IS-IS. Before this, she focused on IPv4, UDP, access lists, policy routing, and routing infrastructure. Rena holds a bachelor's of science and masters of engineering in computer science from MIT.

# Acknowledgments

Many thanks to Brett Bartow, Chris Cleveland, Andrew Cupp, San Dee Phillips, and all of the staff of Cisco Press who made this book possible.

The technical editors, Steven Moore, Rena Yang and Frank Knox, did a fantastic job. We want to thank them for their outstanding advice and recommendations.

We want to thank Frank Knox, Carl Pike, Chris Tonini, and the rest of the employees of Skylabs networks. Skylabs' lab setup and access to the lab is easy to use and had everything we needed to complete all the configurations and case studies in this book.

# This Book Is Safari Enabled

The Safari® Enabled icon on the cover of your favorite technology book means the book is available through Safari Bookshelf. When you buy this book, you get free access to the online edition for 45 days.

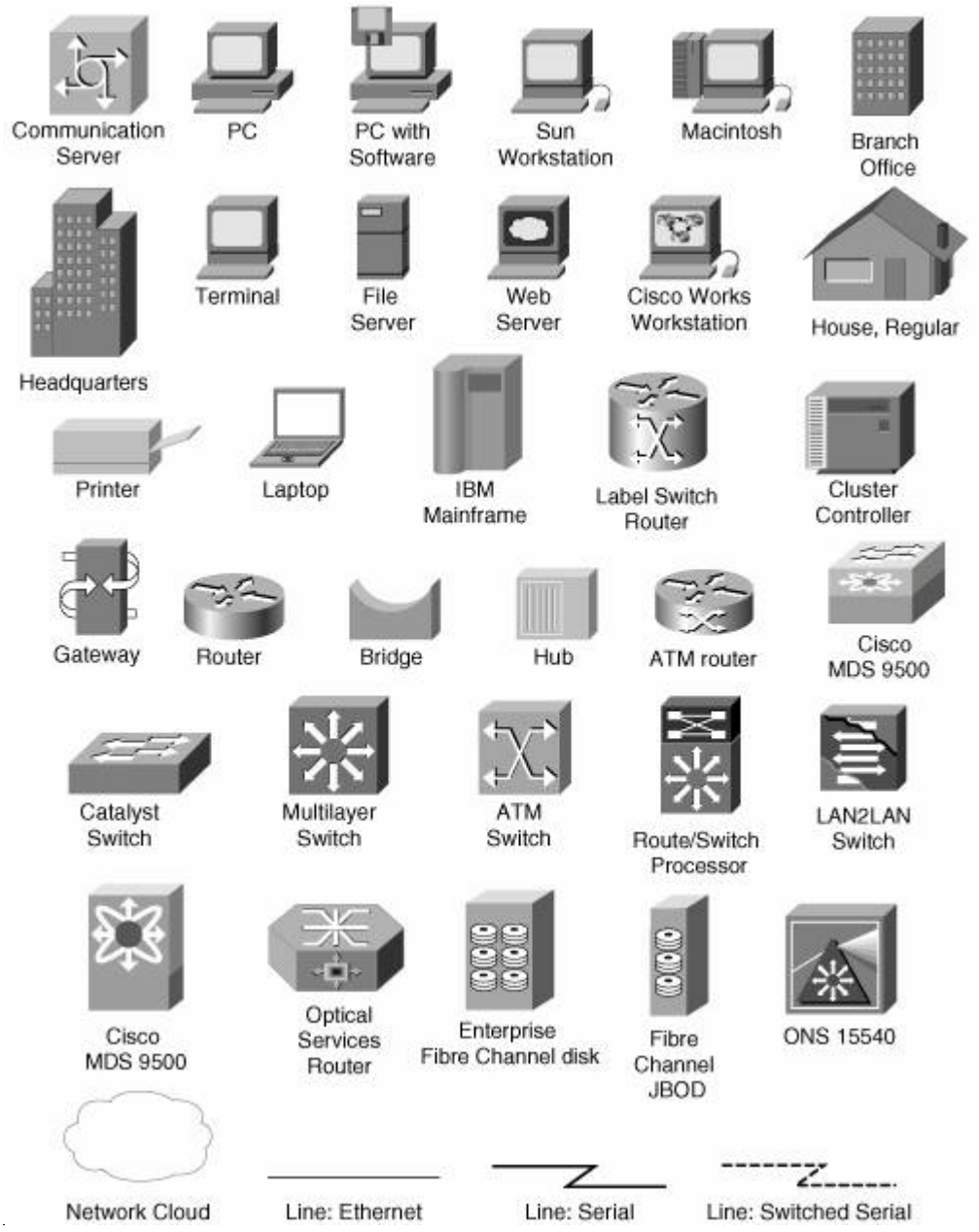
Safari Bookshelf is an electronic reference library that lets you easily search thousands of technical books, find code samples, download chapters, and access technical information whenever and wherever you need it.

To gain 45-day Safari Enabled access to this book:

- Go to http://www.ciscopress.com/safarienabled
- Enter the ISBN of this book (shown on the back cover, above the bar code)
- Log in or Sign up (site membership is required to register your book)
- Enter the coupon code MSJJ-PPVL-4EMT-TVK8-7JDF

If you have difficulty registering on Safari Bookshelf or accessing the online edition, please e-mail customer-service@safaribooksonline.com.

# Icons Used in This Book



# Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italics* indicate arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets [ ] indicate optional elements.
- Braces { } indicate a required choice.
- Braces within brackets [{ }] indicate a required choice within an optional element.

# Foreword

In 1976, when I saw my first Arpanet IMP at Digital Equipment Corporation, networks as we know them today were in their infancy. SNA, XNS, and DECnet were under early development, and packet switching versus circuit switching was the hot topic of the day. Those of us involved in the design of the switching and routing algorithms were dealing with routers (although we didn't call them that) that had 64 kilobytes of memory, data link of 56 kilobits were considered blindingly fast, and networks with 256 nodes were big enough that if you were the salesman who sold those 256 computers, you would retire fabulously wealthy.

Thirty years is a long time, and today the individual networks that make up the Internet contain thousands or tens of thousands of nodes, while the Internet as a whole contains hundreds of millions of computers. Most striking in the evolution over this human generation is that the foundations of the Internet laid down in the TCP/IP protocol suite have survived mostly intact through four or more generations of computing architectures, three complete generations of operating system technology, and an increase of five orders of magnitude in transmission speeds.

Yet, we still treat routing in packet-switched networks as a black art. Why is that?

First, designing robust, scalable distributed algorithms is hard. Despite our best intentions to make them simple, complexity creeps in to deal with the inevitable special cases, optimizations, peculiar topologies, and link technologies one encounters. Because a "fork lift upgrade" of an entire network is rarely feasible, we have multiple generations of technology present simultaneously, and we must maintain backward-compatibility with essentially no disruption to deployed services. As policies governing the routing of packets become more sophisticated, our ability to devise automated discovery and configuration procedures gets overwhelmed, and we fall back on manual configuration and performance tuning techniques. Finally, as the environment in which these networks are operated has evolved from a cooperative one where trust was implicit to one in which the network is subject to both inside and outside attack, designing and deploying routing systems that can be made secure has become an urgent priority.

*Routing TCP/IP* tackles this black art comprehensively. The present Volume 1 covers all the needed fundamentals of TCP/IP networks and gives you all the tools needed to understand how routing is accomplished within a single administrative region of the Internet. Straightforward ideas of packet-switched routing are presented first in the chapters on addressing and static routing. The most popular IGPsRIP, EGRP, OSPF, and ISISare covered in depth. Advanced topics in route redistribution, route filtering, and policy routing round out Volume 1.

This second edition also adds essential material on IPv6 as well as bringing all the material up to date with examples and configurations for the latest releases of Cisco IOS.

For anyone wanting a comprehensive understanding of how routing in TCP/IP networks really works, from the design principles of routing algorithms, to the evolution of addressing schemes, to the practical aspects of designing and configuring the routing of large autonomous systems, this is the book for you.

David Oran

Cisco Fellow

# Introduction

Routing is an essential element of all but the smallest data communications networks. At one level, routing and the configuration of routers are quite simple. But as networks grow in size and complexity, routing issues can become at once both large and subtle. Perversely, perhaps, we are grateful for the difficult problems large-scale routing can presentas network systems consultants, these problems are our bread and butter. Without them, the phrase "You want fries with that?" could be an unfortunate part of our daily vocabulary.

Cisco Certified Internetwork Experts are widely recognized for their ability to design, troubleshoot, and manage large networks. This recognition comes from the fact that you cannot become a CCIE by attending a few classes and then regurgitating some memorized facts onto a written test. A CCIE has proven expertise in an intense, famously difficult hands-on lab exam.

# Objectives

This book is the first of two volumes that focuses on TCP/IP routing issues. Early in the writing of the first edition, Kim Lew, former Cisco Systems program manager, said, "Our objective is to make CCIEs, not to make people who can pass the CCIE lab." We entirely agree with that statement and have used it as a guiding principle throughout the writing of this book. Although the book includes many case studies and exercises to help you prepare for the CCIE lab, my primary objective is to increase your understanding of IP routingboth on a generic level and as it is implemented on Cisco routers.

# Audience

The audience for this book is any network designer, administrator, or engineer who needs a full understanding of the interior routing protocols of TCP/IP. Although the practical aspects of the book focus on the Cisco IOS, the information is applicable to any routing platform.

The book is not only for readers who plan to become CCIEs, but for people who wish to advance their knowledge of TCP/IP routing. These readers will fall into one of three categories:

- The "beginners" who have some basic networking knowledge and wish to begin a deep study of networking.
- The intermediate-level networking professionals who have experience with routers, Cisco or otherwise, and plan to advance that experience to the expert level.
- The highly experienced networking experts. These individuals have extensive hands-on expertise with Cisco routers and are ready to take the CCIE lab; however, they want a structured review and series of exercises for verification and validation.

*CCIE Professional Development: Routing TCP/IP, Volume I* focuses primarily on intermediate-level networking professionals while offering to beginners a structured outline of fundamental information and to experts the required challenges to hone their skills.

# Changes from First Edition

There are several factors influencing the changes contained in this second edition. The first factor is the CCIE itself. When I (Jeff) wrote the first edition of this book, the CCIEspecifically what is now called the Routing and Switching specialty of the CCIEwas the only certification Cisco Systems offered. Now, there is a series of certifications creating a path to the CCIE at the pinnacle. Moreover, the typical networking professional is more knowledgeable than in 1997. Given this, we have eliminated the first chapter of the original book, which covered such very basic concepts as the definition of bridges and routers and network addresses. (When was the last time you even saw a bridge in a network?)

The second factor influencing the changes in this edition is the changes in the Cisco Systems IOS. IGRP, which was frequently used when the first edition was written, is now a legacy protocol whose main significance is as the ancestor of EIGRP. Therefore the IGRP chapter of the first edition has been eliminated and IGRP is covered for historical perspective early in the EIGRP chapter. The IOS command suite itself has expanded to accommodate new functions and options; we have made every effort to include the commands and protocol extensions that did not exist in the late 1990s.

Lastly, a protocol that existed mostly only in proposal form in 1997IPv6is now in the early stages of worldwide deployment. You can expect to need a detailed knowledge of this protocol and the extensions to IP routing protocols that support it in the near future, if not already, so this second edition delves deeply into routing IPv6.

Other changes in this edition are semantic. For example, in the first edition, I (Jeff) made a point of differentiating between a "network" as a data link and an "internetwork" as a set of networks connected by routers. Although that terminology is certainly accurate, it is clumsy, and "internetwork" is seldom used these days. Instead, "network" usually refers to everything from a local link to worldwide autonomous systems operated by the likes of Level 3, NTT, and Sprint. We have attempted to bring the terminology in this edition up to modern, common usage.

# Organization

The 14 chapters of the book are divided into three parts.

Part I, "Routing Basics," examines the basics of IPv4 and IPv6, and the basics of routing. Although more advanced readers may wish to skip the first chapter, we recommend that they at least skim Chapter 3, "Static Routing," and Chapter 4, "Dynamic Routing Protocols." And, of course, if you are not yet familiar with IPv6, Chapter 2, "IPv6 Overview," is a must-read.

Part II, "Interior Routing Protocols," covers the IP Interior Gateway Protocols. Each protocol-specific chapter begins with a discussion of the theory, mechanics, and parameters of the protocol. This general overview is followed by case studies on configuring and troubleshooting the protocol using Cisco Systems' IOS in various network topologies.

The Exterior Gateway Protocol, BGP, and topics such as multicast routing, Quality of Service, router security and management, and Network Address Translation, are covered in "Routing TCP/IP, Volume II."

Part III, "Route Control and Interoperability," examines the tools available for creating and managing interoperability with multiple IP routing protocols, and also such tools as default routes and route filtering. As such, the chapters of this last part provide an introduction to the tools necessary for building the complex routing policies introduced in Volume II. These chapters, like the ones in Part II, begin with concepts and conclude with case studies.

# Book Features

Most chapters conclude with a set of review questions, configuration exercises, and troubleshooting exercises. The review questions focus on the theoretical aspects of the chapter topic, whereas the configuration and troubleshooting exercises address Cisco-specific aspects of the chapter topic.

Also at the end of each chapter is a table with a brief description of all important Cisco IOS commands used in that chapter. The conventions used to present these commands are the same conventions used in the IOS Command Reference and presented earlier in this introduction.

# Part I: Routing Basics

Chapter 1 TCP/IP Review

Chapter 2 IPv6 Overview

Chapter 3 Static Routing

Chapter 4 Dynamic Routing Protocols

# Chapter 1. TCP/IP Review

This chapter covers the following subjects:


- TCP/IP Protocol Layers
- IP Packet Header
- IPv4 Addresses
- Address Resolution Protocol (ARP)
- Internet Control Message Protocol (ICMP)
- Host-to-Host Layer


Given that the title of this book is *Routing TCP/IP*, it is fitting to begin with a review of TCP/IP before getting into how to route it. Presumably, if you are preparing for a Cisco Certified Internetwork Expert (CCIE) examination, or have just bought this book as a routing reference, you already know most or all of the information in this chapter. But reviews never hurt and sometimes help, so here you have it.

The purpose of this chapter is to review the protocols that enable, control, or contribute to the routing of TCP/IP, not to do an in-depth study of the TCP/IP protocol suite. Several books on the recommended reading list at the end of the chapter cover the subject in depth. Read at least one.

Conceived in the early 1970s by Vint Cerf and Bob Kahn, TCP/IP and its layered protocol architecture predates the ISO's Open System Interconnection (OSI) reference model. A brief review of TCP/IP's layers will be useful in understanding how the various functions and services examined in this chapter interrelate.

# TCP/IP Protocol Layers

Figure 1-1 shows the TCP/IP protocol suite in relationship to the OSI reference model.[1] The network interface layer, which corresponds to the OSI physical and data link layers, is not actually part of the specification. However, it has become a de facto layer either as shown in Figure 1-1 or as separate physical and data link layers. It is described in this section in terms of the OSI physical and data link layers.

[1] The OSI protocol suite itself has become, with some rare exceptions, a relic of early Internet history. Its current contribution to networking technology seems to be mainly limited to the usefulness of its reference model in illustrating modular protocol suites to networking studentsand, of course, the IS-IS routing protocol still widely used in large service provider and carrier networks.

Figure 1-1. TCP/IP protocol suite.

<table>
<tr><th>OSI</th><th>TCP/IP</th></tr>
<tr><td>APPLICATION</td><td rowspan="3">APPLICATION</td></tr>
<tr><td>PRESENTATION</td></tr>
<tr><td>SESSION</td></tr>
<tr><td>TRANSPORT</td><td>HOST-TO-HOST</td></tr>
<tr><td>NETWORK</td><td>INTERNET</td></tr>
<tr><td>DATA LINK</td><td rowspan="2">NETWORK INTERFACE</td></tr>
<tr><td>PHYSICAL</td></tr>
</table>

The *physical layer* contains the protocols relating to the physical medium on which TCP/IP will be communicating. Officially, the protocols of this layer fall within four categories that together describe all aspects of physical media:

- *Electrical/optical* protocols describe signal characteristics such as voltage or photonic levels, bit timing, encoding, and signal shape.
- *Mechanical* protocols are specifications such as the dimensions of a connector or the metallic makeup of a wire.
- *Functional* protocols describe *what* something does. For example, "Request to Send" is the functional description of pin 4 of an EIA-232-D connector.
- *Procedural* protocols describe *how* something is done. For example, a binary 1 is represented on an EIA-232-D lead as a voltage more negative than 3 volts.

The *data link layer* contains the protocols that control the physical layer: how the medium is accessed and shared, how devices on the medium are identified, and how data is framed before being transmitted on the medium. Examples of data-link protocols are IEEE 802.3/Ethernet, Frame Relay, ATM, and SONET.

The *internet layer*, corresponding to the OSI network layer, is primarily responsible for enabling the routing of data across logical network paths by defining a packet format and an addressing format. This layer is, of course, the one with which this book is most concerned.

The *host-to-host layer*, corresponding to the OSI transport layer, specifies the protocols that control the internet layer, much as the data link layer controls the physical layer. Both the host-to-host and data link layers can define such mechanisms as flow and error control. The difference is that while data-link protocols control traffic on the data linkthe physical medium connecting two devicesthe transport layer controls traffic on the logical linkthe end-to-end connection of two devices whose logical connection traverses a series of data links.

The *application layer* corresponds to the OSI session, presentation, and application layers. Although some routing protocols such as Border Gateway Protocol (BGP) and routing Information Protocol (RIP) reside at this layer,[2] the most common services of the application layer provide the interfaces by which user applications access the network.

[2] BGP is an application layer protocol because it uses TCP to transport its messages, and RIP because it uses UDP for the same purposes. Other routing protocols such as OSPF are said to operate at the internet layer because they encapsulate their messages directly into IP packets.

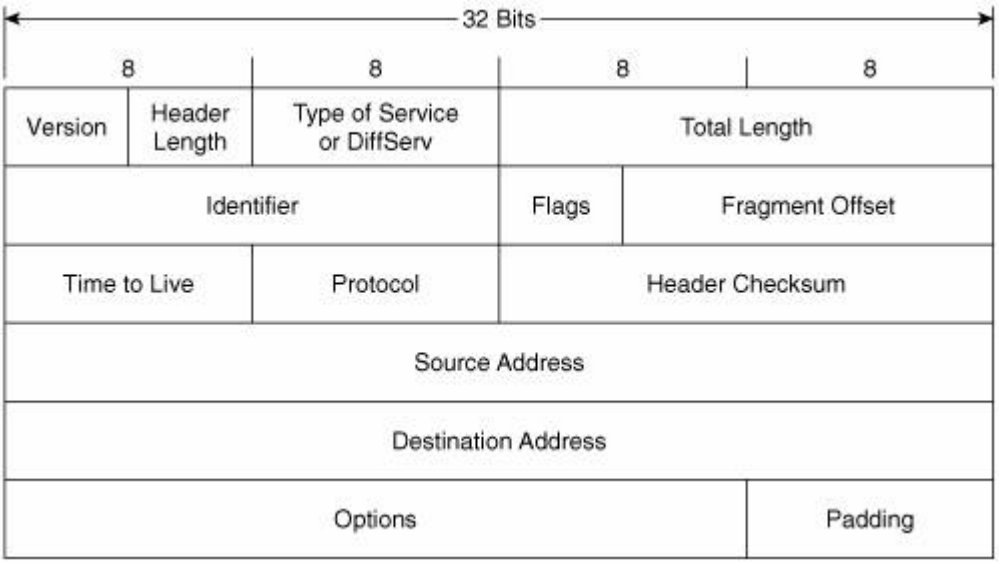
A function common to the protocol suite of Figure 1-1 and any other protocol suite is multiplexing between layers. Many applications might use a service at the host-to-host layer, and many services at the host-to-host layer might use the internet layer. Multiple protocol suites (IP, IPX, and AppleTalk, for example) can share a physical link via common data-link protocols.

PREV NEXT

# IP Packet Header

Figure 1-2 shows the format of the IP packet header, specified in RFC 791. Most fields in this packet have some importance to routing.

Figure 1-2. IP packet protocol.




*Version* identifies the IP version to which the packet belongs. This four-bit field is set to binary 0100 to indicate version 4 (IPv4) or binary 0110 to indicate version 6 (IPv6). This chapter is concerned primarily with IPv4, whereas Chapter 2, "IPv6 Overview," focuses on IPv6. Table 1-1 shows all currently assigned version numbers, along with a few of the relevant RFCs. All versions other than 4 and 6 (built on an earlier proposal called Simple Internet Protocol, or SIP, which also carried a version number of 6) now exist only as "culture," and it will be left to the curious to read their cited RFCs.

Table 1-1. IP version numbers.

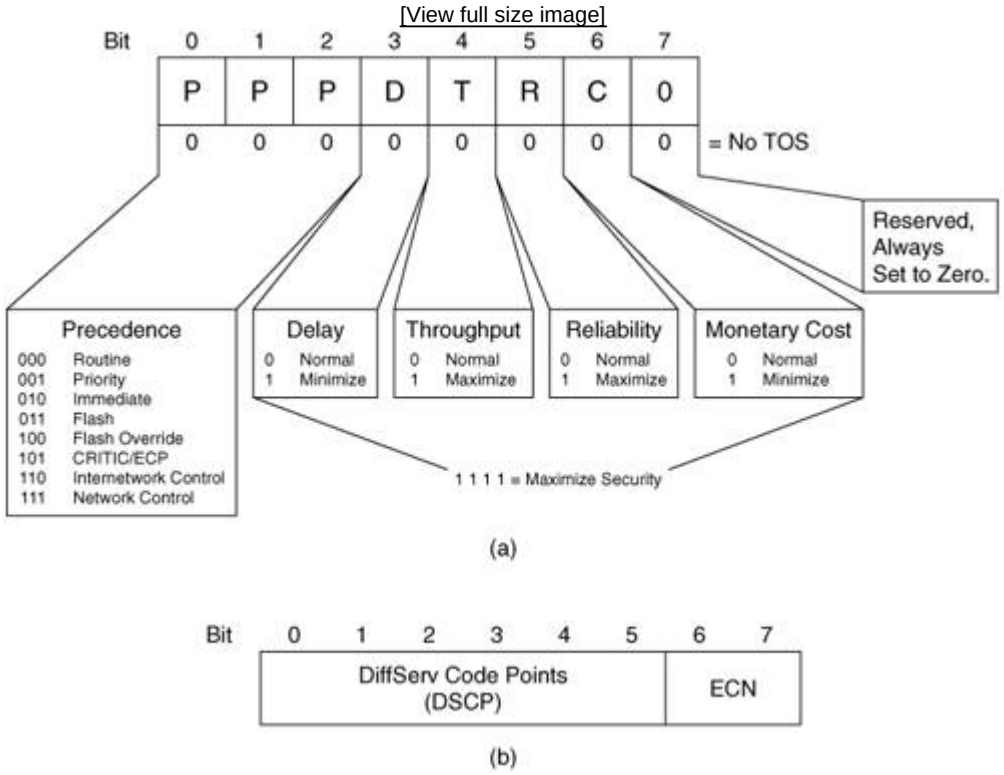
| Number | Version | RFC |
|---|---|---|
| 0 | Reserved | |
| 13 | Unassigned | |
| 4 | Internet Protocol version 4 (IPv4) | 791 |
| 5 | ST Datagram Mode | 1190 |
| 6 | Simple Internet Protocol (SIP) | |
| 6 | Internet Protocol version 6 (IPv6) | 1883 |

| Number | Version | RFC |
|---|---|---|
| 7 | TP/IX | 1475 |
| 8 | P Internet Protocol (PIP) | 1621 |
| 9 | TCP and UDP over Bigger Addresses (TUBA) | 1347 |
| 1014 | Unassigned | |
| 15 | Reserved | |

*Header Length* is a four-bit field that tells, as the name implies, the length of the IP header in 32-bit words. This field is included because the Options field (described later in this section) can vary in size. The minimum length of the IP header is 20 octets, and the options might increase this size up to a maximum of 60 octetsthe maximum length in 32-bit words that can be described by this field.

*Type of Service* (TOS) is an eight-bit field that can be used for specifying special handling of the packet. This field actually can be broken down into two subfields: Precedence and TOS. Precedence sets a priority for the packet, the way a package might be sent overnight, two-day delivery, or general post. TOS allows the selection of a delivery service in terms of throughput, delay, reliability, and monetary cost. Although this field is not commonly used (all the bits will usually be set to zero), early specifications of the Open Shortest Path First (OSPF) protocol called for TOS routing. Also, the Precedence bits are occasionally used in quality of service (QoS) applications. Part (a) of Figure 1-3 summarizes the eight TOS bits; for more information, see RFC 1340 and RFC 1349.

Figure 1-3. Type of Service (a) or DiffServ and ECN (b) field.



In recent years, the ToS field has been redefined as a part of the *Differentiated Services* (DiffServ) framework.[3] This framework creates a much more flexible handling of IP packets than was allowed by the relatively rigid ToS definitions. With DiffServ, you can define service classes on a router and then sort packets into these classes. The router can then queue and forward packets with different levels of priority, according to their classification. Each queuing and forwarding treatment is called a *Per-Hop Behavior* (PHB). While DiffServ defines the framework or architecture, the mechanism itself is called Differentiated Class of Service or simply *Class of Service* (COS).

[3] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, December 1998.

Part (b) of Figure 1-3 shows how the ToS field has been redefined, so that the first six bits now compose the *DiffServ Code Point* (DSCP). With these six bits you can define, either arbitrarily or according to service classes predefined in the DiffServ architecture, up to 64 different service classes that can then be sorted into PHBs. Note that the field in the IP header remains 8 bits; the DiffServ architecture just redefines how a router interprets the value in that field.

*Explicit Congestion Notification* (ECN), in part (b) of Figure 1-3, is used by some routers to signal support for Explicit Congestion Notification and, when it is supported, the bits can be used to signal congestion (ECN = 11).[4]

[4] K. Ramakrishnan, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, September 2001.

*Total Length* is a 16-bit field specifying the total length of the packet, including the header, in octets. By subtracting the header length, a receiver might determine the size of the packet's data payload. Because the largest decimal number that can be described with 16 bits is 65,535, the maximum possible size of an IP packet is 65,535 octets.

*Identifier* is a 16-bit field used in conjunction with the *Flags* and *Fragment Offset* fields for fragmentation of a packet. Packets must be fragmented into smaller packets if the original length exceeds the Maximum Transmission Unit (MTU) of a data link through which they pass. For example, consider a 5000-byte packet traveling through a network. It encounters a data link with a 1500 byte MTU. That is, the frame can contain a maximum packet size of 1500 bytes. The router that places the packet onto this data link must first fragment the packet into chunks of no more than 1500 octets each. The router then marks each fragment with the same number in the Identifier field so that a receiving device can identify the fragments that go together.[5]

[5] A fragmented packet is not reassembled at the other end of the data link; the packet stays fragmented until it reaches its final destination.

*Flags* is a three-bit field in which the first bit is unused. The second is the Don't Fragment (DF) bit. When the DF bit is set to one, a router cannot fragment the packet. If the packet cannot be forwarded without fragmenting, the router drops the packet and sends an error message to the source. This function enables the testing of MTUs in a network. The DF bit can be set using the Extended Ping utility in IOS, as shown in Example 1-1.

### Example 1-1. The IOS Extended Ping utility allows the setting of the DF bit to test MTUs across a network. In this ping Output, the largest MTU of the path to destination 172.16.113.17 is 1478 octets.

```
Handy#ping
Protocol [ip]:
Target IP address: 172.16.113.17
Repeat count [5]: 1
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address:
Type of service [0]:
Set DF bit in IP header? [no]: y
```

```
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: r
Number of hops [9]:
Loose, Strict, Record, Timestamp, Verbose[RV]:
Sweep range of sizes [n]: y
Sweep min size [76]: 500
Sweep max size [18024]: 2000
Sweep interval [1]: 500
Type escape sequence to abort.
Sending 4, [500..2000]-byte ICMP Echos to 172.16.113.17, timeout is 2 seconds:
Packet has IP options:  Total option bytes= 39, padded length=40
 Record route: <*> 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
         0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0

Reply to request 0 (16 ms) (size 500).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route: 172.16.192.5 172.16.113.18 172.16.113.17 172.16.113.17
         172.16.192.6 172.16.192.5 <*> 0.0.0.0 0.0.0.0 0.0.0.0
 End of list

Reply to request 1 (24 ms) (size 1000).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route: 172.16.192.5 172.16.113.18 172.16.113.17 172.16.113.17
         172.16.192.6 172.16.192.5 <*> 0.0.0.0 0.0.0.0 0.0.0.0
 End of list

Reply to request 2 (28 ms) (size 1500).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route: 172.16.192.5 172.16.113.18 172.16.113.17 172.16.113.17
         172.16.192.6 172.16.192.5 <*> 0.0.0.0 0.0.0.0 0.0.0.0
 End of list

Unreachable from 172.16.192.6, maximum MTU 1478 (size 2000).
 Received packet has options
 Total option bytes= 39, padded length=40
 Record route: <*> 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
         0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0

Success rate is 75 percent (3/4), round-trip min/avg/max = 16/22/28 ms
Handy#
```

The third bit is the More Fragments (MF) bit. When a router fragments a packet, it sets the MF bit to one in all but the last fragment so that the receiver knows to keep expecting fragments until it encounters a fragment with MF = 0.

*Fragment Offset* is a 13-bit field that specifies the offset, in units of eight octets, from the beginning of the header to the beginning of the fragment.[6] Because fragments might not always arrive in sequence, the Fragment Offset field allows the pieces to be reassembled in the correct order.

[6] Units of eight octets are used so that a maximum-size packet of 65,535 bytes might be described with 13 bits.

Note that if a single fragment is lost during a transmission, the entire packet must be resent and refragmented at the same point in the network. Therefore, error-prone data links could cause a disproportionate delay. And if a fragment is lost because of congestion, the retransmission of the entire series of fragments might increase the congestion.

*Time to Live* (TTL) is an eight-bit field that will be set with a certain number when the packet is first generated. As the packet is passed from router to router, each router will decrement this

number. If the number reaches zero, the packet will be discarded and an error message will be sent to the source. This process prevents "lost" packets from wandering endlessly through a network.

As originally conceived, the TTL was specified in seconds; if a packet was delayed more than a second in a router, the router would adjust the TTL accordingly. However, this approach is difficult to implement and has never been generally supported. Modern routers simply decrement the TTL by one, no matter what the actual delay, so the TTL is really a hop count.[7] The recommended default TTL is 64, although values such as 15 and 32 are not uncommon.

[7] As you will read in Chapter 2, the equivalent field in the IPv6 header has been renamed Hop Limit to more accurately reflect its true usage.

Some trace utilities, such as the IOS `trace` command, make use of the TTL field. If the router is told to trace the route to a host address such as 10.11.12.13, the router will send three packets with the TTL set to one; the first router will decrement it to zero, drop the packets, and send back error messages to the source. By reading the source address of the error messages, the first router on the path is now known. The next three packets will be sent with a TTL of two. The first router decrements to one, the second to zero, and an error message is received from the second router. The third set has a TTL of three, and so forth, until the destination is found. All routers along the network path will have identified themselves. Example 1-2 shows the output from an IOS trace.

### Example 1-2. The trace utility uses the TTL field to identify routers along a route. Asterisks indicate timed-out packets.

```
Elvis#traceroute www.cisco.com

Type escape sequence to abort.
Tracing the route to cio-sys.Cisco.COM (192.31.7.130)

  1 172.18.197.17 4 msec   4 msec 4 msec
  2 ltlrichard-s1-13.hwy51.com (172.18.197.1) 36 msec 44 msec  2536 msec
  3 cperkins-rtr-fr2.hwy51.com (10.168.204.3) 104 msec 64 msec *
  4 cberry.hwy51.com (10.168.193.1)  92 msec *
  5 jllewis-inner.hwy51.com (10.168.207.59) 44 msec  *  44 msec
  6 bholly-fw-outer-rt.hwy51.com (10.168.207.94) 44 msec *   48 msec
  7 sl-stk-14-S10/0:6-512k.sprintlink.net (144.228.214.107) 92 msec *
  8 sl-stk-2-F1/0/0.sprintlink.net  (144.228.40.2) 52 msec 1156 msec *
  9 sl-mae-w-H1/0-T3.sprintlink.net  (144.228.10.46) 100 msec 124 msec 2340 msec
 10 sanjose1-br1.bbnplanet.net  (198.32.136.19) 2264 msec   164 msec *
 11 paloalto-br2.bbnplanet.net  (4.0.1.10) 64 msec 60 msec    *
 12 su-pr2.bbnplanet.net  (131.119.0.218) 76 msec 76 msec    76 msec
 13 cisco.bbnplanet.net  (131.119.26.10) 2560 msec 76 msec    936 msec
 14 sty.cisco.com  (192.31.7.39)  84 msec 72 msec  *
 15 cio-sys.Cisco.COM  (192.31.7.130) 60 msec  *   64 msec
Elvis#
```

*Protocol* is an eight-bit field that gives the "address," or protocol number, of the host-to-host or transport layer protocol for which the information in the packet is destined. Table 1-2 shows a few of the more common of the 100 or so different protocol numbers currently assigned.

### Table 1-2. A few well-known protocol numbers.

| Protocol Number | Host-to-Host Layer Protocol |
|---|---|
| 1 | Internet Control Message Protocol (ICMP) |
| 2 | Internet Group Management Protocol (IGMP) |
| 4 | IP in IP (encapsulation) |
| 6 | Transmission Control Protocol (TCP) |
| 17 | User Datagram Protocol (UDP) |
| 45 | Inter-Domain Routing Protocol (IDRP) |
| 46 | Resource Reservation Protocol (RSVP) |
| 47 | Generic Routing Encapsulation (GRE) |
| 54 | NBMA Next Hop Resolution Protocol (NHRP) |
| 88 | Cisco Internet Gateway Routing Protocol (IGRP) |
| 89 | Open Shortest Path First (OSPF) |

*Header Checksum* is the error detection field for the IP header. The checksum is not calculated for the encapsulated data; UDP, TCP, and ICMP have their own checksums for doing this. The field contains a 16-bit one's complement checksum, calculated by the originator of the packet. The receiver will again calculate a 16-bit one's complement sum, including the original checksum. If no errors have occurred during the packet's travels, the resulting checksum will be all ones. Remember that each router decrements the TTL; therefore, the checksum must be recalculated at each router. RFC 1141 discusses some strategies for simplifying this calculation.

*Source* and *Destination Addresses* are the 32-bit IP addresses of the originator of the packet and the destination of the packet. The format of IP addresses is covered in the next section, "IPv4 Addresses."

*Options* is a variable-length field and, as the name says, is optional. Space is added to the packet header to contain either source-generated information or for other routers to enter information; the options are used primarily for testing. The most frequently used options are

- *Loose source routing*, in which a series of IP addresses for router interfaces is listed. The packet must pass through each of these addresses, although multiple hops might be taken between the addresses.
- *Strict source routing*, where again a series of router addresses is listed. Unlike loose source routing, the packet must follow the route exactly. If the next hop is not the next address on the list, an error occurs.
- *Record route* provides room for each router to enter the address of its outgoing interface as the packet transits so that a record is kept of all routers the packet encounters. Record route provides a function similar to *trace* except that the outgoing interfaces, both on the path to the destination and on the return path, are recorded.
- *Timestamp* is an option similar to record route except each router also enters a timestamp: the packet not only keeps track of where it has been but also records when it was there.

All these options might be invoked by using the Extended Ping on Cisco routers. Record route is used in Example 1-1, loose source routing and timestamp are used in Example 1-3, and strict

source routing is used in Example 1-4.

Example 1-3. The Cisco Extended Ping can be used to set parameters in the Options field of the IP header. In this example, loose source routing and timestamp are used.

```
Handy#ping
Protocol [ip]:
Target IP address: 172.16.113.9
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address:
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: l
Source route: 172.16.113.14 172.16.113.10
Loose, Strict, Record, Timestamp, Verbose[LV]: t
Number of timestamps [ 6 ]: 2
Loose, Strict, Record, Timestamp, Verbose[LTV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.113.9, timeout is 2 seconds:
Packet has IP options: Total option bytes= 23, padded length=24
  Loose source route: <*> 172.16.113.14 172.16.113.10
  Timestamp: Type 0. Overflows: 0 length 12, ptr 5
    >>Current pointer<<
    Time= 0
    Time= 0

Request 0 timed out
Reply to request 1 (76 ms). Received packet has options
  Total option bytes= 24, padded length=24
  Loose source route: 172.16.113.13 172.16.192.6 <*>
  Timestamp: Type 0. Overflows: 6 length 12, ptr 13
    Time= 80FF4798
    Time= 80FF4750
    >>Current pointer<<
  End of list

Request 2 timed out
Reply to request 3 (76 ms). Received packet has options
  Total option bytes= 24, padded length=24
  Loose source route: 172.16.113.13 172.16.192.6 <*>
  Timestamp: Type 0. Overflows: 6 length 12, ptr 13
  Time= 80FF4FC0
  Time= 80FF4F78
  >>Current pointer<<
End of list

Request 4 timed out
Success rate is 40 percent (2/5), round-trip min/avg/max = 76/76/76 ms
Handy#
```

Example 1-4. Extended Ping is used here to set strict source routing in

the ping packets.

```
Handy#ping
Protocol [ip]:
Target IP address: 172.16.113.10
Repeat count [5]: 2
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address:
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]: s
Source route: 172.16.192.6 172.16.113.17 172.16.113.10
Loose, Strict, Record, Timestamp, Verbose[SV]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 2, 100-byte ICMP Echos to 172.16.113.10, timeout is 2 seconds:
Packet has IP options: Total option bytes= 15, padded length=16
  Strict source route: <*> 172.16.192.6 172.16.113.17 172.16.113.10

Reply to request 0 (80 ms). Received packet has options
  Total option bytes= 16, padded length=16
  Strict source route: 172.16.113.10 172.16.113.17 172.16.192.6 <*>
  End of list

Reply to request 1 (76 ms). Received packet has options
  Total option bytes= 16, padded length=16
  Strict source route: 172.16.113.10 172.16.113.17 172.16.192.6 <*>
  End of list

Success rate is 100 percent (2/2), round-trip min/avg/max = 76/78/80 ms
Handy#
```

*Padding* ensures that the header ends on a 32-bit boundary by adding zeros after the option field until a multiple of 32 is reached.

A protocol analyzer capture of an IP header is shown in Example 1-5. Compare the information shown with Figure 1-2.

## Example 1-5. You can see the fields of an IP packet's header and the values contained in each field in this protocol analyzer display.

```
Internet Protocol, Src Addr: 172.16.1.102 (172.16.1.102), Dst Addr: 224.0.0.5
(224.0.0.5)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00)
    Total Length: 64
    Identification: 0x6e61 (28257)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 1
    Protocol: OSPF IGP (0x59)
```

# IPv4 Addresses

IPv4 addresses are 32 bits long; like all network-level addresses, they have a network portion and a host portion. The network portion uniquely identifies a physical or logical link and is common to all devices attached to that link. The host portion uniquely identifies a particular device attached to the link.

There are several ways to represent the 32 bits of an IP address. For instance, the 32-bit IP address

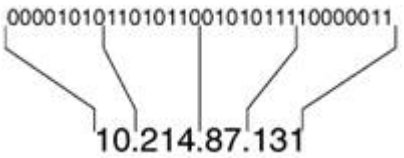
00001010110101100101011110000011

can be represented in decimal as

181,819,267.

The binary format is cumbersome, and a decimal format of the entire 32-bit number is time-consuming to calculate. Figure 1-4 shows a better format.

Figure 1-4. The dotted-decimal format is a convenient way to write IPv4 addresses, but it should not be confused with what the router (or host) sees: a 32-bit string.




The 32 bits of the address comprise four octets, each of which can be represented with a decimal number between 0 and 255, with dots between the decimal representations. In Figure 1-4, the 32-bit address is mapped into a dotted-decimal representation.[8]

[8] Dotted decimal is used only with IPv4 addresses. As you will read in Chapter 2, IPv6 addresses are represented entirely differently.

An important distinction to remember when working with IPv4 addresses is that dotted decimal is just an easy way for humans to read and write IP addresses. Always remember that the router is not reading an address in terms of four octets; rather, the router sees a 32-bit binary string. Many pitfalls can be avoided by keeping this fact firmly in mind. If you have not worked with binary numbersparticularly converting between binary and decimalyou might want to read the tutorial in Appendix A, "Tutorial: Working with Binary and Hex," before continuing on with this chapter.

Probably the most distinctive characteristic of IPv4 addresses is that unlike other network-level addresses, the network and host portions can vary in size within the 32-bit boundaries. That is, the network portion might take up most of the 32 bits, or the host portion might, or they might divide the bits equally. Protocols such as NetWare and AppleTalk were designed for use in relatively small networks, and as a result their network-level addresses have fixed-length network and host portions. This arrangement certainly makes life easier; a receiving device knows to read a certain number of bits into the address to find the network part, and the rest is

host address.

TCP/IP, however, was designed from the first to be flexible enough to be used in any network, from the tiny to the colossal. This flexibility makes IP addresses more difficult to manage. The basics of administering IP addresses are presented in this section, and then some more advanced techniques are introduced in Chapter 6, "RIPv2, RIPng, and Classless Routing."

## First Octet Rule

Without putting too fine a point on it, it can be said that there are three sizes of networks as measured by the number of hosts: big, medium, and small:

- Big networks, by definition, have a huge number of hosts. Relatively few big networks exist.
- Small networks are just the opposite. Each one is small because it has a small number of hosts; a huge number of small networks exist.
- Medium networks are just that: a medium number of them (in relation to big and small ones) and a medium number of hosts in each one.

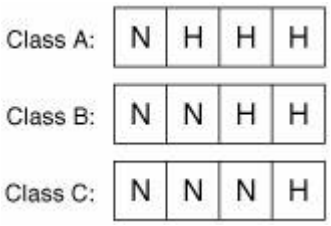
This high level of addressing focus requires three types*classes*of network address for the three sizes of networks. Addresses for big networks need to be capable of addressing many hosts, but because so few big networks exist, only a few big-network addresses are required.

The situation is reversed for small networks. Because there are many small networks, a large number of small-network addresses are needed. But because a small network has a small number of hosts, each of the many network addresses only requires a few host addresses.

For medium-sized networks, a medium number of network addresses and a medium number of host addresses will be available for each network address.

Figure 1-5 shows how the network and host portions of IPv4 addresses are divided up for these three classes.

Figure 1-5. Class A, B, and C IPv4 address formats.




The big, medium, and small networks described thus far map to address classes as follows:

- *Class A* IPv4 addresses are for big networks. The first octet is the network portion, and the last three octets are the host portion. Only 256 numbers are available in the eight-bit network part, but $2^{24}$ or 16,777,216 numbers are available in the host part of each of those network addresses.
- *Class B* addresses are for medium-size networks. The first two octets are the network portion, and the last two octets are the host portion. There are $2^{16}$ or 65,536 available

numbers in the network part and an equal number in the host part.

- *Class C* addresses are just the opposite of Class A. The first three octets are the network portion, and the last octet is the host portion.

Because all IPv4 addresses are 32-bit binary strings, a way of distinguishing the class to which a particular address belongs is necessary. The *first octet rule*, demonstrated in Table 1-3, provides the means to make such a distinction and can be described as follows:

- For Class A addresses, the first bit of the first octetthat is, the left-most bit of the entire 32-bit stringis always set to zero. Therefore, we can find the minimum and maximum numbers in the Class A range by setting all the remaining bits in the first octet to zero (for the minimum) and one (for the maximum). This action results in the decimal numbers 0 and 127 with a few exceptions: 0 is reserved as part of the default address (Chapter 12, "Default Routes and On-Demand Routing"), and 127 is reserved for internal loopback addresses.[9] That leaves 1 through 126; any IP address whose first octet is between 1 and 126 inclusive is a Class A address.

  [9] Devices use loopback addresses (typically 127.0.0.1) to send traffic to themselves. Data might be sent to this address and returned to the transmitting process without ever leaving the device.

- Class B addresses always have their left-most bit set to one and the second bit set to zero. Again, finding the minimum and maximum number of the first octet by setting all remaining bits to zero and then to one, you see in Figure 1-4 that any address whose first octet is in the decimal range 128 through 191 is a Class B address.

- In Class C addresses, the first two bits are set to one, and the third bit is set to zero. The result is a first octet range of 192 through 223.[10]

  [10] Notice that 223 does not exhaust all available numbers in the first octet. See Configuration Exercise 1 at the end of this chapter.

Table 1-3. First octet rule.

| Rule | Minimum and Maximum | Decimal Range |
|---|---|---|
| Class A: First bit is always 0 | 00000000 = 0<br>01111111 = 127 | 1126[*] |
| Class B: First two bits are always 10 | 10000000 = 128<br>10111111 = 191 | 128191 |
| Class C: First three bits are always 110 | 11000000 = 192<br>11011111 = 223 | 192223 |

[*] 0 and 127 are reserved

So far IPv4 addressing doesn't seem so difficult. A router or host could easily determine the network part of an IP address by using the first octet rule. If the first bit is 0, then read the first eight bits to find the network address. If the first two bits are 10, then read the first 16 bits; and if the first three bits are 110, then read 24 bits in to get the network address. Unfortunately, things are not that easy.

## Address Masks

The address for an entire data linka non-host-specific network addressis represented by the network portion of an IP address, with all host bits set to zero. For instance, an addressing authority[11] might assign to an applicant an address of 172.21.0.0.[12] This address is a Class B address because 172 is between 128 and 191, so the last two octets make up the host bits. Notice that they are all set to zero. The first 16 bits (172.21.) are assigned, but address owners are free to do whatever they please with the host bits.

[11] The high-level organizations responsible for managing and assigning IP addresses are APNIC in Asia, ARIN in North America, LACNIC in Central and South America, and RIPE in EMEA.

[12] Actually, this address would never be assigned. It is from a group of addresses reserved for private use; most of the addresses used in this book are from this reserved pool, described in RFC 1918. Reserved addresses are 10.0.0.010.255.255.255, 172.16.0.0172.31.255.255, and 192.168.0.0192.168.255.255.

Each device or interface will be assigned a unique, host-specific address such as 172.21.35.17. The device, whether a host or a router, obviously needs to know its own address, but it also needs to be able to determine the network to which it belongsin this case, 172.21.0.0.

This task is accomplished by means of an *address mask*. The address mask is a 32-bit string, one bit for each bit of the IPv4 address. As a 32-bit string, the mask can be represented in dotted-decimal format just like an IPv4 address. This representation tends to be a stumbling block for some beginners: Although the address mask can be written in dotted decimal, it is not an address. Table 1-4 shows the standard address masks for the three classes of IPv4 address.

Table 1-4. Address masks for Class A, B, and C IPv4 addresses.

| Class | Mask | Dotted Decimal |
|---|---|---|
| A | 11111111000000000000000000000000 | 255.0.0.0 |
| B | 11111111111111110000000000000000 | 255.255.0.0 |
| C | 11111111111111111111111100000000 | 255.255.255.0 |

For each bit of the IPv4 address, the device performs a Boolean (logical) AND function with the corresponding bit of the address mask. The AND function can be stated as follows:

> Compare two bits and derive a result. The result will be one, if and only if, both bits are one. If either or both bits are zero, the result will be zero.

Figure 1-6 shows how, for a given IPv4 address, the address mask is used to determine the network address. The mask has a one in every bit position corresponding to a network bit of the address and a zero in every bit position corresponding to a host bit. Because 172.21.35.17 is a Class B address, the mask must have the first two octets set to all ones and the last two octets, the host part, set to all zeros. As Table 1-4 shows, this mask can be represented in dotted decimal as 255.255.0.0.

Figure 1-6. Each bit of this Class B address is ANDed with the corresponding bit of the address mask to derive the network address.

**Truth Table for Boolean AND**

| | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

```
      10101100000101010010001100010001 = 172.21.35.17
AND   11111111111111110000000000000000 = 255.255.0.0
      10101100000101010000000000000000 = 172.21.0.0
```

A logical AND is performed on the IPv4 address and its mask for every bit position; the result is shown in Figure 1-6. In the result, every network bit is repeated, and all the host bits become 0s. So by assigning an address of 172.21.35.17 and a mask of 255.255.0.0 to an interface, the device will know that the interface belongs to network 172.21.0.0. Applying the AND operator to an IPv4 address and its address mask always reveals the network address.

An address and mask are assigned to an interface of a Cisco router (in this example, the E0 interface) by means of the following commands:

```
Smokey(config)# interface ethernet 0
Smokey(config-if)# ip address 172.21.35.17 255.255.0.0
```

But why use address masks at all? So far, using the first octet rule seems much simpler.

## Subnets and Subnet Masks

Never lose sight of why network-level addresses are necessary in the first place. For routing to be accomplished, each and every data link (network) must have a unique address; in addition, each and every host on that data link must have an address that both identifies it as a member of the network and distinguishes it from any other host on that network.

As defined so far, a single Class A, B, or C address can be used only on a single data link. To build a network, separate addresses must be used for each data link so that those networks are uniquely identifiable. If a separate Class A, B, or C address were assigned to each data link, fewer than 17 million data links could be addressed before all IPv4 addresses were depleted. This approach is obviously impractical,[13] as is the fact that to make full use of the host address space in the previous example, more than 65,000 devices would have to reside on data link 172.21.0.0!

[13] Seventeen million data links might seem like a lot until you consider that even a single moderate-size business might have dozens or hundreds of data links.

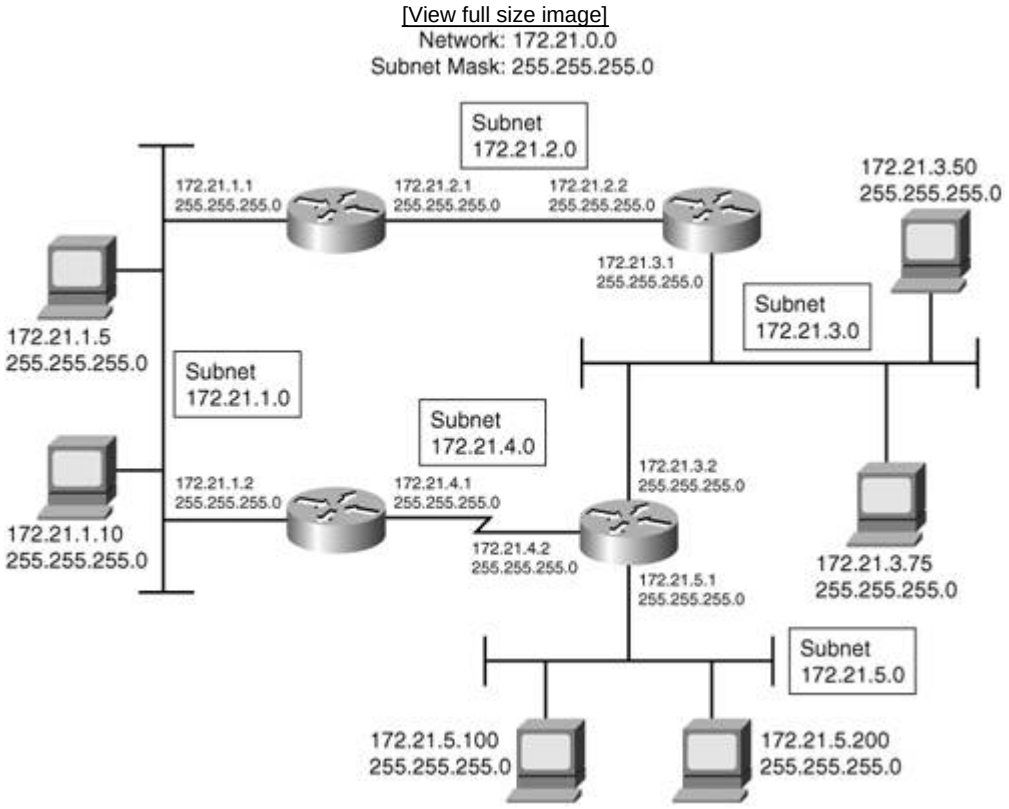
The only way to make Class A, B, or C addresses practical is by dividing each major address, such as 172.21.0.0, into subnetwork addresses. Recall two facts:

- The host portion of an IPv4 address can be used as desired.
- The network portion of an IPv4 address is determined by the address mask assigned to that interface.

Figure 1-7 shows a network to which the major Class B address 172.21.0.0 has been assigned. Five data links are interconnecting the hosts and routers, each one of which requires a network

address. As it stands, 172.21.0.0 would have to be assigned to a single data link, and then four more addresses would have to be requested for the other four data links.

Figure 1-7. Subnet masks allow a single network address to be used on multiple data links by "borrowing" some of the host bits for use as subnet bits.




Notice what was done in Figure 1-7. The address mask is not a standard 16-bit mask for Class B addresses; the mask has been extended another eight bits so that the first 24 bits of the IP address are interpreted as network bits. In other words, the routers and hosts have been given a mask that causes them to read the first eight host bits as part of the network address. The result is that the major network address applies to the entire network, and each data link has become a subnetwork, or subnet. A *subnet* is a subset of a major Class A, B, or C address space.

The IPv4 address now has three parts: the network part, the subnet part, and the host part. The address mask is now a *subnet mask*, or a mask that is longer than the standard address mask. The first two octets of the address will always be 172.21, but the third octetwhose bits are now subnet bits instead of host bitsmight range from 0 to 255. The network in Figure 1-6 has subnets 1, 2, 3, 4, and 5 (172.21.*1*.0 through 172.21.*5*.0). Up to 256 subnets might be assigned under the single Class B address, using the mask shown.

Two words of caution are in order. First, not all routing protocols can support subnet addresses in which the subnet bits are all zeros or all ones. The reason is that these protocols, called *classful* protocols, cannot differentiate between an all-zero subnet and the major network number. For instance, subnet 0 in Figure 1-7 would be 172.21.0.0; the major IP address is also 172.21.0.0. The two cannot be distinguished without further information.

Likewise, classful routing protocols cannot differentiate a broadcast on the all-ones subnet from an all-subnets broadcast address.[14] For example, the all-ones subnet in Figure 1-7 would be 172.21.255.0. For that subnet, the all-hosts broadcast address would be 172.21.255.255, but

that is also the broadcast for all hosts on all subnets of major network 172.21.0.0. Again, the two addresses cannot be distinguished without further information. RIP version 1 and IGRP are both classful routing protocols; Chapter 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)," introduces *classless* routing protocols, which can indeed use the all-zeros and all-ones subnets.

[14] The all-hosts IP broadcast address is all ones: 255.255.255.255. An all-hosts broadcast for a particular subnet would set all host bits to one; for instance, an all-hosts broadcast for subnet 172.21.1.0 would be 172.21.1.255. Finally, a broadcast for all hosts on all subnets sets the subnet bits and the host bits to all ones: 172.21.255.255.

The second caution has to do with the verbal description of subnets and their masks. Subnetting the third octet of a Class B address, as is done is Figure 1-7, is very common; also common is hearing people describe such a subnet design as "using a Class C mask with a Class B address," or "subnetting a Class B address into a Class C." Both descriptions are wrong! Such descriptions frequently lead to misunderstandings about the subnet design or to a poor understanding of subnetting itself. The proper way to describe the subnetting scheme of Figure 1-6 is either as "a Class B address with 8 bits of subnetting," or as "a Class B address with a 24-bit mask."

The subnet mask might be represented in any of the following three formats:

Dotted decimal: 255.255.255.0

Bitcount: 172.21.0.0/24

Hexadecimal: 0xFFFFFF00

Dotted decimal is commonly used in software that has been around for a while, although the bitcount format is becoming increasingly preferred. Compared to dotted decimal, the bitcount format is easier to write. (The address is followed by a forward slash and the number of bits that are masked for the network part.) In addition, the bitcount format is more descriptive of what the mask is really doing and therefore avoids the type of semantic misunderstandings described in the previous paragraph. Some UNIX systems use the hexadecimal format.

Although the address mask must be specified to Cisco routers in dotted decimal, using the command shown previously, the mask might be displayed by various `show` commands in any of the three formats by using the command `ip netmask-format [decimal| hexadecimal| bit-count]` in line configuration mode. For example, to configure a router to display its masks in bitcount format, use

```
Gladys(config)# line vty 0 4
Gladys(config-line)# ip netmask-format bit-count
```

## Designing Subnets

As established in the previous section, subnet bits cannot be all zeros or all ones in classful environments. Likewise, an IPv4 host address cannot have all its host bits set to zerothis setting is reserved for the address that routers use to represent the network or subnet itself. And the host bits cannot be set to all ones, as this setting is the broadcast address. These restrictions apply to the host bits with no exceptions and are starting points for designing subnets. Beyond these starting points, network designers need to choose the most appropriate subnetting scheme in terms of matching the address space to the particulars of a network.

When designing subnets and their masks, the number of available subnets under a major network address and the number of available hosts on each subnet are both calculated with the same formula: $2^n$ 2, where *n* is the number of bits in the subnet or host space and 2 is subtracted to account for the unavailable all-zeros and all-ones addresses. For example, given a Class A address of 10.0.0.0, a subnet mask of 10.0.0.0/16 (255.255.0.0) means that the 8-bit subnet space will yield $2^8$ 2 = 254 available subnets and $2^{16}$ 2 = 65,534 host addresses available on each of those subnets. On the other hand, a mask of 10.0.0.0/24 (255.255.255.0) means that a 16-bit subnet space is yielding 65,534 subnets and an 8-bit host space is yielding 254

host addresses for each subnet.

The following steps are used to subnet an IPv4 address:

Step 1. Determine how many subnets are required and how many hosts per subnet are required.

Step 2. Use the $2^n$ 2 formula to determine the number of subnet bits and the number of host bits that will satisfy the requirements established in Step 1. If multiple subnet masks can satisfy the requirements, choose the one that will best scale to future needs. For example, if the network is most likely to grow by adding subnets, choose more subnet bits; if the network is most likely to grow by adding hosts to existing subnets, choose more host bits. Avoid choosing a scheme in which either all subnets or all host addresses within the subnets will be used up immediately, leaving no room for future growth.

Step 3. Working in binary, determine all available bit combinations in the subnet space; in each instance, set all the host bits to zero. Convert the resulting subnet addresses to dotted decimal. These are the subnet addresses.

Step 4. For each subnet address, again working in binary, write all possible bit combinations for the host space without changing the subnet bits. Convert the results to dotted decimal; these are the host addresses available for each subnet.

The importance of doing the last two steps in binary cannot be overemphasized. The single greatest source of mistakes when working with subnets is trying to work with them in dotted decimal *without understanding what is happening at the binary level*. Again, dotted decimal is for convenience in reading and writing IPv4 addresses. Routers and hosts see the addresses as 32-bit binary strings; to successfully work with these addresses, they must be seen the way the routers and hosts see them.

The previous paragraph might seem a bit overzealous in light of the examples given so far; the patterns of subnet and host addresses have been quite apparent without having to see the addresses and masks in binary. The next section uses the four design steps to derive a subnet design in which the dotted-decimal representations are not so obvious.

## Breaking the Octet Boundary

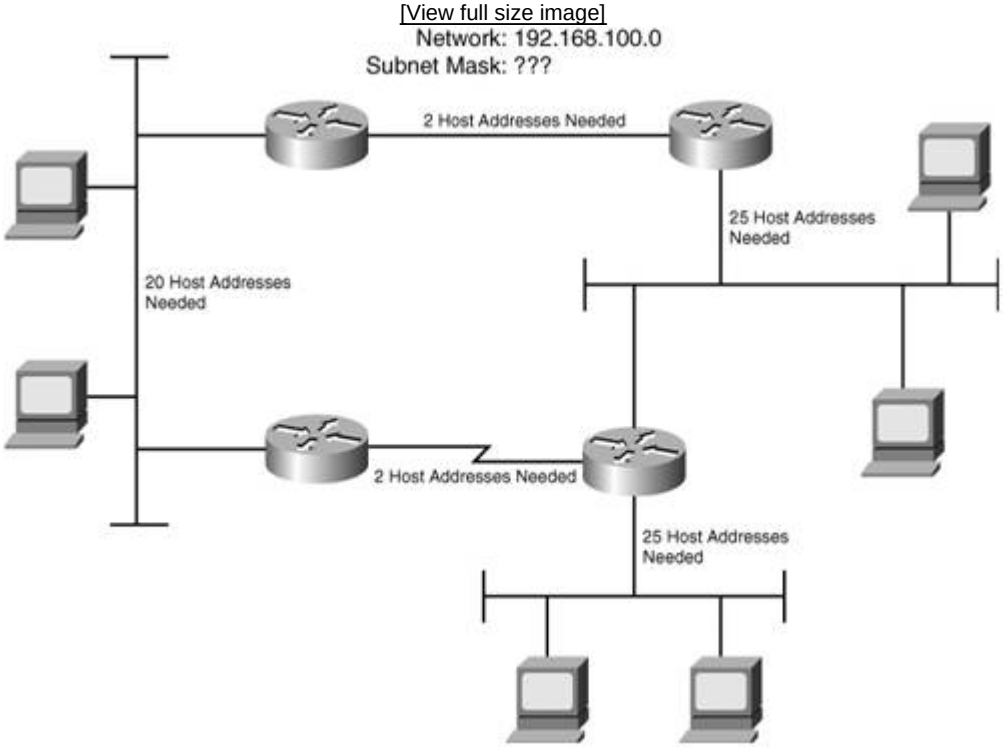
In the examples given so far, the subnet spaces have fallen on octet boundaries. This arrangement is not always the most practical or efficient choice. What if, for instance, you need to subnet a Class B address across 500 data links, each with a maximum of 100 hosts? This requirement is easily met, but only by using nine bits in the subnet field: $2^9$ 2 = 510 available subnets, leaving seven bits for the host field, and $2^7$ 2 = 126 available hosts per subnet. No other bit combination will satisfy this requirement.

Notice, also, that there is no way to subnet a class C address on an octet boundarydoing so would use up all of the last byte, leaving no room for host bits. The subnet bits and host bits must share the last octet, as the following example shows.

Figure 1-8 shows the network of Figure 1-7 but with a Class C address of 192.168.100.0 assigned.

Figure 1-8. The network from Figure 1-7 but with a Class C prefix assigned. Subnetting an entire octet will not work here; there would be no space left for host bits.

[View full size image]

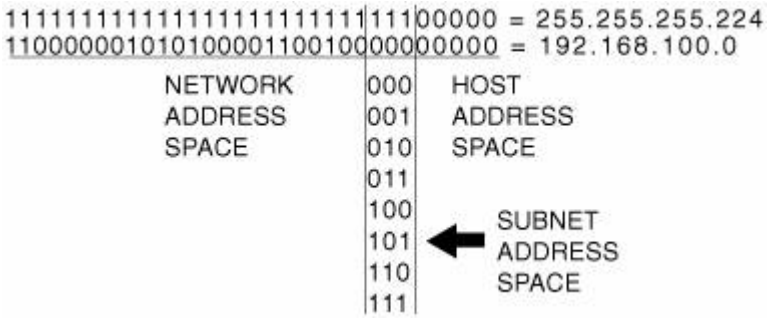



There are five data links; therefore, the address must be subnetted to provide for at least five subnet addresses. The illustration also indicates the number of hosts (including router interfaces) that need to be addressed on each subnet. The maximum host address requirement is 25 for the two Ethernets. Therefore, the full subnetting requirements are at least five subnets and at least 25 host addresses per subnet.

Applying the $2^n$ 2 formula, three subnet bits and five host bits will satisfy the requirements: $2^3$ 2 = 6 and $2^5$ 2 = 30. A Class C mask with three bits of subnetting is represented as 255.255.255.224 in dotted decimal.

Figure 1-9 shows the derivation of the subnet bits. The subnet mask derived in Step 2 is written in binary, and the IP address is written below it. Vertical lines are drawn as markers for the subnet space, and within this space all possible bit combinations are written by counting up from zero in binary.

Figure 1-9. The subnet bits are derived by marking the masked subnet bit space and then writing all possible bit combinations in the space by counting up from zero in binary.



In Figure 1-10, the unchanged network bits are filled in to the left of the subnet space and the host bits, which are all zeros in the subnet addresses, are filled in to the right of the subnet space. The results are converted to dotted decimal, and these are the six subnet addresses (remembering that the first and last addresses, which have 000 and 111 in the subnet space, cannot be used).

Figure 1-10. The subnet addresses are derived by filling in the network address to the left of the subnet space, setting all host bits to zero to the right of the subnet space, and converting the results to dotted decimal.

```
11111111111111111111111111100000 = 255.255.255.224
11000000101010000110010000000000 = 192.168.100.0
11000000101010000110010000000000 = 192.168.100.0
11000000101010000110010000100000 = 192.168.100.32
11000000101010000110010001000000 = 192.168.100.64
11000000101010000110010001100000 = 192.168.100.96
11000000101010000110010010000000 = 192.168.100.128
11000000101010000110000010100000 = 192.168.100.160
11000000101010000110000011000000 = 192.168.100.192
11000000101010000110000011100000 = 192.168.100.224
```

The last step is to calculate the host addresses available to each subnet. This step is done by choosing a subnet and, keeping the network and subnet bits unchanged, writing all bit combinations in the host space by counting up from zero in binary. Figure 1-11 shows this step for subnet 192.168.100.32.

Figure 1-11. The host addresses for a subnet are derived by writing all possible bit combinations in the host space. These are the host bits for subnet 192.168.100.32.

```
NETWORK      HOST
BITS         BITS
11000000101010000110010000100000 = 192.168.100.32 ← SUBNET
11000000101010000110010000100001 = 192.168.100.33
11000000101010000110010000100010 = 192.168.100.34
11000000101010000110010000100011 = 192.168.100.35
11000000101010000110010000100100 = 192.168.100.36
11000000101010000110010000100101 = 192.168.100.37
11000000101010000110010000100110 = 192.168.100.38
11000000101010000110010000100111 = 192.168.100.39
11000000101010000110010000101000 = 192.168.100.40
11000000101010000110010000101001 = 192.168.100.41
11000000101010000110010000101010 = 192.168.100.42
11000000101010000110010000101011 = 192.168.100.43
11000000101010000110010000101100 = 192.168.100.44
11000000101010000110010000101101 = 192.168.100.45
11000000101010000110010000101110 = 192.168.100.46
11000000101010000110010000101111 = 192.168.100.47
11000000101010000110010000110000 = 192.168.100.48
11000000101010000110010000110001 = 192.168.100.49
11000000101010000110010000110010 = 192.168.100.50   VALID
11000000101010000110010000110011 = 192.168.100.51   HOST
11000000101010000110010000110100 = 192.168.100.52   ADDRESSES
11000000101010000110010000110101 = 192.168.100.53
11000000101010000110010000110110 = 192.168.100.54
11000000101010000110010000110111 = 192.168.100.55
11000000101010000110010000111000 = 192.168.100.56
11000000101010000110010000111001 = 192.168.100.57
11000000101010000110010000111010 = 192.168.100.58
11000000101010000110010000111011 = 192.168.100.59
11000000101010000110010000111100 = 192.168.100.60
11000000101010000110010000111101 = 192.168.100.61
11000000101010000110010000111110 = 192.168.100.62
11000000101010000110010000111111 = 192.168.100.63 ← BROADCAST
```

Notice the patterns in the results: The first address, in which the host bits are all zero, is the subnet address. The last address, in which the host bits are all one, is the broadcast address for subnet 192.168.100.32. The host addresses count up from the subnet address to the broadcast address, and if the sequence were to continue, the next address would be the second subnet, 192.168.100.64.

The importance of understanding subnetting at the binary level should now be clear. Presented with an address such as 192.168.100.160, you cannot be sure whether it is a host address, a subnet address, or a broadcast address. Even when the subnet mask is known, things are not always readily apparent.

Readers are encouraged to calculate all host addresses for all the remaining subnets in the example and to observe the patterns that result in the addresses. Understanding these patterns will help in situations such as the one presented in the next section.

## Troubleshooting a Subnet Mask

The necessity frequently arises to "dissect" a given host address and mask, usually to identify the subnet to which it belongs. For instance, if an address is to be configured on an interface, a good practice is to first verify that the address is valid for the subnet to which the subnet is connected.

Use the following steps to reverse-engineer an IP address:

Step 1. Write the given subnet mask in binary.

Step 2. Write the IPv4 host address in binary.

Step 3. Knowing the class of the host address, the subnet bits of the mask should be apparent. Using the mask bits as a guide, draw a line between the last network bit and the first subnet bit of the address. Draw another line between the last subnet bit and the first host bit.

Step 4. Write the network and subnet bits of the address, setting all host bits to zero. The result is the address of the subnet to which the host address belongs.

Step 5. Again write the network and subnet bits of the address, this time setting all host bits to one. The result is the broadcast address of the subnet.

Step 6. Knowing that the subnet address is the first address in the sequence and that the broadcast address is the last address in the sequence, you also know that all addresses between these two are valid host addresses.

Figure 1-12 shows these steps applied to 172.30.0.141/25.

Figure 1-12. Given an IPv4 address and a subnet mask, follow these steps to find the subnet, the broadcast, and the host addresses.

172.30.0.141/25

| | |
|---|---|
| (1) Write subnet mask: | 11111111111111111111111110000000 = 255.255.255.128 |
| (2) Write IP address: | 10101100000111100000000010001101 = 172.30.0.141 |

| | |
|---|---|
| (3) Mark the subnet space. | 11111111111111111111111110000000 = 255.255.255.128 |
| | 10101100000111100000000010001101 = 172.30.0.141 |

| | |
|---|---|
| Derive the... | 11111111111111111111111110000000 = 255.255.255.128 |
| | 10101100000111100000000010001101 = 172.30.0.141 |
| (4) subnet address: | 10101100000111100000000010000000 = 172.30.0.128 |
| (5) broadcast address: | 10101100000111100000000011111111 = 172.30.0.255 |

(6) Valid host addresses for this subnet are 172.30.0.129 – 172.30.0.254.

The address is a Class B, so it is known that the first 16 bits are the network bits; therefore, the last nine bits of the 25-bit mask mark the subnet space. The subnet address is found to be 172.30.0.128, and the broadcast address is 172.30.0.255. Knowing that the valid host addresses for the subnet are bounded by these two addresses, it is determined that the host addresses for subnet 172.30.0.128 are 172.30.0.129 through 172.30.0.254.

Several things about this example tend to bother folks who are new to subnetting. Some are bothered by the third octet of the address, which is all zeros. Some are bothered by the single subnet bit in the last octet. Some think that the broadcast address looks suspiciously invalid. All of these uneasy feelings arise from reading the addresses in dotted decimal. When the addresses and the mask are seen in binary, these suspicions are assuaged and everything is seen to be legitimate; the mask sets a nine-bit subnet spaceall of the third octet, and the first bit of the fourth octet. The moral of the story is that if everything is known to be correct in binary, don't

worry if the dotted-decimal representation looks funny.

PREV NEXT

# Address Resolution Protocol (ARP)

Routers pass packets across a logical path, composed of multiple data links, by reading and acting on the network addresses in the packets. The packets are passed across the individual data links by encapsulating the packets in frames, which use data-link identifiers (MAC addresses, for example) to get the frame from source to destination on the link. One of the major topics of this book concerns the mechanisms by which routers discover and share information about network addresses so that routing might take place. Similarly, devices on a data link need a way to discover their neighbors' data-link identifiers so that frames might be transmitted to the correct destination.

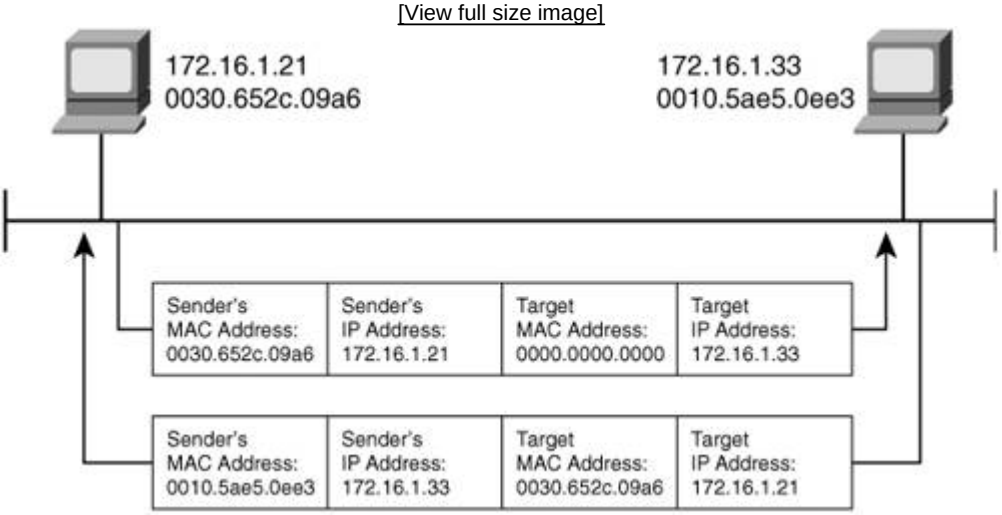
Several mechanisms can provide this information;[15] IPv4 uses the Address Resolution Protocol (ARP), described in RFC 826. Figure 1-13 shows how ARP works. A device needing to discover the data-link identifier of another device will create an ARP Request packet. This request will contain the IPv4 address of the device in question (the target) and the source IPv4 address and data-link identifier (MAC address) of the device making the request (the sender). The ARP Request packet is then encapsulated in a frame with the sender's MAC address as the source and a broadcast address for the destination (see Example 1-6).[16]

[15] NetWare, for example, makes the MAC address of the device the host portion of the network-level addressa very sensible thing to do.

[16] Like an IP broadcast, the MAC broadcast is an address of all ones: ffff.ffff.ffff.

Figure 1-13. ARP is used to map a device's data-link identifier to its IP address.

[View full size image]




Example 1-6. An analyzer capture of the ARP Request depicted in Figure 1-13, with its encapsulating frame.

```
Ethernet II, Src: 00:30:65:2c:09:a6, Dst: ff:ff:ff:ff:ff:ff
    Destination: ff:ff:ff:ff:ff:ff (Broadcast)
    Source: 00:30:65:2c:09:a6 (AppleCom_2c:09:a6)
    Type: ARP (0x0806)
```

```
Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: 00:30:65:2c:09:a6 (AppleCom_2c:09:a6)
    Sender IP address: 172.16.1.21 (172.16.1.21)
    Target MAC address: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Target IP address: 172.16.1.33 (172.16.1.33)
```

The broadcast address means that all devices on the data link will receive the frame and examine the encapsulated packet. All devices except the target will recognize that the packet is not for them and will drop the packet. The target will send an ARP Reply to the source address, supplying its MAC address (see Example 1-7).

## Example 1-7. An analyzer capture of the ARP Reply depicted in Figure 1-13.

```
Ethernet II, Src: 00:10:5a:e5:0e:e3, Dst: 00:30:65:2c:09:a6
    Destination: 00:30:65:2c:09:a6 (AppleCom_2c:09:a6)
    Source: 00:10:5a:e5:0e:e3 (3com_e5:0e:e3)
    Type: ARP (0x0806)
    Trailer: 151515151515151515151515151515...
Address Resolution Protocol (reply)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    Sender MAC address: 00:10:5a:e5:0e:e3 (3com_e5:0e:e3)
    Sender IP address: 172.16.1.33 (172.16.1.33)
    Target MAC address: 00:30:65:2c:09:a6 (AppleCom_2c:09:a6)
    Target IP address: 172.16.1.21 (172.16.1.21)
```

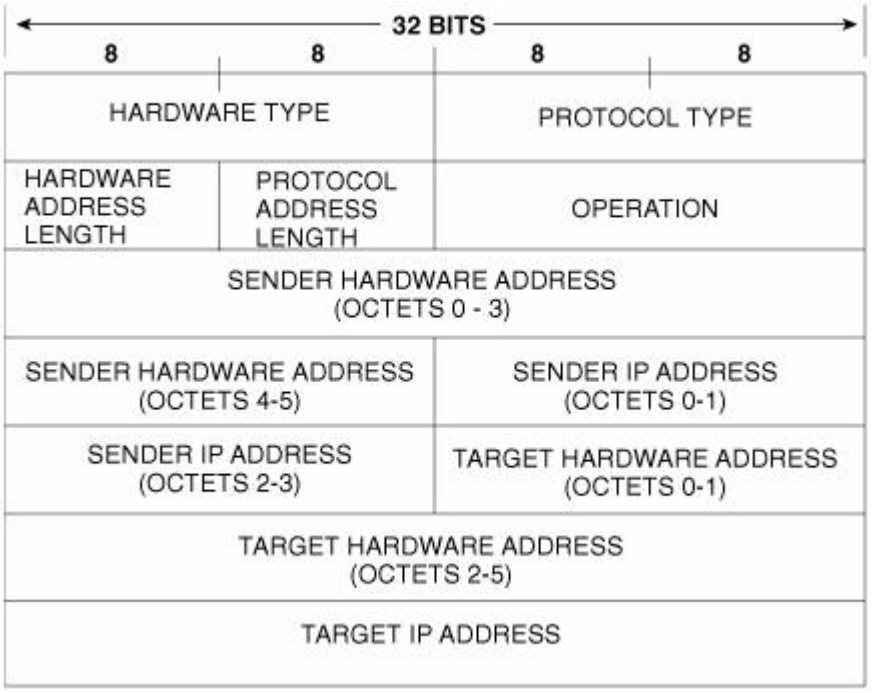
Cisco routers will display ARP activity when the debug function `debug arp` is invoked, as shown in Example 1-8.

## Example 1-8. Router Aretha (172.21.5.1) responds to an ARP request from host 172.19.35.2.

```
Aretha#debug arp
IP ARP: rcvd req src 172.19.35.2 0002.6779.0f4c, dst 172.21.5.1 Ethernet0
IP ARP: sent rep src 172.21.5.1 0000.0c0a.2aa9,
                 dst 172.19.35.2 0002.6779.0f4c Ethernet0
Aretha#
```

Figure 1-14 shows the ARP packet format. As the fields are described, compare them with the ARP packets in Example 1-6 and Example 1-7.

## Figure 1-14. ARP packet format.




*Hardware Type* specifies the type of hardware, as specified by the IETF.[17] Table 1-5 shows some examples of some of the more common type numbers.

[17] All numbers in use in various fields throughout the TCP/IP protocol suite were originally listed in: J. Postel and J. Reynolds, "Assigned Numbers," RFC 1700, October 1994. This large document (230 pages) is a valuable reference, but is now a bit outdated. A current list of assigned numbers can be found at www.iana.org.

Table 1-5. Common hardware type codes.

| Number | Hardware Type |
|---|---|
| 1 | Ethernet |
| 3 | X.25 |
| 4 | Proteon ProNET Token Ring |
| 6 | IEEE 802 Networks |
| 7 | ARCnet |
| 11 | Apple LocalTalk |
| 14 | SMDS |
| 15 | Frame Relay |
| 16 | ATM |
| 17 | HDLC |
| 18 | Fibre Channel |
| 19 | ATM |
| 20 | Serial Link |

*Protocol Type* specifies the type of network-level protocol the sender is mapping to the data link identifier; IPv4 is 0x0800.

*Hardware Address Length* specifies the length, in octets, of the data link identifiers. MAC addresses would be 6.

*Protocol Address Length* specifies the length, in octets, of the network-level address. IPv4 would be 4.

*Operation* specifies whether the packet is an ARP Request (1) or an ARP Reply (2). Other values might also be found here, indicating other uses for the ARP packet. Examples are Reverse ARP Request (3), Reverse ARP Reply (4), Inverse ARP Request (8), and Inverse ARP Reply (9).

The final 20 octets are the fields for the sender's and target's data-link identifiers and IPv4 addresses.

In the top screen in Example 1-9, the IOS command `show arp` is used to examine the ARP table in a Cisco router.

## Example 1-9. The ARP table for three devices connected to the same network: a Cisco router, a Microsoft Windows host, and a Linux host.

```
Martha#show arp
Protocol  Address          Age (min)     Hardware Addr  Type  Interface
Internet  10.158.43.34           2     0002.6779.0f4c  ARPA  Ethernet0
Internet  10.158.43.1            -     0000.0c0a.2aa9  ARPA  Ethernet0
Internet  10.158.43.25          18     00a0.24a8.a1a5  ARPA  Ethernet0
Internet  10.158.43.100          6     0000.0c0a.2c51  ARPA  Ethernet0
Martha#
```

---

```
C:\WINDOWS>arp -a

Interface: 148.158.43.25
  Internet Address      Physical Address     Type
  10.158.43.1           00-00-0c-0a-2a-a9    dynamic
  10.158.43.34          00-02-67-79-0f-4c    dynamic
  10.158.43.100         00-00-0c-0a-2c-51    dynamic
```

---

```
Linux:~# arp -a
Address                 HW type          HW address           Flags   Mask
10.158.43.1             10Mbps Ethernet  00:00:0C:0A:2A:A9    C       *
10.158.43.100           10Mbps Ethernet  00:00:0C:0A:2C:51    C       *
10.158.43.25            10Mbps Ethernet  00:A0:24:A8:A1:A5    C       *
Linux:~#
```

Notice the Age column. As this column would indicate, ARP information is removed from the table after a certain time to prevent the table from becoming congested with old information. Cisco routers hold ARP entries for four hours (14,400 seconds); this default can be changed. The following example changes the ARP timeout to 30 minutes (1800 seconds):

```
Martha(config)# interface ethernet 0
Martha(config-if)# arp timeout 1800
```

The middle screen of Example 1-9 shows the ARP table of a Microsoft Windows PC, and the bottom shows the ARP table from a Linux machine. Although the format is different from the IOS display, the essential information is the same in all three tables.

ARP entries might also be permanently placed in the table. To statically map 172.21.5.131 to hardware address 0000.00a4.b74c, with a SNAP (Subnetwork Access Protocol) encapsulation type, use the following:

```
Martha(config)# arp 172.21.5.131 0000.00a4.b74c snap
```

The command clear arp-cache forces a deletion of all dynamic entries from the ARP table. It also clears the fast-switching cache and the IP route cache.

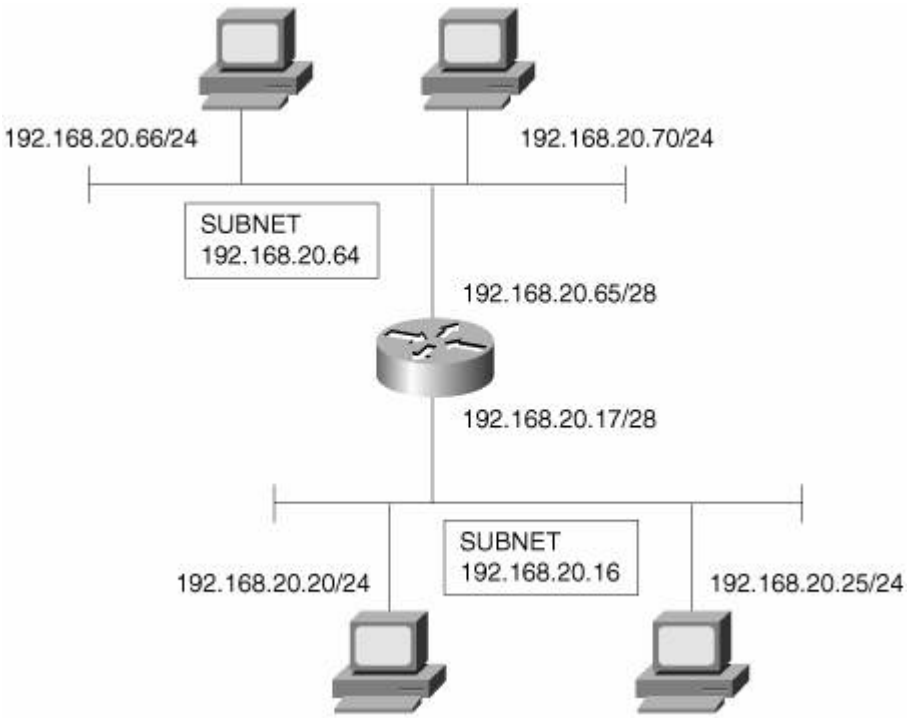
Several variations of ARP exist; at least one, proxy ARP, is important to routing.

## Proxy ARP

Sometimes called *promiscuous ARP* and described in RFCs 925 and 1027, proxy ARP is a method by which routers might make themselves available to hosts. For example, a host 192.168.12.5/24 needs to send a packet to 192.168.20.101/24, but it is not configured with default gateway information and therefore does not know how to reach a router. It might issue an ARP Request for 192.168.20.101; the local router, receiving the request and knowing how to reach network 192.168.20.0, will issue an ARP Reply with its own data link identifier in the hardware address field. In effect, the router has tricked the local host into thinking that the router's interface is the interface of 192.168.20.101. All packets destined for that address are then sent to the router.

Figure 1-15 shows another use for proxy ARP. Of particular interest here are the address masks. The router is configured with a 28-bit mask (four bits of subnetting for the Class C address), but the hosts are all configured with 24-bit, default Class C mask. As a result, the hosts will not be aware that subnets exist. Host 192.168.20.66, wanting to send a packet to 192.168.20.25, will issue an ARP Request. The router, recognizing that the target address is on another subnet, will respond with its own hardware address. Proxy ARP makes the subnetted network topology transparent to the hosts.

Figure 1-15. Proxy ARP enables the use of transparent subnets.



The ARP cache in Example 1-10 gives a hint that proxy ARP is in use. Notice that multiple IPv4 addresses are mapped to a single MAC identifier; the addresses are for hosts, but the hardware MAC identifier belongs to the router interface.

### Example 1-10. This ARP table from host 192.168.20.66 in Figure 1-15 shows multiple IPv4 addresses mapped to one MAC identifier, indicating that proxy ARP is in use.

```
C:\WINDOWS>arp -a

Interface: 192.168.20.66
  Internet Address      Physical Address      Type
  192.168.20.17         00-00-0c-0a-2a-a9     dynamic
  192.168.20.20         00-00-0c-0a-2a-a9     dynamic
  192.168.20.25         00-00-0c-0a-2a-a9     dynamic
  192.168.20.65         00-00-0c-0a-2c-51     dynamic
  192.168.20.70         00-02-67-79-0f-4c     dynamic
```

Proxy ARP is enabled by default in IOS and might be disabled on a per interface basis with the command `no ip proxy-arp`.

## Gratuitous ARP

A host might occasionally issue an ARP Request with its own IPv4 address as the target address. These ARP Requests, known as *gratuitous* ARPs, have several uses:

- A gratuitous ARP might be used for duplicate address checks. A device that issues an ARP Request with its own IPv4 address as the target and receives an ARP Reply from another device will know that the address is a duplicate.
- A gratuitous ARP might be used to advertise a new data-link identifier. This use takes advantage of the fact that when a device receives an ARP Request for an IPv4 address that is already in its ARP cache, the cache will be updated with the sender's new hardware address.
- A router running Hot Standby Router Protocol (HSRP) that has just taken over as the active router from another router on a subnet issues a gratuitous ARP to update the ARP caches of the subnet's hosts.

Many IP implementations do not use gratuitous ARP, but you should be aware of its existence. It is disabled by default in IOS but can be enabled with the command `ip gratuitous-arps`.

## Reverse ARP

Instead of mapping a hardware address to a known IPv4 address, Reverse ARP (RARP) maps an IPv4 address to a known hardware address. Some devices, such as diskless workstations, might not know their IPv4 address at startup. RARP might be programmed into firmware on these devices, allowing them to issue an ARP Request that has their burned-in hardware address. The reply from a RARP server will supply the appropriate IPv4 address.

RARP has been largely supplanted by Dynamic Host Configuration Protocol (DHCP), an extension

# Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol, or ICMP, described in RFC 792, specifies a variety of messages whose common purpose is to manage the network. ICMP messages might be classified as either error messages or queries and responses. Figure 1-16 shows the general ICMP packet format. The packets are identified by type; many of the packet types have more specific types, and these are identified by the code field. Table 1-6 lists the various ICMP packet types and their codes, as described in RFC 1700.

Figure 1-16. The ICMP packet header includes a type field, a code field that further identifies some types, and a checksum. The rest of the fields depend on the type and code.

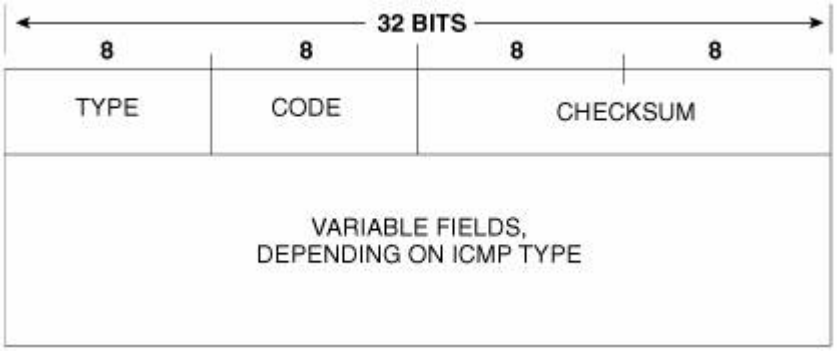



Table 1-6. ICMP packet types and code fields.

| Type | Code | Name |
|---|---|---|
| 0 | 0 | ECHO REPLY |
| 3 | | DESTINATION UNREACHABLE |
| | 0 | Network Unreachable |
| | 1 | Host Unreachable |
| | 2 | Protocol Unreachable |
| | 3 | Port Unreachable |
| | 4 | Fragmentation Needed and Don't Fragment Flag Set |
| | 5 | Source Route Failed |
| | 6 | Destination Network Unknown |
| | 7 | Destination Host Unknown |
| | 8 | Source Host Isolated |
| | 9 | Destination Network Administratively Prohibited |

| Type | Code | Name |
|---|---|---|
| | 10 | Destination Host Administratively Prohibited |
| | 11 | Destination Network Unreachable for Type of Service |
| | 12 | Destination Host Unreachable for Type of Service |
| 4 | 0 | SOURCE QUENCH (deprecated) |
| 5 | | REDIRECT |
| | 0 | Redirect Datagram for the Network (or Subnet) |
| | 1 | Redirect Datagram for the Host |
| | 2 | Redirect Datagram for the Network and Type of Service |
| | 3 | Redirect Datagram for the Host and Type of Service |
| 6 | 0 | ALTERNATE HOST ADDRESS |
| 8 | 0 | ECHO |
| 9 | 0 | ROUTER ADVERTISEMENT |
| 10 | 0 | ROUTER SELECTION |
| 11 | | TIME EXCEEDED |
| | 0 | Time to Live Exceeded in Transit |
| | 1 | Fragment Reassembly Time Exceeded |
| 12 | | PARAMETER PROBLEM |
| | 0 | Pointer Indicates the Error |
| | 1 | Missing a Required Option |
| | 2 | Bad Length |
| 13 | 0 | TIMESTAMP |
| 14 | 0 | TIMESTAMP REPLY |
| 15 | 0 | INFORMATION REQUEST (Obsolete) |
| 16 | 0 | INFORMATION REPLY (Obsolete) |
| 17 | 0 | ADDRESS MASK REQUEST (Near-obsolete) |
| 18 | 0 | ADDRESS MASK REPLY (Near-obsolete) |
| 30 | - | TRACEROUTE |

Example 1-11 and Example 1-12 show analyzer captures of two of the most well-known ICMP messagesEcho Request and Echo Reply, which are used by the ping function.

## Example 1-11. ICMP Echo message, shown with its IPv4 header.

```
Internet Protocol, Src Addr: 172.16.1.21 (172.16.1.21),
    Dst Addr: 198.133.219.25 (198.133.219.25)
```

```
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 84
    Identification: 0xabc3 (43971)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (0x01)
    Header checksum: 0x8021 (correct)
    Source: 172.16.1.21 (172.16.1.21)
    Destination: 198.133.219.25 (198.133.219.25)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xa297 (correct)
    Identifier: 0x0a40
    Sequence number: 0x0000
    Data (56 bytes)

0000  40 fd ab c2 00 0e 73 57 08 09 0a 0b 0c 0d 0e 0f   @.....sW........
0010  10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0020  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f    !"#$%&'()*+,-./
0030  30 31 32 33 34 35 36 37                            01234567
```

## Example 1-12. ICMP Echo Reply.

```
Internet Protocol, Src Addr: 198.133.219.25 (198.133.219.25),
    Dst Addr: 172.16.1.21 (172.16.1.21)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 84
    Identification: 0xabc3 (43971)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 242
    Protocol: ICMP (0x01)
    Header checksum: 0xce20 (correct)
    Source: 198.133.219.25 (198.133.219.25)
    Destination: 172.16.1.21 (172.16.1.21)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xaa97 (correct)
    Identifier: 0x0a40
    Sequence number: 0x0000
    Data (56 bytes)

0000  40 fd ab c2 00 0e 73 57 08 09 0a 0b 0c 0d 0e 0f  @.....sW........
0010  10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f  ................
0020  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f   !"#$%&'()*+,-./
0030  30 31 32 33 34 35 36 37                           01234567
```

Although most ICMP types have some bearing on routing functionality, three types are of particular importance:

- *Router Advertisement* and *Router Selection*, types 9 and 10, respectively, are used by the ICMP Router Discovery Protocol (IRDP), a protocol used by some operating systems (such as most versions of Microsoft Windows) to discover local routers.

- *Redirect*, ICMP type 5, is used by routers to notify hosts of another router on the data link that should be used for a particular destination. Suppose two routers, Router A and Router B, are connected to the same Ethernet. Host X, also on the Ethernet, is configured to use Router A as its default gateway; the host sends a packet to Router A, and A sees that the destination address of the packet is reachable via Router B (that is, Router A must forward the packet out the same interface on which it was received). Router A forwards the packet to B but also sends an ICMP redirect to host X informing it that in the future, to reach that particular destination, X should forward the packet to Router B. Example 1-13 shows a router sending a redirect.

### Example 1-13. Using the debugging function *debug ip icmp*, this router can be seen sending a redirect to host 10.158.43.25, informing it that the correct router for reaching destination 10.158.40.1 is reachable via gateway (gw) 10.158.43.10.

```
Pip#debug ip icmp
ICMP packet debugging is on
ICMP: redirect sent to 10.158.43.25 for dest 10.158.40.1, use gw 10.158.43.10
0
Pip#
```

An occasionally used trick to avoid redirects on data links with multiple attached gateways is to set each host's default gateway as its own IPv4 address. The hosts will then ARP for any address, and if the address is not on the data link, the correct router should respond via proxy ARP. The benefits of using this tactic merely to avoid redirects are debatable; redirects are

PREV NEXT

# Host-to-Host Layer

The host-to-host layer of the TCP/IP protocol is aptly named. Whereas the internet layer is responsible for the logical paths between networks, the host-to-host layer is responsible for the full logical path between two hosts on disparate networks.[18] From another viewpoint, the host-to-host layer is an interface to the lower layers of the protocol suite, freeing applications from any concern about how their data is actually being delivered.

[18] Similarly, it can be said that the equivalent functions of the OSI session layer, residing above the transport layer, provide a logical, end-to-end path between two applications across a network.

An analogy to this service is a corporate mailroom. A package might be given to the mailroom with requirements stated for its delivery (general delivery, overnight). The person making the delivery request does not need to know, and is probably not interested in, the actual mechanics of delivering the package. The mailroom people will arrange for the proper service (postal, FedEx, cross-town bicycle courier) to fulfill the delivery requirements.

The two primary services offered by the host-to-host layer are TCP and UDP.

## TCP

The Transmission Control Protocol, or TCP, described in RFC 793, provides applications with a reliable, connection-oriented service. In other words, TCP provides the appearance of a point-to-point connection.

Point-to-point connections have two characteristics:

- They have only one path to the destination. A packet entering one end of the connection cannot become lost, because the only place to go is the other end.
- Packets arrive in the same order in which they are sent.

TCP provides the appearance of a point-to-point connection, although in reality there is no such connection. The internet layer TCP uses a connectionless, best-effort packet delivery service. The analogy of this is the Postal Service. If a stack of letters is given to the mail carrier for delivery, there is no guarantee that the letters will arrive stacked in the same order, that they will all arrive on the same day, or indeed that they will arrive at all. The Postal Service merely commits to making its best effort to deliver the letters.

Likewise, the internet layer does not guarantee that all packets will take the same route, and therefore there is no guarantee that they will arrive in the same sequence and time intervals as they were sent, or that they will arrive at all.

On the other hand, a telephone call is connection-oriented service. Data must arrive sequentially and reliably, or it is useless. Like a telephone call, TCP must first establish a connection, then transfer data, and then perform a disconnect when the data transfer is complete.

TCP uses three fundamental mechanisms to accomplish a connection-oriented service on top of a connectionless service:

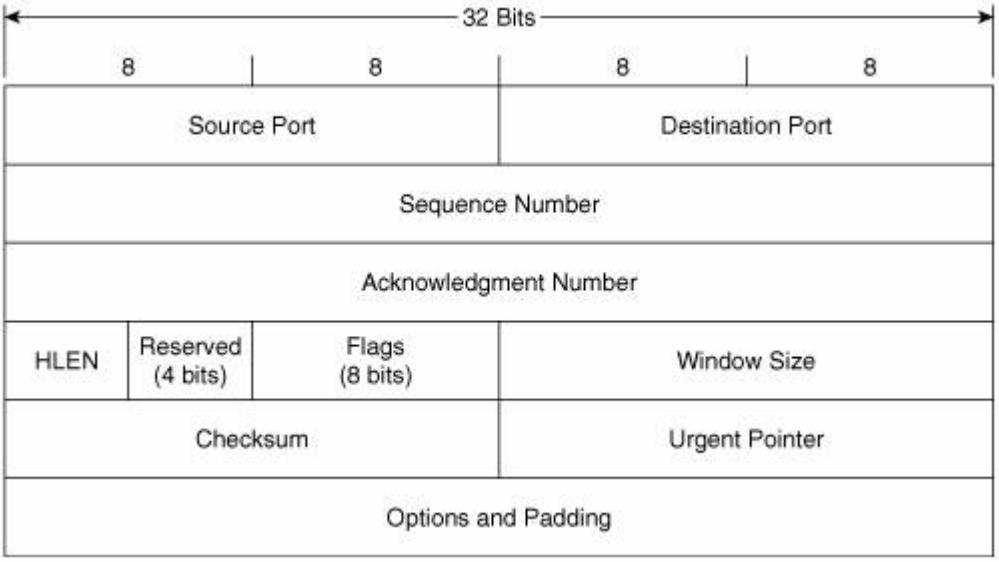
- Packets are labeled with sequence numbers so that the receiving TCP service can put out-of-sequence packets into the correct sequence before delivering them to the destination application.

- TCP uses a system of acknowledgments, checksums, and timers to provide reliability. A receiver might notify a sender when it recognizes that a packet in a sequence has failed to arrive or has errors, or a sender might assume that a packet has not arrived if the receiver does not send an acknowledgment within a certain amount of time after transmission. In both cases, the sender will resend the packet in question.

- TCP uses a mechanism called *windowing* to regulate the flow of packets; windowing decreases the chances of packets being dropped because of full buffers in the receiver.

TCP attaches a header to the application layer data; the header contains fields for the sequence numbers and other information necessary for these mechanisms, and fields for addresses called port numbers, which identify the source and destination applications of the data. The application data with its attached TCP header is then encapsulated within an IP packet for delivery. Figure 1-17 shows the fields of the TCP header, and Example 1-14 shows an analyzer capture of a TCP header.

Figure 1-17. TCP header format.




## Example 1-14. Analyzer display of a TCP header.

```
Ethernet II, Src: 00:0c:41:3c:2b:18, Dst: 00:30:65:2c:09:a6
Internet Protocol, Src Addr: 66.218.71.112 (66.218.71.112),
    Dst Addr: 172.16.1.21 (172.16.1.21)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 52
    Identification: 0xc0b7 (49335)
    Flags: 0x04
    Fragment offset: 0
    Time to live: 50
    Protocol: TCP (0x06)
    Header checksum: 0x509d (correct)
    Source: 66.218.71.112 (66.218.71.112)
    Destination: 172.16.1.21 (172.16.1.21)
Transmission Control Protocol, Src Port: http (80),
    Dst Port: 60190 (60190), Seq: 288, Ack: 811, Len: 0
    Source port: http (80)
```

```
    Destination port: 60190 (60190)
    Sequence number: 288
    Acknowledgement number: 811
    Header length: 32 bytes
    Flags: 0x0010 (ACK)
    Window size: 66608
    Checksum: 0xb32a (correct)
    Options: (12 bytes)
        NOP
        NOP
        Time stamp: tsval 587733966, tsecr 1425164062
    SEQ/ACK analysis
        This is an ACK to the segment in frame: 17
        The RTT to ACK the segment was: 0.047504000 seconds
```

*Source* and *Destination Port* are 16-bit fields that specify the source and destination applications for the encapsulated data. Like other numbers used by TCP/IP, RFC 1700 describes all port numbers in common and not-so-common use. A port number for an application, when coupled with the IP address of the host the application resides on, is called a *socket*. A socket uniquely identifies every application in a network.

*Sequence Number* is a 32-bit number that identifies where the encapsulated data fits within a data stream from the sender. For example, if the sequence number of a segment is 1343 and the segment contains 512 octets of data, the next segment should have a sequence number of 1343 + 512 + 1 = 1856.

*Acknowledgment Number* is a 32-bit field that identifies the sequence number the source next expects to receive from the destination. If a host receives an acknowledgment number that does not match the next sequence number it intends to send (or has sent), it knows that packets have been lost.

*Header Length*, sometimes called *Data Offset*, is a four-bit field indicating the length of the header in 32-bit words. This field is necessary to identify the beginning of the data because the length of the Options field is variable.

The *Reserved* field is four bits, which are always set to zero.

*Flags* are eight 1-bit flags that are used for data flow and connection control. The flags, from left to right, are Congestion Window Reduced (CWR), ECN-Echo (ECE), Urgent (URG), Acknowledgment (ACK), Push (PSH), Reset (RST), Synchronize (SYN), and Final (FIN).

*Window Size* is a 16-bit field used for flow control. It specifies the number of octets, starting with the octet indicated by the Acknowledgment Number, that the sender of the segment will accept from its peer at the other end of the connection before the peer must stop transmitting and wait for an acknowledgment.

*Checksum* is 16 bits, covering both the header and the encapsulated data, allowing error detection.

*Urgent Pointer* is used only when the URG flag is set. The 16-bit number is added to the Sequence Number to indicate the end of the urgent data.

*Options*, as the name implies, specifies options required by the sender's TCP process. The most commonly used option is Maximum Segment Size, which informs the receiver of the largest segment the sender is willing to accept. The remainder of the field is padded with zeros to ensure that the header length is a multiple of 32 octets.

## UDP

User Datagram Protocol, or UDP, described in RFC 768, provides a connectionless, best-effort packet delivery service. At first take, it might seem questionable that any application would prefer an unreliable delivery over the connection-oriented TCP. The advantage of UDP, however, is that no time is spent setting up a connectionthe data is just sent. Applications that send short bursts of data will realize a performance advantage by using UDP instead of TCP.

Figure 1-18 shows another advantage of UDP: a much smaller header than TCP. The Source and Destination Port fields are the same as they are in the TCP header; the UDP length indicates the length of the entire segment in octets. The checksum covers the entire segment, but unlike TCP, the checksum here is optional; when no checksum is used, the field is set to all zeros. Example 1-15 shows an analyzer capture of a UDP header.

Figure 1-18. UDP header format.

## Example 1-15. Analyzer display of a UDP header.

```
Ethernet II, Src: 00:30:65:2c:09:a6, Dst: 00:0c:41:3c:2b:18
Internet Protocol, Src Addr: 172.16.1.21 (172.16.1.21),
    Dst Addr: 198.133.219.25 (198.133.219.25)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 40
    Identification: 0x8a4d (35405)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 1
    Protocol: UDP (0x11)
    Header checksum: 0xe0b3 (correct)
    Source: 172.16.1.21 (172.16.1.21)
    Destination: 198.133.219.25 (198.133.219.25)
User Datagram Protocol, Src Port: 35404 (35404), Dst Port: 33435 (33435)
    Source port: 35404 (35404)
    Destination port: 33435 (33435)
    Length: 20
    Checksum: 0x0000 (none)
Data (12 bytes)

0000 01 01 00 00 40 fd ac 74 00 00 d2 45                 ....@..t...E
```

PREV NEXT

# Looking Ahead

The focus of this chapter has largely been on the mechanisms by which a device's internet layer (or OSI network layer) identifies itself and how it maps to the network interface (or OSI data link) layer. Internet layer protocols such as ARP and ICMP, that are important to routing, were also examined. The following chapter examines a newer version of IP, IP version 6, how it differs from IPv4, and why a new version of IP is needed.

# Summary Table: Chapter 1 Command Review

| Command | Description |
|---|---|
| arp ip-address hardware-address type [alias] | Statically maps an IP address to a hardware address |
| arp timeout *seconds* | Sets the amount of time a Cisco router holds ARP entries |
| clear arp-cache | Forces the deletion of all dynamic entries from the ARP table |
| debug ip icmp | Displays ICMP events as they occur on the router |
| ip address *ip-address mask* [secondary] | Assigns an IP address and mask to an interface |
| ip gratuitous-arp | Enables gratuitous ARP |
| ip netmask-format {bit-count\|decimal\|hexadecimal} | Configures a router to display IP (address, mask) pairs in bitcount, dotted-decimal, or hexadecimal format |
| ip proxy-arp | Enables proxy ARP |
| ip redirects | Enables ICMP redirects |

# Recommended Reading

Baker, F., ed. "Requirements for IP Version 4 Routers," RFC 1812, June 1995. This paper documents both requirements and recommendations for routers that will run IP.

Braden, R., ed. "Requirements for Internet HostsCommunication Layers," RFC 1122, October 1989. The host-centric companion paper to RFC 1812.

Comer, D. E. *Internetworking with TCP/IP*, Vol. 1. Englewood Cliffs, New Jersey: Prentice-Hall; 1991. This book, like Perlman's, is a classic. Although you don't need to read both Comer and Stevens, doing so certainly couldn't hurt.

Stevens, W. R. *TCP/IP Illustrated*, Vol. 1. Reading, Massachusetts: Addison-Wesley; 1994. This is an excellent book on TCP/IP. Along with an in-depth introduction to the protocols, Stevens offers a wealth of captures from a live network that is diagrammed inside the front cover.

# Review Questions

1. What are the five layers of the TCP/IP protocol suite? What is the purpose of each layer?
2. What is the most common IP version presently in use?
3. What is fragmentation? What fields of the IP header are used for fragmentation?
4. What is the purpose of the TTL field in the IP header? How does the TTL process work?
5. What is the first octet rule?
6. How are Class A, B, and C IP addresses recognized in dotted decimal? How are they recognized in binary?
7. What is an address mask, and how does it work?
8. What is a subnet? Why are subnets used in IP environments?
9. Why can't a subnet of all zeros or all ones be used in a classful routing environment?
10. What is ARP?
11. What is proxy ARP?
12. What is a redirect?
13. What is the essential difference between TCP and UDP?
14. What mechanisms does TCP use to provide connection-oriented service?
15. Instead of ARP, Novell NetWare uses a network address that includes a device's MAC address as the host portion. Why can't IP do this?
16. What purpose does UDP serve by providing a connectionless service on top of what is already a connectionless service?

# Configuration Exercises

1. The first octet rule says that the highest Class C address is 223, but it is known that for eight bits the highest decimal number is 255. There are two more classes: Class D addresses are for multicast, and Class E addresses are for experimental usage. Class D addresses have, as their first four bits, 1110. What is the decimal range of the first octet of Class D addresses?

2. Select a subnet mask for 10.0.0.0 so that there will be at least 16,000 subnets with at least 700 host addresses available on each subnet. Select a subnet mask for 172.27.0.0 so that there are at least 500 subnets with at least 100 host addresses available on each subnet.

3. How many subnets are available if a Class C address has six bits of subnetting? How many host addresses are available per subnet? Is there a practical use for such a subnetting scheme?

4. Use a 28-bit mask to derive the available subnets of 192.168.147.0. Derive the available host addresses of each subnet.

5. Use a 29-bit mask to derive the available subnets of 192.168.147.0. Derive the available host addresses of each subnet.

6. Use a 20-bit mask to derive the available subnets of 172.16.0.0. Write the range (that is, the numerically lowest to the numerically highest address) of available host addresses for each subnet.

# Troubleshooting Exercises

1. For the following host addresses and subnet masks, find what subnet each address belongs to, the broadcast address of that subnet, and the range of host addresses for that subnet:

   10.14.87.60/19

   172.25.0.235/27

   172.25.16.37/25

2. You have been told to configure 192.168.13.175 on an interface with a mask of 255.255.255.240. Is there a problem? If so, what is it?

# Chapter 2. IPv6 Overview

This chapter covers the following subjects:

- IPv6 Addresses
- IPv6 Packet Header Format
- Extension Headers
- ICMPv6
- Neighbor Discovery Protocol

When the networks that eventually evolved into what we now call the Internet were first launched, they were the exclusive realm of academics and researchers. And when Vint Cerf and Bob Kahn invented TCP/IP for these networks, no one envisioned the Internet as it now is. At the time a 32-bit address space, yielding almost 4.3 billion addresses, seemed inexhaustible.

But as the kids who worked with these networks in college went out into the "real world," they took with them an appreciation of the possibilities for what could be done with a peer-to-peer network built on open standards. Increasingly useful network applications began cropping up, and recognition of the value of corporate connections to a public network began the push for a commercial Internet. At the same time that all this was happening, desktop computers were becoming common not only in the office but, most significantly, in the home. Yet modems were not a common accessory on those early home computers because few home users saw the value of being connected to a public network.

That changed with the advent of the World Wide Web. Suddenly, easy acquisition and sharing of information exponentially increased the value of desktop computers as a tool for nontechnical users. As a result, in less than 20 years the Internet has changed the way we communicate, do business, and learn. It has made the world a much smaller place, and has had profound impact on world economics and politics.

But this explosion in the size and diversity of the "Internet population" has introduced, along with daily nuisances such as spam and viruses, a serious technical concern: The once inexhaustible supply of IPv4 addresses has become distinctly finite.

The problem of IPv4 address exhaustion was recognized in the early 1990s, when various experts made projections showing that if the increasing rate of the allotment of IPv4 addresses continued, the entire address space could be depleted in just a few short years. A new version of IPknown in the development stage as IP Next Generation or IPng, and which is now IPv6was the proposed solution. But it was recognized that developing the new standards would take time, and that a short-term solution to IPv4 address depletion also was needed.

That short-term solution was *Network Address Translation* (NAT), which allows multiple hosts to share one or a few public IP addresses. Behind the NAT device, private IP addresses as specified in RFC 1918, and which you see in most examples in this book, are used. NAT has been so successful in slowing IPv4 address depletion, and has become such a standard part of most networks, that to this day many still question the need for a new version of IP. But the widespread use of NAT has changed the open, transparent, peer-to-peer Internet into something much more like a huge collection of client-server networks. Users are seen as being connected around the "edge" of the Internet, and services flow out to them. Seldom do users contribute to the overall wealth of the Internet. Seen from a more economic perspective, Internet users have become consumers only, not producers.

Although most of the IPv6 standards were completed years ago, it is only recently that serious interest in migrating from IPv4 to IPv6 has been shown. There are two fundamental drivers behind the growing recognition of the need for IPv6. The first is widespread vision of new applications using core concepts such as mobile IP, service quality guarantees, end-to-end security, grid computing, and peer-to-peer networking. NAT stifles innovation in these areas, and the only way to get NAT out of the way is to make public IP addresses abundant and readily available.

The second fundamental driver for IPv6 is the rapid modernization of heavily populated countries such as India and China. A compelling statistic is that the number of remaining unallocated IPv4 addresses is almost the same as the population of China: about 1.3 billion. With its aggressive expansion of its Internet infrastructure, China alone in the near future will represent an unsupportable pressure on an already strained IPv4 address pool. In India, with a population size close to China's, 4- and 5-layer NAT hierarchies exist just to support the present demands for IP addresses.

IPv6 replaces the 32-bit IPv4 address with a 128-bit address, making 340 *trillion trillion trillion* IP addresses available. That number will meet the demands for public IP addresses, and answer the needs of the two fundamental drivers discussed here, well into the foreseeable future.[1]

[1] Given what was unforeseen when IPv4's 4.3 billion addresses were thought to be limitless for all practical purposes, the almost inconceivably vast IPv6 address space will never be considered inexhaustible.

PREV NEXT

# IPv6 Addresses

IPv6 addresses are different from IPv4 addresses in far more ways than just their length. The "shorthand" for writing them is different, they have significantly different formats, and their functional organization is different. This section introduces you to those differences.

## Address Representation

You certainly already know that 32-bit IPv4 addresses are represented by breaking them into four 8-bit segments and writing each of those segments in decimal between 0 and 255, separating them with periods; hence the term dotted decimal.

128-bit IPv6 addresses are represented by breaking them up into eight 16-bit segments. Each segment is written in hexadecimal between 0x0000 and 0xFFFF, separated by colons. An example of a written IPv6 address is

3ffe:1944:0100:000a:0000:00bc:2500:0d0b

Remembering more than a few such addresses is practically impossible, and writing them is not much fun either. Fortunately, there are two rules for reducing the size of written IPv6 addresses. The first rule is

*The leading zeroes in any 16-bit segment do not have to be written; if any 16-bit segment has fewer than four hexadecimal digits, it is assumed that the missing digits are leading zeroes*.

In the example address, the third, fourth, fifth, sixth, and eighth segments have leading zeroes. Using the first address compression rule, the address can be written as

3ffe:1944:100:a:0:bc:2500:d0b

Notice that only leading zeroes can be omitted; trailing zeroes cannot, because doing so would make the segment ambiguous. You would not be able to tell whether the missing zeroes belonged before or after the written digits.

Notice also that the fifth segment in the example address is all zeroes, and is written with a single zero. Many IPv6 addresses have long strings of zeroes in them. Take, for example, the following address:

ff02:0000:0000:0000:0000:0000:0000:0005

This address can be reduced as follows:

ff02:0:0:0:0:0:0:5

However, using the second rule can reduce this address even further:

*Any single, contiguous string of one or more 16-bit segments consisting of all zeroes can be represented with a double colon*.

Using this rule, the example address can be represented as the following:

ff02::5

The increased convenience in writing such an address is obvious. But notice that the rule says only a *single* contiguous string of all-zero segments can be represented with a double colon. Using the double colon more than once in an IPv6 address can create ambiguity. Take, for example, the following address:

2001:0d02:0000:0000:0014:0000:0000:0095

Either of the following reductions of the address is correct because they use a double colon only once:

2001:d02::14:0:0:95
2001:d02:0:0:14::95

But the following reduction is illegal because it uses the double colon twice:

2001:d02::14::95

It is illegal because the length of the two all-zero strings is ambiguous; it could represent any of the following IPv6 addresses:

2001:0d02:0000:0000:0014:0000:0000:0095
2001:0d02:0000:0000:0000:0014:0000:0095
2001:0d02:0000:0014:0000:0000:0000:0095

Unlike IPv4, in which the prefixthe network portion of the addresscan be identified by a dotted decimal or hexadecimal address mask or a bitcount, IPv6 prefixes are always identified by bitcount. That is, the address is followed by a forward slash and a decimal number indicating how many of the first bits of the address are the prefix bits. For example, the prefix of the following address is the first 64 bits:

3ffe:1944:100:a::bc:2500:d0b/64

When you are writing just an IPv6 prefix, you set all the host bits to 0 the same way you do with IPv4 addresses. For example

3ffe:1944:100:a::/64

An IPv6 address consisting of all zeroes can be written simply with a double colon. There are two cases where an all-zeroes address is used. The first is a default address, discussed in Chapter 12, "Default Routes and On-Demand Routing," in which the address is all zeroes and the prefix length is zero:

::/0

The second all-zeroes IPv6 address is an *unspecified* address, which is used in some Neighbor Discovery Protocol procedures described later in this chapter. An unspecified address is a filler, indicating the absence of a real IPv6 address. When writing an unspecified address, it is differentiated from a default address by its prefix length:

::/128

## IPv6 Address Types

The three types of IPv6 address follow:

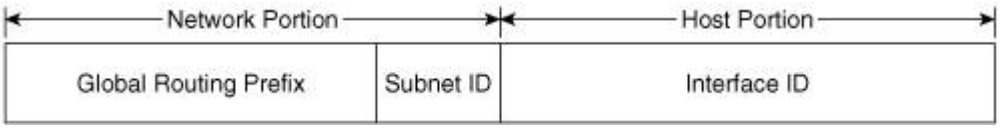
- Unicast
- Anycast
- Multicast

Unlike IPv4, there is no IPv6 broadcast address. There is, however, an "all nodes" multicast address, which serves essentially the same purpose as a broadcast address.

### Global Unicast Addresses

A unicast address is an address that identifies a single device. A global unicast address is a unicast address that is globally unique. The general format of the IPv6 unicast address is shown in Figure 2-1. This format, specified in RFC 3587, obsoletes and simplifies an earlier format that divided the IPv6 unicast address into Top Level Aggregator (TLA), Next-Level Aggregator (NLA), and other fields. However, you should be aware that this obsolescence is relatively recent and you are likely to encounter some books and documents that show the old IPv6 address format.

Figure 2-1. The IPv6 general unicast address format.



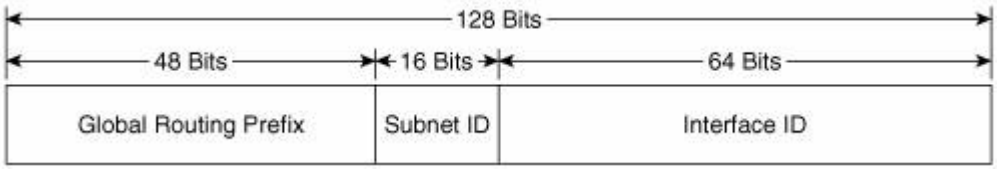

The host portion of the address is called the Interface ID. The reason for this name is that a host can have more than one IPv6 interface, and so the address more correctly identifies an interface on a host than a host itself. But that subtlety only goes so far: A single interface can have multiple IPv6 addresses, and can have an IPv4 address in addition, in which case the Interface ID is only one of that interface's several identifiers.

Perhaps the most striking difference between IPv4 addresses and IPv6 addresses, aside from their lengths, is the location of the Subnet Identifier as a part of the network portion of the address rather than the host portion. A legacy of the IPv4 address class architecture is that the subnet portion of an IPv4 address is taken from the host portion of the address. As a result, the host portion of the IPv4 address varies not only with its class, but also with the number of bits you use for subnet identification.

The immediate benefit of making the IPv6 Subnet ID field a part of the network portion of the address is that the Interface ID can be a consistent size for all IPv6 addresses, simplifying the parsing of the address. And making the Subnet ID a part of the network portion creates a clear separation of functions: The network portion provides the location of a device down to the specific data link and the host portion provides the identity of the device on the data link.

The Interface ID of the global IPv6 address is, with very few exceptions, 64 bits long. Also with very few exceptions, the Subnet ID field is 16 bits (Figure 2-2). A 16-bit Subnet ID field provides for 65,536 separate subnets; it seems that using a fixed Subnet ID size such as this, when in most cases the capacity will not be nearly fully used, is wasteful. But given the overall size of the IPv6 address space, and given the benefits of easy address assignment, design, management, and parsing that comes from using a fixed size, the waste is justified.

Figure 2-2. The standard field sizes of the global unicast IPv6 address.




The IANA and the Regional Internet Registries (RIRs)[2] assign IPv6 prefixesnormally /32 or /35 in lengthto the *Local Internet Registries* (LIRs). The LIRs, which are usually large Internet Service Providers, then allocate longer prefixes to their customers. In the majority of cases, the prefixes assigned by the LIRs are /48. There are, however, as mentioned in the previous paragraph, a few exceptions in which the LIR might assign a prefix of a different length:

[2] As of this writing there are five RIRs: Réseaux IP Européens (RIPE) serves Europe, the Middle East, and Central Asia; Latin American and Caribbean Internet Address Registry (LACNIC) serves Central and South America and the Caribbean; American Registry for Internet Numbers (ARIN) serves North America and parts of the Caribbean; AfriNIC serves Africa; and Asia Pacific Network Information Centre serves Asia and the Pacific Ocean nations.

- If the customer is very large, a prefix shorter than /48 might be assigned.
- If one and only one subnet is to be addressed, a /64 might be assigned.
- If one and only one device is to be addressed, a /128 might be assigned.

## Identifying IPv6 Address Types

The first few bits of the address specify the address type. For example, the first three bits of all global unicast addresses currently are 001. As a result, recognizing the hexadecimal representations of global unicast addresses is fairly easy: They all start with either 2 or 3, depending on the value of the fourth bit in the global routing prefix. So, for instance, currently allocated prefixes used by the 6Bone (the public IPv6 research network) begin with 3ffe, and IPv6 addresses currently allocated by the RIRs begin with 2001.

Binary 001 is expected to suffice for global unicast addresses for some time to come; a few other bit combinations are assigned to other defined address types, and the majority of leading bit combinations are reserved. Table 2-1 lists the currently allocated leading bit combinations, and the following subsections describe the other major IPv6 address types.

Table 2-1. High-order bits of IPv6 address types.

| Address Type | High-Order Bits (binary) | High-Order Bits (Hex) |
|---|---|---|
| Unspecified | 00...0 | ::/128 |
| Loopback | 00...1 | ::1/128 |
| Multicast | 11111111 | FF00::/8 |
| Link-Local Unicast | 1111111010 | FF80::/10 |
| Site-Local Unicast (Deprecated) | 1111111011 | FFC0::/10 |

| Address Type | High-Order Bits (binary) | High-Order Bits (Hex) |
|---|---|---|
| Global Unicast (Currently allocated) | 001 | 2xxx::/4 or 3xxx::/4 |
| Reserved (Future global unicast allocations) | Everything else | |

## Local Unicast Addresses

When we talk of global unicast addresses, we mean an address with global *scope*. That is, an address that is globally unique and can therefore be routed globally with no modification.

IPv6 also has a *link-local* unicast address, which is an address whose scope is confined to a single link. Its uniqueness is assured only on one link, and an identical address might exist on another link, so the address is not routable off its link. As you can see in Table 2-1, the first 10 bits of the link-local unicast address are always 1111111010 (FF80::/10).

As subsequent sections in this chapter demonstrate, link-local addresses have great utility for functions such as the Neighbor Discovery Protocol that communicates only on a single link. It also allows devices that are on links that do not have assigned global prefixes, or devices that do not yet know the global prefix assigned to the link, to create IPv6 addresses that allow them to communicate with other devices on the link. The section "Address Autoconfiguration" shows how link-local prefixes are used in this situation.

IPv6 originally defined a site-local unicast address in addition to the link-local address. A site-local address is unique only within a given site; devices in other sites can use the same address. Therefore a site-local address is routable only within the site to which it is assigned. Site-local IPv6 addresses are, then, functionally similar to private IPv4 addresses as defined in RFC 1918.

Advocates of site-local addresses cite several applications. One prominent application is for network operators that wish to use NAT, even with IPv6 addresses, to maintain independence of their address architecture from that of their service providers. Site-local addresses are also key to several proposed IPv6 multihoming mechanisms.

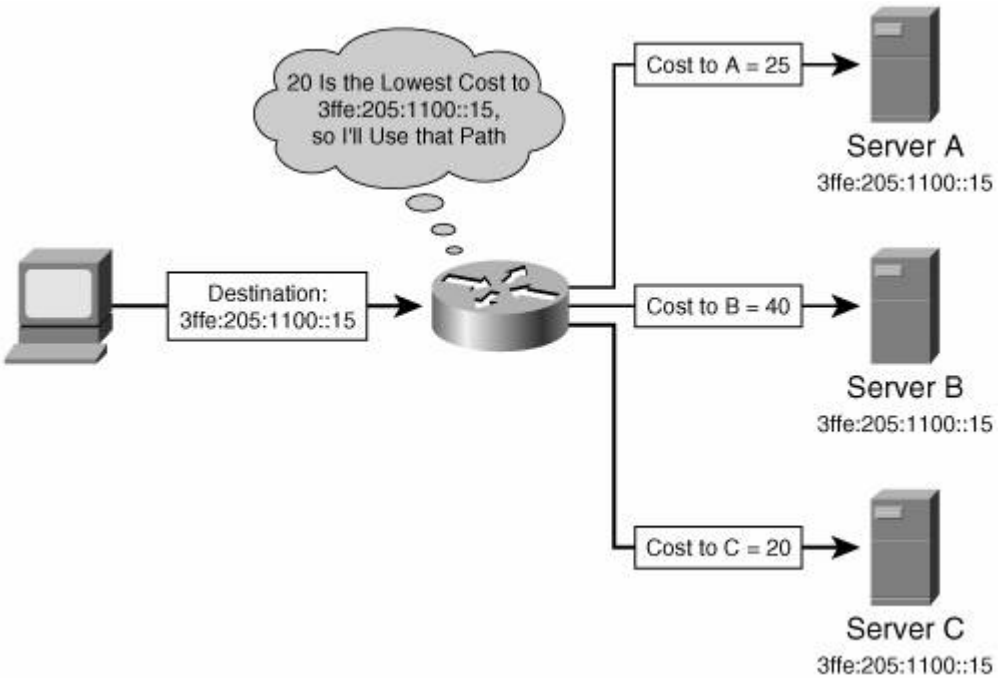
However, the IETF IPv6 Working Group determined that site-local unicast addresses introduced a number of difficulties. Not the least of the difficulties is the fact that the definition of a "site" is vague and can mean different things to different network administrators. Another problem is concern over, like RFC 1918 IPv4 addresses, the administrative difficulties introduced when such addresses are mistakenly "leaked" outside of their intended site boundaries. Other potential problems cited include increased complexity for applications and routers that must recognize and cope with site-local addresses. As a result of these concerns, and after some heated debate, the IPv6 Working Group deprecated site-local addresses in RFC 3879. An assurance has been given to those who see advantages in site-local addresses to introduce another scheme with similar "bigger scope than link but smaller scope than global" benefits, but as of this writing such a replacement scheme has yet to be seen.

The first 10 bits of site-local unicast addresses, as shown in Table 2-1, is 1111111011 (FFC0::/10).

## Anycast Addresses

An anycast address represents a service rather than a device, and the same address can reside on one or more devices providing the same service. In Figure 2-3, some service is offered by three servers, all advertising the service at the IPv6 address 3ffe:205:1100::15. The router, receiving advertisements for the address, does not know that it is being advertised by three different devices; instead, the router assumes that it has three routes to the same destination and chooses the lowest-cost route. In Figure 2-3 this is the route to server C with a cost of 20.

Figure 2-3. An anycast address represents a service that might appear on multiple devices.




The advantage of anycast addresses is that a router always routes to the "closest" or "lowest-cost" server.[3] So servers providing some commonly used service can be spread across a large network and traffic can be localized or scoped to the nearest server, making traffic patterns in the network more efficient. And if one server becomes unavailable, the router routes to the next nearest server. In Figure 2-3, for example, if server C becomes unavailable due to a network or server failure, the router chooses the path to server A as the next-lowest-cost route. From the router's viewpoint, it is just choosing the next-best route to the same destination.

[3] The methods by which a router chooses among multiple routes to the same destination is covered in Chapter 4, "Dynamic Routing Protocols.

Anycast addresses are defined by their service function only, not by format, and theoretically might be any IPv6 unicast address of any scope. However, there is a format for reserved anycast addresses, defined in RFC 2526. Anycast addresses have been used for some time in IPv4 networks, but are formalized in their definition in IPv6.

## Multicast Addresses

A multicast address identifies not one device but a set of devicesa *multicast group*. A packet being sent to a multicast group is originated by a single device; therefore a multicast packet normally has a unicast address as its source address and a multicast address as its destination address. A multicast address never appears in a packet as a source address.

The members of a multicast group might include only a single device, or even all devices in a network. In fact, IPv6 does not have a reserved broadcast address like IPv4, but it does have a reserved all-nodes multicast group, which is essentially the same thing: a multicast group to which all receiving devices belong.

Multicasting is essential to the basic operation of IPv6, particularly some of its plug-and-play features such as router discovery and address autoconfiguration. These functions are a part of

the Neighbor Discovery Protocol, discussed later in this chapter.

The format of the IPv6 multicast address is shown in Figure 2-4. The first eight bits of the address are always all ones, and the next four bits are designated as flags. Currently the first three of these bits are unused and always set to 0. The fourth bit indicates whether the address is a permanent, well-known address (0) or an administratively assigned transient address (1). The next four bits indicate the scope of the address as shown in Table 2-2. Table 2-3 shows several reserved, well-known IPv6 multicast addresses, all of which are link-local scope. Because a multicast group is always a set of individual nodes, there is no needor sensefor having a subnet field in the multicast address. So the last 112 bits are used as the Group-ID, identifying individual multicast groups. Current usage sets the first 80 bits to 0 and just uses the last 32 bits.

Figure 2-4. The IPv6 multicast address format.

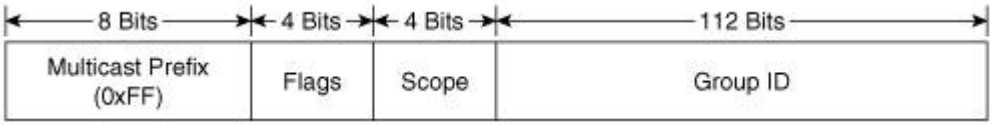



Table 2-2. Multicast address scopes.

| Scope Field Value | Scope |
|---|---|
| 0x0 | Reserved |
| 0x1 | Node-Local |
| 0x2 | Link-Local |
| 0x5 | Site-Local |
| 0x8 | Organization Local |
| 0xE | Global |
| 0xF | Reserved |

Table 2-3. Examples of well-known IPv6 multicast addresses.

| Address | Multicast Group |
|---|---|
| FF02::1 | All Nodes |
| FF02::2 | All Routers |
| FF02::5 | OSPFv3 Routers |

| Address | Multicast Group |
|---|---|
| FF02::6 | OSPFv3 Designated Routers |
| FF02::9 | RIPng Routers |
| FF02::A | EIGRP Routers |
| FF02::B | Mobile Agents |
| FF02::C | DHCP Servers/Relay Agents |
| FF02::D | All PIM Routers |

## Embedded IPv4 Addresses

There are several transition technologiesmeans of helping to transition a network from IPv4 to IPv6 or otherwise help IPv4 and IPv6 to coexistthat require an IPv4 address to be communicated within an IPv6 address. The individual technology specifies how the IPv4 address is to be embedded in the IPv6 address, and the implementation of the technology knows where among the 128 bits of the IPv6 address to find the 32 bits of the IPv4 address. But you will also find that many of these technologies have unique formats for their address representations that allow you to identify the embedded IPv4 address. Examples of IPv6 addresses with an embedded IPv4 address of 10.23.1.5 are

FE80::5EfE:10.23.1.5 (An ISATAP address)
::FFFF:10.23.1.5 and ::FFFF:0:10.23.1.5 (SIIT addresses)
FEC0:0:0:1::10.23.1.5 (TRT address)

In each of these examples, the IPv4 address is the last 32 bits of the IPv6 address and is represented in dotted decimal.

Other transition technologies using embedded IPv4 addresses do not use dotted decimal but encode the IPv4 address into hexadecimal. 6to4, for example, does this. 10.23.1.5 in hexadecimal is 0A17:0105. A 6to4 prefix with 10.23.1.5 embedded is then

PREV NEXT

# IPv6 Packet Header Format

The format of the IPv6 packet header is shown in Figure 2-5. There are some distinct similarities and some differencessome distinct, some subtlewith the IPv4 packet header shown in Figure 1.2 of the previous chapter.

Figure 2-5. The IPv6 packet header.

Version is, as with the IPv4 header, a four-bit field indicating the IP version. Here, of course, it is set to binary 0110 to indicate version 6.

*Traffic Class* is an eight-bit field that corresponds to the eight-bit IPv4 ToS field. But given the evolution of the ToS field over the years, both are now used for Differentiated Class of Service (DiffServ). So even though there is a correspondence of this field with the old ToS field, its name more accurately reflects the current usage of the values carried here.

*Flow Label* is a field unique to IPv6. The intention of this 20-bit field is to allow labeling of particular flows of traffic; that is, packets that are not just originated by the same source and going to the same destination, but that belong to the same applications at the source and destination. There are several advantages to differentiating flows, from providing a finer-grained differentiated class-of-service treatment to ensuring, when balancing traffic loads across multiple paths, that packets belonging to the same flow are always forwarded over the same path to prevent possible reordering of packets. Flows (or more accurately, *microflows*) typically are identified by a combination of source and destination address plus source and destination port.

But to identify the source and destination port, a router must look beyond the IP header and into the TCP or UDP (or other transport-layer protocol) header, adding to the complexity of the forwarding process and possibly affecting router performance. Finding the transport layer header in an IPv6 packet can be especially problematic because of extension headers, described in the next section. An IPv6 router must step through possibly many extension headers to find the transport-layer header.

By marking the Flow Label field appropriately when the packet is originated, routers can identify a flow by looking no further than the packet header. As of this writing, however, the complete

specification of how to use the flow label field is still being debated, and routers currently ignore the field. It nevertheless holds promise of allowing IPv6 to provide better Quality of Service (QoS) features than IPv4 for applications such as Voice over IP (VoIP).

*Payload Length* specifies the length of the payload, in bytes, that the packet is encapsulating. Recall from Chapter 1, "TCP/IP Review," that IPv4 headers, because of the Options and Padding fields, can vary in length. Therefore, to find the payload length in an IPv4 packet, the value of the Header Length field must be subtracted from the Total Length field. The IPv6 packet header, on the other hand, is always a fixed length of 40 bytes, and so the single Payload Length field is enough to find the beginning and end of the payload.

Notice also that whereas the IPv4 Total Length field is 16 bits, the IPv6 Payload Length field is 20 bits. The implication here is that because a much longer payload (1,048,575 bytes, versus 65,535 in IPv4) can be specified in this field, the IPv6 packet itself is theoretically capable of carrying a far larger payload.

*Next Header* specifies which header follows the IPv6 packet header. In this, it is very similar to the Protocol field in the IPv4 header and, in fact, is used for the same purpose when the next header is an upper-layer protocol header. Like that IPv4 field, this field is also eight bits. But in IPv6, the header following the packet header might not be an upper-layer protocol header, but an extension header (again, described in the next section). So the Next Header field is named to reflect this wider range of responsibility.

*Hop Limit* corresponds exactly, both in length (eight bits) and function, to the IPv4 Time to Live (TTL) field. As you read in Chapter 1, the original intention of the TTL field was that it would be decremented by the number of seconds a packet is queued in a router during forwarding, but that this function was never implemented. Instead, routers decrement the TTL by one no matter how long the packet is queued (and in modern networks it is highly unusual for a packet to be queued anywhere near as long as one second). Therefore, the TTL has always been a measure of the maximum router hops a packet can take on its way to a destination. If the TTL decrements to 0, the packet is discarded. Hop Limit is used for exactly the same, but is named more appropriately for this function.

*Source* and *Destination Address* correspond to the IPv4 Source and Destination fields, except of course these fields are 128 bits each to accommodate IPv6 addresses.

Noticeably missing from the IPv6 header is a Checksum field like that of the IPv4 header. Given the overall increase in reliability of modern transport mediawireless perhaps being a notable exceptionalong with the fact that upper-layer protocols usually carry their own error-checking and recovery mechanisms, checksumming of the IPv6 header itself adds little value, and is therefore eliminated.

PREV NEXT

# Extension Headers

Comparing the IPv6 header in Figure 2-5 with the IPv4 header in Figure 1.2, you can see that although the Source and Destination Address fields are each four times as long in the IPv6 header, the IPv6 header itself is not that much larger than an IPv4 header: 40 bytes for IPv6 versus a minimum of 20 bytes for IPv4. If extensive use is made of the IPv4 Options field, although unusual, the IPv4 header can actually be larger than the IPv6 header.

Also notice that in addition to the Options field, other fields that are not always used, such as those associated with fragmentation, are eliminated from the IPv6 header. So given its fixed length and exclusion of all fields that do not carry information necessary for the forwarding of every packet, the IPv6 header is both compact and efficient.

But what if you do want to use one of those optional IP features, such as fragmentation or source routing or authentication? When an optional function is used in IPv6, an *extension header* appropriate for the function is added after the packet header. If, for example, source routing, fragmentation, and authentication options are to be used, three extension headers formatted to carry the information needed for each of those functions are added as shown in Figure 2-6. Because of these headers, efficiency is added to IPv6 packets in two ways:

- The packet carries only the information required by that individual packet. No unused fields are carried.
- New optional functions can be added to the IPv6 packet by defining new extension headers.

Figure 2-6. Extension headers allow IPv6 packets to carry all the information required for that packet, but only the information required for that packet.

[View full size image]

Each extension header, like the IPv6 header, has a Next Header field. So each header tells which header follows it. Table 2-4 shows the currently defined extension headers and their next header values. So, for example, in Figure 2-7, the Next Header value in the IPv6 header indicates that the next header is a Routing extension header (43), that header's Next Header field indicates that the next header is a Fragmentation extension header (44), and so on. The last extension header, AH, indicates that the next header is a TCP header (Protocol Number 6).

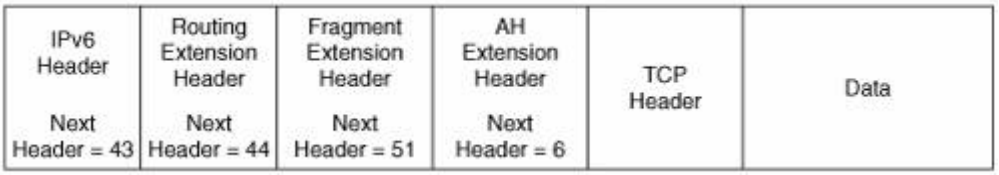
Table 2-4. Next Header values.

| Header | Next Header Value |
|---|---|
| Hop-By-Hop Options | 0 |
| Routing | 43 |

| Header | Next Header Value |
|---|---|
| Fragment | 44 |
| Encapsulating Security Payload (ESP) | 50 |
| Authentication Header (AH) | 51 |
| Destination Options | 60 |
| TCP/IP Protocols | Protocol number value defined for that protocol (such as TCP = 6, UDP = 17, OSPF = 89, and so on) |
| No Next Header | 59 |

Figure 2-7. The Next Header field in the IPv6 header and each extension header specifies which header follows it.

[View full size image]




The format of each of the extension headers is described in RFC 1883. But briefly, the function of each extension header is as follows:

- Hop-By-Hop Optionscarries information that must be examined by every node along the forwarding path, such as Router Alert and Jumbo Payload options.
- Routingprovides source routing functionality by listing nodes that the packet must pass through on the way to its destination.
- Fragmentis used when a packet is fragmented, to provide the information necessary for the receiving node to reassemble the packet. A significant difference between IPv4 and IPv6 is that only originating nodes can fragment packets; IPv6 routers do not fragment the packets. So originating nodes must either use Path MTU Discovery (PMD) to find the lowest MTU along a path to the destination, or never produce packets larger than 1280 bytes. PMD is described in the next section. IPv6 specifies that all links on which it runs must be able to support packet sizes of at least 1280 bytes so that originators can use the minimum-size option rather than PMD if they so choose.
- Encapsulating Security Payload (ESP) is used when the payload is encrypted.
- Authentication Header (AH) is used when the packet must be authenticated between the source and destination.
- Destination Options carries information to be examined only by the destination node or possibly by nodes listed in the Routing header.

RFC 1883 also specifies the order in which extension headers, if they are used, should appear. The only hard-and-fast rule here is that if the Hop-By-Hop Options header is used, it must directly follow the IPv6 header so that it can be easily found by the transit nodes that must examine it. The recommended extension header order is as follows:

1. IPv6 Header
2. Hop-By-Hop Options
3. Destination Options (only if intermediate routers specified in the Routing header must examine this header)
4. Routing
5. Fragment
6. Authentication
7. Encapsulating Security Payload
8. Destination Options (if only the final destination must examine this header)
9. Upper-Layer Header

PREV NEXT

# ICMPv6

IPv6 requires a control protocol for exchanging and processing error and informational messages, just as IPv4 does. And like IPv4, it uses ICMP to do this. But the ICMP used by IPv6 is not the same ICMP as used by IPv4. Although ICMP for IPv4 has a Protocol Number of 1, ICMPv6 for IPv6 has a Next Header value of 58.

ICMPv6 is specified in RFC 2463. Many of the functions defined in this RFC are the same ones defined for ICMP for IPv4; but there are many ICMP messages, such as Source Quench and Timestamp, that have no equivalent in ICMPv6.

Comparing the ICMPv6 header shown in Figure 2-8 to the ICMP header shown in Figure 1.28, you can see that they are identical. And like ICMP, ICMPv6 uses a combination of type and code values to identify general types and then subtypes under them. The values defined in RFC 1885 are listed in Table 2-5.

Figure 2-8. The ICMPv6 header format.

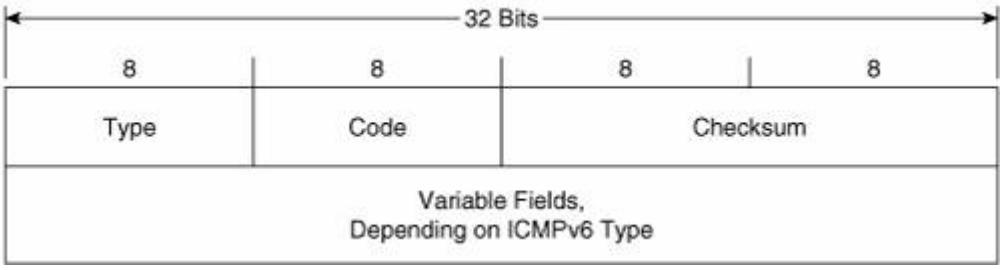



Table 2-5. ICMPv6 Message Type and Code fields.

| Type | Code | Message |
|---|---|---|
| 1 | | DESTINATION UNREACHABLE |
| | 0 | No route to destination |
| | 1 | Communication with destination administratively prohibited |
| | 2 | Not a neighbor |
| | 3 | Address unreachable |
| | 4 | Port unreachable |
| 2 | 0 | PACKET TOO BIG |
| 3 | | TIME EXCEEDED |
| | 0 | Hop limit exceeded in transit |
| | 1 | Fragment reassembly time exceeded |
| 4 | | PARAMETER PROBLEM |

| Type | Code | Message |
|---|---|---|
| | 0 | Erroneous header field encountered |
| | 1 | Unrecognized Next Header type encountered |
| | 2 | Unrecognized IPv6 option encountered |
| 128 | 0 | ECHO REQUEST |
| 129 | 0 | ECHO REPLY |
| 130 | 0 | GROUP MEMBERSHIP QUERY |
| 131 | 0 | GROUP MEMBERSHIP REPORT |
| 132 | 0 | GROUP MEMBERSHIP REDUCTION |

In addition to the basic error and informational functions of ICMPv6, there are mechanisms that use the ICMPv6 messages. For example, the Path MTU Discovery mechanism mentioned in the previous section sends packets of increasing size to a destination. When the smallest MTU of the links on the path to the destination is exceeded by a given packet size, the packet is dropped and a Packet Too Big message is sent to the source address; the source then knows the smallest MTU on the path. And, as with IPv4, Echo and Echo Reply messages are used by the Ping function.

But in addition to basic error and information messages, there is a separate set of ICMPv6 messages defined that are used by an essential IPv6 protocol: the Neighbor Discovery Protocol,

PREV NEXT

# Neighbor Discovery Protocol

The most distinct characteristics of IPv6 after its increased address space are its plug-and-play features. *Neighbor Discovery Protocol* (NDP) is the enabler of these plug-and-play features, using the following functions:

- Router Discovery A node can discover, when it is connected to an IPv6 link, the local routers without the aid of Dynamic Host Configuration Protocol (DHCP).
- Prefix Discovery A node can discover, when it is connected to an IPv6 link, the prefix or prefixes assigned to that link.
- Parameter Discovery A node can discover parameters such as the link MTU and hop limits for its connected link.
- Address Autoconfiguration A node can determine its full address, again without the aid of DHCP.
- Address Resolution A node can discover the link-layer addresses of other nodes on the link without the use of Address Resolution Protocol (ARP).
- Next-Hop Determination A node on a link can determine the link-layer next hop for a destination, either as a local destination or a router to the destination.
- Neighbor Unreachability Detection A node can determine when a neighbor on a link, either another host or a router, is no longer reachable.
- Duplicate Address Detection A node can determine if an address it wants to use is already being used by another node on the link.
- Redirect A router can notify a host of a better next-hop than itself to an off-link destination. The redirect function is a part of basic ICMP functionality in IPv4, but is redefined as part of NDP in IPv6.

NDP messages should always be link-local in scope, and therefore the packets encapsulating them always use either link-local IPv6 addresses or multicast addresses with a link-local scope. To add a further layer of security, the Hop Limit of the IPv6 packet carrying all NDP messages is 255. If one of these packets is received with a Hop Limit less than that value, it means the packet has passed through at least one router, and the packet is dropped. This prevents NDP from being attacked or spoofed from a source not attached to the local link.

## NDP Messages

NDP is defined in RFC 2461. It uses ICMPv6 to exchange the messages necessary for its functions; specifically, five new ICMPv6 messages are specified in RFC 2461:

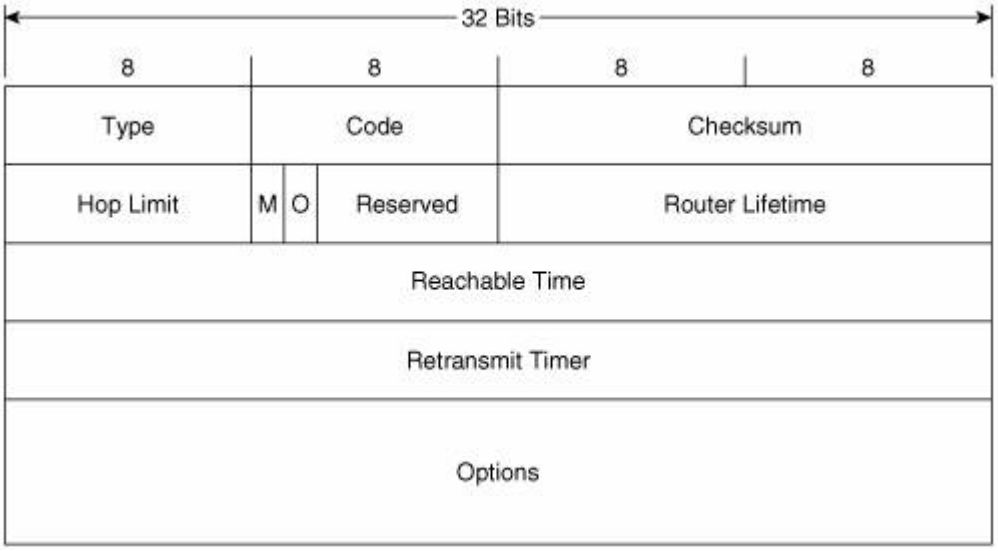
- Router Advertisement (RA) messages are originated by routers to advertise their presence and link-specific parameters such as link prefixes, link MTU, and hop limits. These messages are sent periodically, and also in response to Router Solicitation messages.
- Router Solicitation (RS) messages are originated by hosts to request that a router send an RA.
- Neighbor Solicitation (NS) messages are originated by nodes to request another node's

link layer address and also for functions such as duplicate address detection and neighbor unreachability detection.

- Neighbor Advertisement (NA) messages are sent in response to NS messages. If a node changes its link-layer address, it can send an unsolicited NA to advertise the new address.
- Redirect messages are used the same way that redirects are used in ICMP for IPv4; they have merely been moved from being a part of the base ICMPv6 protocol to being a part of NDP.

Figure 2-9 shows the format of the Router Advertisement message. Its ICMPv6 type is 134 and the code is 0. The source address of the IPv6 packet encapsulating the RA is always the IPv6 link-local address of the interface from which the packet originates. The destination address is either the all-nodes multicast address (FF02::1) if the RA is a periodic transmit, or the link-local address of the soliciting node if the RA is sent in response to a Router Solicitation.

Figure 2-9. The Router Advertisement message format.




Hop Limit indicates the value of the Hop Limit field that nodes attached to the link should give to any packets they originate on the link. If no Hop Limit is specified by this router, the field is set to all zeroes.

M is the Managed Address Configuration flag. If this bit is set, the originating router is telling hosts on the link to use stateful address autoconfiguration via DHCPv6. If the flag is cleared, hosts on the link should use stateless address autoconfiguration. Address autoconfiguration is described later in this chapter.

O is the Other Stateful Configuration flag. When set, the originating router is telling hosts on the link to use DHCPv6 for the acquisition of other link information. The M and O flags can be used together. For example, by clearing the M flag but setting the O flag, the router is telling hosts to use stateless address autoconfiguration but then consult a DHCPv6 server for other configuration parameters.

Router Lifetime is set to a value other than 0 only if the originating router is a default router. In that case, this field specifies the lifetime of the default router in seconds, up to a maximum value of 18.2 hours.

Reachable Time is used by the Neighbor Unreachability Detection function of NDP. It specifies the time, in milliseconds, that a node should assume a neighbor is reachable after the node has

confirmed reachability of the neighbor.

Retransmit Timer is used by the Address Resolution and Neighbor Unreachability Detection functions of NDP. It specifies the minimum time, in milliseconds, between retransmitted Neighbor Solicitation messages.

Possible options that can be carried in the Options field of the RA include the following:

- The link-layer address of the interface from which the RA is originated.
- An MTU specification for the link.
- One or more prefixes assigned to the link. This information is essential to stateless address autoconfiguration, telling hosts on the link what the link prefixes are.

Figure 2-10 shows the format of the Router Solicitation message. Its ICMPv6 type is 133 and the code is 0. The source address of the IPv6 packet encapsulating the RS is either the IPv6 address assigned to the originating interface or, if no address has been assigned (as would be the case if the originating host is beginning address autoconfiguration), an unspecified address of :: (all zeroes). The destination address is the all-routers multicast address (FF02::2).

Figure 2-10. The Router Solicitation message format.

The Options field can contain the link-layer address of the originating interface, if it is known. However, the source link-layer address must not be included if the source address of the encapsulation packet is unspecified, such as when the originator is soliciting a router during address autoconfiguration.

Figure 2-11 shows the format of the Neighbor Solicitation message. Its ICMPv6 type is 135 and the code is 0. The source address of the IPv6 packet encapsulating the NS is either the IPv6 address assigned to the originating interface or, if the NS is sent for Duplicate Address Detection, the unspecified address of :: (all zeroes). The destination address is either a solicited-node multicast address corresponding to the target address, or the target address.

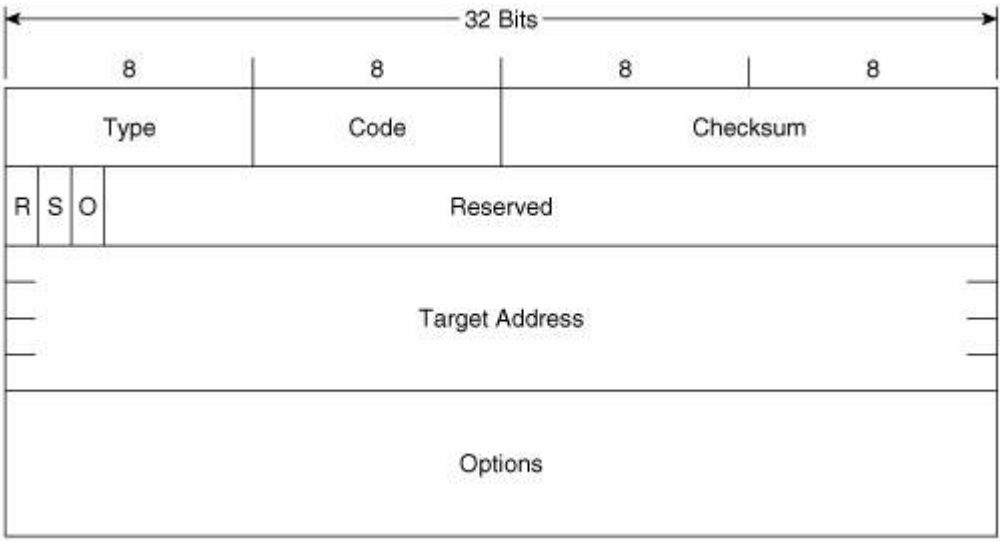
Figure 2-11. The Neighbor Solicitation message format.

Target Address is the IPv6 address of the target of the solicitation. The target address is never a multicast address.

The Options field of the NS can contain the link-layer address of the originating interface.

Figure 2-12 shows the format of the Neighbor Advertisement message. Its ICMPv6 type is 136 and the code is 0. The source address of the IPv6 packet encapsulating the NS is always the IPv6 address assigned (or autoconfigured) to the originating interface. The destination address is either the source address of the packet containing the NS to which the NA is sent in response, or the all-nodes multicast address (FF02::1).

Figure 2-12. The Neighbor Advertisement message format.




R is the *Router* flag. When set, it indicates that the originator is a router. This bit is used during Neighbor Reachability Detection to detect a router that has changed to a host.

S is the *Solicited* flag. This bit is set when the NA is sent in response to an NS.

O is the *Override* flag. When set, it indicates that the information in the NA should override any

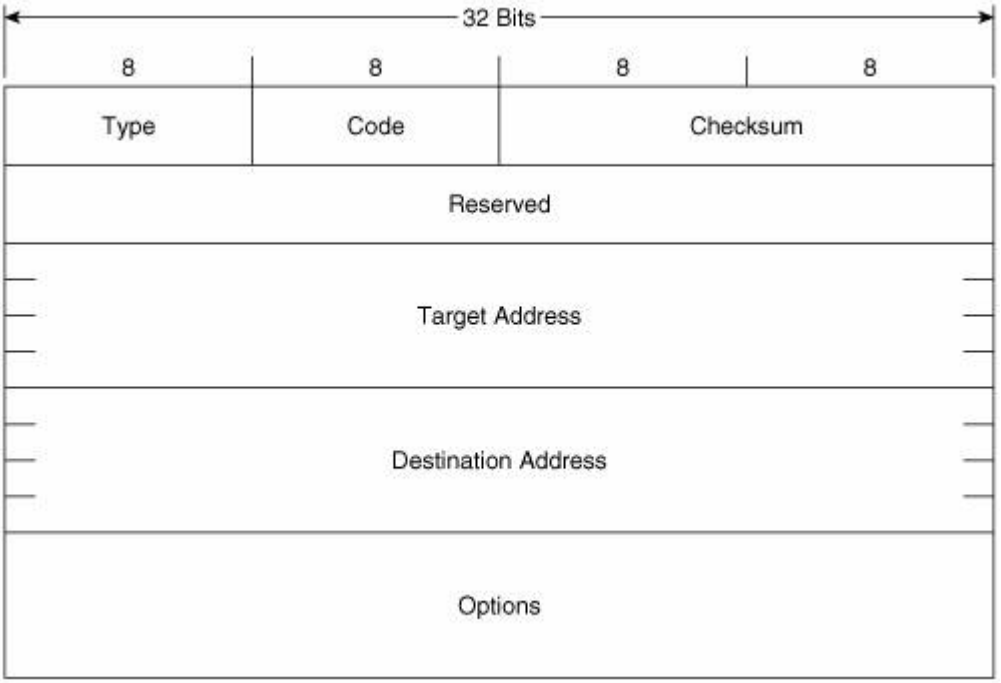
existing neighbor cache entry and update the cached link-layer address. When the O bit is cleared the NA will not override an existing neighbor cache entry.

Target Address is, when the NA is sent in response to a NS, the address in the Target Address field of the NS. If the NA is unsolicited (that is, sent to advertise a change of the originator's link-layer address), the Target Address is the originator's address.

The Options field of the NA can contain the target link-layer addressthat is, the link-layer address of the NA's originator.

Figure 2-13 shows the format of the Redirect message. Its ICMPv6 type is 137 and the code is 0. The source address of the IPv6 packet encapsulating the Redirect is always the link-local IPv6 address of the interface from which the message is originated. The destination address is always the source address of the packet that triggered the redirect.

Figure 2-13. The Redirect message format.




Target Address is the address of the better first-hopusually the link-local address of another router on the link.

Destination Address is the IPv6 address of the destination that is redirected to the target address.

The Options field of the Redirect message can contain the link-layer address of the target, and as much of the header of the packet that triggered the redirect, without making the redirect packet exceed 1280 bytes.

The Options field of all of these five messages, when it contains any information, consists of one or more Type/Length/Value (TLV) triplets. Each TLV consists, as shown in Figure 2-14, of an 8-bit Type field specifying the type of information carried in the value field, an 8-bit Length field specifying the length in units of 8 octets of the value field, and the variable length Value field.

Figure 2-14. The format of the TLVs used in the Options fields of the

NDP messages.

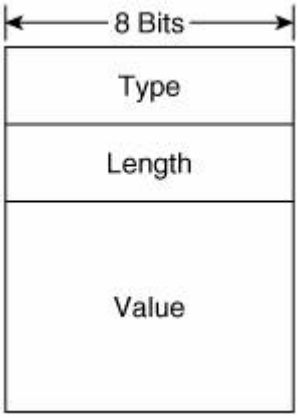



Table 2-6 shows the possible values and their associated type numbers. The format of the individual value fields is not provided in this chapter; consult RFC 2461 for the details on the value fields.

Table 2-6. Value fields and their types.

| Type | Value |
|---|---|
| 1 | Source Link-Layer Address |
| 2 | Target Link-Layer Address |
| 3 | Prefix Information |
| 4 | Redirected Header |
| 5 | MTU |

## Router Discovery

A router makes its presence known, along with any parameters it has been configured to advertise, by periodically sending RAs on its attached links. Presumably the links on which the RA do the most good are broadcast links such as Ethernet, where hosts can receive the RAs and thus learn necessary information about the link.

RFC 2461 specifies that the period between transmissions of RAs should be between 4 and 1800 seconds, with a default of 600 seconds. It also specifies a minimum period between advertisements of RAs with a default of 200 seconds. The advertisements are jittered between the maximum and minimum values to prevent synchronization on a link.

These unsolicited RAs are sent with their source address set to the link-local IPv6 address of the router's interface. The destination address is the all-nodes multicast address (FF02::1).

Cisco routers automatically send RAs on Ethernet and FDDI interfaces whenever IPv6 is enabled

on the router with the command ipv6 unicast-routing. The default interval is 200 seconds, and can be changed with the command ipv6 nd ra-interval. The Router Lifetime of the transmitted RAs is 1800 seconds by default, and can be changed with the command ipv6 nd ra-lifetime. If you do not want a router to be a default router on a link, you can use this command to set the Router Lifetime value to 0. The default Reachable Time of the RAs is 0 (which means unspecified), and can be changed with the command ipv6 nd reachable-time. The Retransmit Timer field is set to a default of 0 ms (unspecified) and can be changed with the command ipv6 nd ns-interval. The M and O flags can be set with the commands ipv6 nd managed-config-flag and ipv6 nd other-config-flag, respectively. If you do not want an interface to transmit RAs at all, you can disable them with the command ipv6 nd suppress-ra.

By default, Cisco routers include in the RAs all IPv6 prefixes configured on the originating interface. You can control the prefixes advertised, and parameters associated with those prefixes, with the command ipv6 nd prefix.

Of course, 200 seconds is a long time for a host that has just attached to an interface to wait for an RA so that it can find the routers and learn the link parameters. So when a host first becomes active on a link, it can send an RS to solicit the immediate transmission of an RA. The source of the RS can either be the unspecified address (::) or the host's link-local IPv6 address. The destination is always the all-routers multicast (FF02::2).

When a router receives an RS, it sends (after a delay of .5 seconds) an RA in response. If the source address of the RS that triggered the RA is a host's link-local address, the RA is unicast to the host using its link-local address. If the source address of the RS was unspecified, the solicited RA is multicast to the all-nodes address.

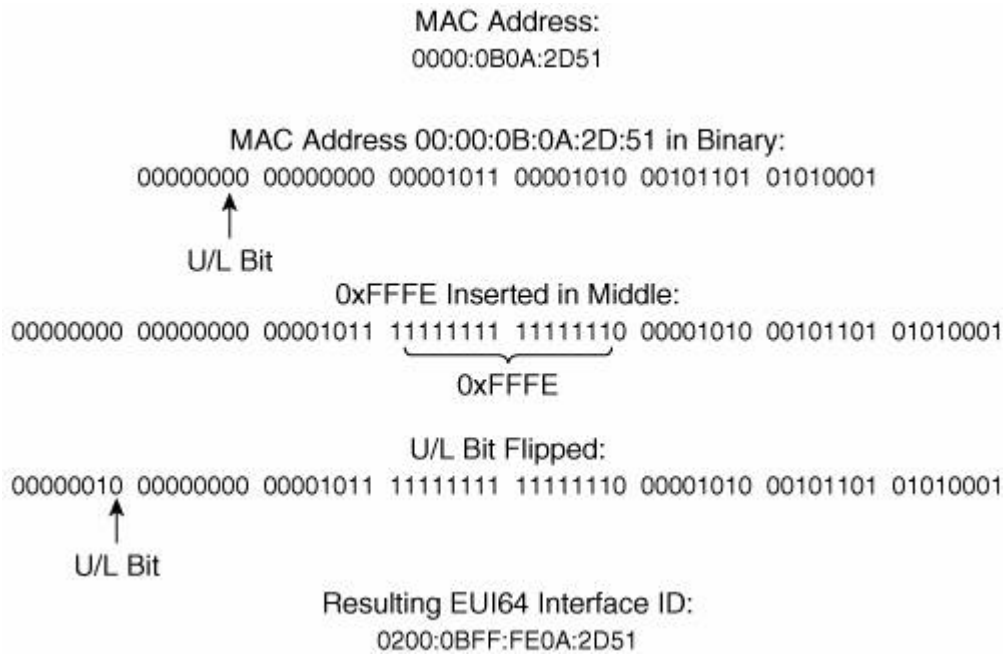
When a host receives an RA, it adds the router to its default router list (unless the RA indicates by a Router Lifetime value of 0 that it cannot be used as a default). If there is more than one router on the default router list, how the host selects a default router is implementation-specific. It could either rotate through the list, or select and keep a single router as default. In either instance, the Redirect function is essential for updating the host when a different default than the one it selected should be used.

## Address Autoconfiguration

When an IPv6 host first becomes active on a link, it can self-configure its own interface address. The first step in this process is the determination of the 64-bit Interface ID portion of the address. On broadcast interfaces (where hosts are most likely to appear), a mechanism called *MAC-to-EUI64 conversion* is used. Quite simply, this mechanism takes the 48-bit Media Access Control (MAC) address of the interfacewhich can normally be assumed to be globally uniqueand converts it into a 64-bit Interface ID by inserting a reserved 16-bit value of 0xFFFE into the middle of the MAC address and "flipping" the Universal/Local (U/L) bit of the MAC address to 1 (Universal).

Figure 2-15 illustrates this process. A MAC address, 0000:0B0A:2D51, is to be converted. The first step, shown in binary for easier understanding of what is happening, is to "split" the MAC address in the middle and insert 0xFFFE between the two 24-bit halves. The address is now 64 bits long. Next, the U/L bit of the original MAC addresswhich is always the 7th bitis flipped from 0 to 1. The resulting address, 0200:0BFF:FE0A:2D51, is now a valid 64-bit Interface ID.

Figure 2-15. MAC-to-EUI64 conversion is used to create a 64-bit Interface ID from an interface's 48-bit MAC address.




Of course, the Interface ID is only half of the IPv6 address; a 64-bit prefix is also required. Recall from Table 2-1 that the link-local prefix is a reserved, well-known value of 0xFF80::/10. Using this as a full 64-bit prefix (0xFF80::/64), it can be added onto the derived Interface ID, and the host now has a complete IPv6 address that can be used for communication with other devices on the same link. For example, combining the link-local prefix with the Interface ID derived in Figure 2-15 gives a link-local address of FF80::0200:0BFF:FE0A:2D51. 2-The following shows an example of a link-local address, in this case from an Ethernet interface "en1" on a Macintosh OS X host. Using the link-local prefix FF80::/10 and a MAC-to-EUI64 conversion, an IPv6 interface derives its link-local address with no help from any other device:

```
[Jeff-Doyles-Computer:~] jdoyle% ifconfig en1
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1300
        inet6 fe80::211:24ff:fe23:334e prefixlen 64 scopeid 0x5
        inet 10.10.24.13 netmask 0xffffff00 broadcast 10.10.24.255
        ether 00:11:24:23:33:4e
        media: autoselect status: active
        supported media: autoselect
[Jeff-Doyles-Computer:~] jdoyle%
```

If the host only needs to communicate with devices on the link, autoconfiguring its link-local address is sufficient. But if it needs to communicate with devices off-link, it needs an address with a wider scopenormally a global IPv6 address. There are two ways it can acquire this address: *stateful* or *stateless* address autoconfiguration.

If a host uses stateful address autoconfiguration, it consults a DHCPv6 server for the necessary address information. It might either be preconfigured to find a DHCPv6 server, or a received RA might have its M flag set telling it to use DHCPv6. DHCPv6, described in RFC 3315, is not much different in its end results than DHCP for IPv4.

Much more interesting is stateless autoconfiguration. With this very simple process, the host acquires one or more link prefixes from the RAs it receives. It then adds the prefix to its previously determined Interface ID, and it now has a globally unique IPv6 address. For example, if the host from Figure 2-15 received an RA advertising a prefix of 3FFE: 1104:404:1::/64, it would add that prefix to its Interface ID for a global address of 3FFE:1104: 404:1:0200:0BFF:FE0A:2D51.

## Duplicate Address Detection

Although the use of MAC addresses to derive an Interface ID almost always guarantees a unique address of any scope, it is wise to ensure that the address is unique. So whenever a device acquires a unicast address, it must perform Duplicate Address Detection before using the address. It does not matter whether the address was acquired via stateful or stateless configuration, or if the address was statically configured. The only exception to the rule is an anycast address, because anycast addresses by definition can appear on more than one device. There is also an assumption that if a Duplicate Address Detection has been performed on a link-local address that has an Interface ID that was derived from MAC-to-EUI64 conversion, and if the address passes, other addresses using the same Interface ID will also be unique, and so the Duplicate Address Detection does not need to be repeated.

A node that has acquired a new address classifies the address as tentative. The address cannot be used until the Duplicate Address Detection operation has been completed with verification that no other node on the link uses that address. The node sends an NS with the Target Address field set to the address to be verified. The source address of the NS is the unspecified address, and the destination of the NS is a *solicited-node multicast* address.

A solicited-node multicast address is formed by prepending the prefix FF02:0:0:0:0:1:FF00::/104 to the last 24 bits of the target address. For example, given the Interface ID derived in Figure 2-15, the solicited-node multicast address is FF02::1:FF0A:2D51. The reason for this is that if a node has autoconfigured more than one interface address, the last 24 bits of all of its addresses should be the same. So the one NS with a solicited-node multicast address should match all of its interface addresses. More important, using a solicited-node multicast address ensures that if two nodes attempt to do a Duplicate Address Detection on the same address simultaneously, they will detect each other.

If a node receives an NS and the target address matches one of its assigned addresses, it sends an NA with the Target Address and the destination address set to the tentative address. The node that had originated the NS, on receipt of the NA, knows that the tentative address is duplicate and cannot be used.

## Neighbor Address Resolution

You know from Chapter 1 that when an IPv4 node wants to communicate with another IPv4 node on a local link, it must first discover the destination's link-layer (or data link) address. This address is then used as the destination address in the frame that encapsulates the IP packets to that node. For example, a node might want to send a packet to examplehost.com. A DNS query returns the address 3FFE:521:2400:15:211:24FF:FE23:334E. The sending node must now discover the link-layer address to use as a destination address of the frame for the local link. As the previous chapter discussed, IPv4 uses ARP for this discovery. IPv6, however, uses NDP.

When the node examines the prefix of the IPv6 address returned by DNS, it either concludes that the destination is a neighbor on the local link or that it is off-link and therefore reachable through the default router. If the latter is the case, the node should already know the link-layer address of the default router from the RAs. But if the destination is on the local link, the node first looks in its *neighbor cache* to see if the address is known. The neighbor cache in IPv6 is very similar to the ARP cache in IPv4; it records known network-layer addresses and the link-layer addresses associated with them. The following shows a neighbor cache from a Microsoft Windows XP host. The neighbor cache stores known IPv6 addresses and their associated link-layer addresses:

```
C:\Documents and Settings\Jeff Doyle>ipv6 nc
5: fe80::202:2dff:fe25:5e4c 00-02-2d-25-5e-4c permanent
4: fe80::260:83ff:fe7b:2df3 00-60-83-7b-2d-f3 stale (router)
4: fe80::210:a4ff:fea0:bc97 00-10-a4-a0-bc-97 permanent
4: 3ffe:3700:1100:1:210:a4ff:fea0:bc97 00-10-a4-a0-bc-97 permanent
4: 3ffe:3700:1100:1:d9e6:b9d:14c6:45ee 00-10-a4-a0-bc-97 permanent
```

```
  4: 2001:468:1100:1:210:a4ff:fea0:bc97 00-10-a4-a0-bc-97 permanent
  4: 2001:468:1100:1:d9e6:b9d:14c6:45ee 00-10-a4-a0-bc-97 permanent
  3: 2002:c058:6301::c058:6301 192.88.99.1       permanent
  3: 2002:836b:213c::836b:213c 131.107.33.60     permanent
  3: 2002:4172:a85b::4172:a85b 127.0.0.1         permanent
  3: 2002:836b:213c:1:e0:8f08:f020:6 131.107.33.60     permanent
  3:  2001:708:0:1::624                   incomplete
  2:    ::65.114.168.91 127.0.0.1         permanent
  2: fe80::5efe:65.114.168.91 127.0.0.1         permanent
  2: fe80::5efe:169.254.113.126 127.0.0.1         permanent
  1:            fe80::1                   permanent
  1:                ::1                   permanent
```

If the address is not in the neighbor cache, it is entered but tagged Incomplete, indicating that address resolution is in progress. The node then sends an NS to the solicited-node multicast address associated with the target node. The NS should include the Source Link-Layer option (type 1), so that the solicited node would have the link-layer address of the soliciting node, and therefore would know where to send the responding NA. If a value other than 0 is included in the RAs, multiple NSs can be sent at that specified interval. If the Retransmit Timer value in the RAs is unspecified (0), the NS is retransmitted every 1000 ms until an NA is received. If no NA is received from the solicited node after three NS transmissions, the neighbor address resolution has failed and an ICMP message of type 1/code 3 (Destination Unreachable/Address Unreachable) is returned for each packet queued for transmission to the now unknown destination.

If the solicited node exists and the NS is valid, it responds with an NA. The Target Address field of the NA is set to the value of the Target Address field of the NS that triggered it. The soliciting node, upon receipt of the NA, can add the target node's link-layer address to the neighbor cache entry and change the entry from Incomplete to Reachable.

The neighbor cache of a Cisco router can be observed with the command `show ipv6 neighbors`, as shown in Example 2-1.

### Example 2-1. The neighbor cache of a Cisco router can be displayed with the command *show ipv6 neighbors*.

```
Confucius# show ipv6 neighbors

IPv6 Address                            Age  Link-layer Addr  State  Interface
2001:201:1502:1:210:a4ff:fea0:bc97        0  0010.a4a0.bc97   REACH  Ethernet0
fe80::210:a4ff:fea0:bc97                  0  0010.a4a0.bc97   REACH  Ethernet0
fe80::260:83ff:fe4c:5df2                  0  0060.834c.5df2   REACH  Ethernet0
3ffe:1300:a47:20:d9e6:b9d:14c6:45ee       0  0002.2d25.5e4c   REACH  Ethernet1
```

## Privacy Addresses

The stateless address autoconfiguration has raised a security concern for some: Even if a device moves from subnet to subnet or even major network to major network, its Interface ID always remains the same; and if the Interface ID remains the same, it can be tracked. At the least, this becomes a privacy issue. For example, suppose you are using IPv6 to connect to your company network. Recording and analyzing packets coming into some part of the network can identify you by your unchanging Interface ID. And by further analyzing the different prefixes prepended to that Interface ID, your employer can infer where you are at all times: at work, at home, traveling, or whatever. More insidious uses can also be made of such tracking, keeping record of your location and activities for everything from marketing to criminal exploitation.

RFC 3041 addresses this security concern by defining IPv6 *privacy addresses*. A privacy address

is one in which the Interface ID is generated by an algorithm using a pseudo-random number. What is significant about it, and makes it reasonably private, is that the Interface ID changes approximately once a day (or on some configurable period) and also whenever the node acquires a new IPv6 prefix.

Of course, a constantly changing address is not practical for reachability. Nodes that want to communicate with you, and hence DNS servers, must know you by only one or a few static addresses. So the standard statelessly configured IPv6 address remains your *public* address. Anyone wanting to send packets to you uses this address as the destination. But when you send packets back, you use the *private* address. This is a bit like having Caller ID in your home but blocking your number from appearing on anyone else's Caller ID. You can see who is calling you, but others cannot see your number when you call them.

The following shows the addresses assigned to a Microsoft Windows XP machine. There are two public IPv6 addresses assigned to the interface, and you can see that although the prefixes are different, the MAC-to-EUI64generated Interface IDs are the same. You can easily identify the public Interface IDs by the 0xFFFE inserted in the middle. But for both of these public addresses there is also a private address (which Windows labels as "anonymous"). These private addresses are created by prepending the RA-discovered IPv6 prefix onto the randomly generated Interface ID. Public and private (called "anonymous" here) are used together to create anonymity for the host but at the same time maintain reachability:

```
C:\Documents and Settings\Jeff Doyle>ipv6 if 4
Interface 4: Ethernet: Local Area Connection 2
  uses Neighbor Discovery
  uses Router Discovery
  link-layer address: 00-10-a4-a0-bc-97
    preferred global 2001:484:1200:1:d9e6:b9d:14c6:45ee,
     life 6d21h14m26s/21h12m4s (anonymous)
    preferred global 2001:468:1200:1:210:a4ff:fea0:bc97,
     life 29d23h59m25s/6d23h59m25s (public)
    preferred global 3ffe:3705:1200:1:d9e6:b9d:14c6:45ee,
     life 6d21h14m26s/21h12m4s (anonymous)
    preferred global 3ffe:3705:1200:1:210:a4ff:fea0:bc97,
     life 29d23h59m25s/6d23h59m25s (public)
    preferred link-local fe80::210:a4ff:fea0:bc97, life infinite
    multicast interface-local ff01::1, 1 refs, not reportable
    multicast link-local ff02::1, 1 refs, not reportable
    multicast link-local ff02::1:ffa0:bc97, 3 refs, last reporter
    multicast link-local ff02::1:ffc6:45ee, 2 refs, last reporter
  link MTU 1500 (true link MTU 1500)
  current hop limit 64
  reachable time 22000ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 1
```

## Neighbor Unreachability Detection

The discussion of neighbor address resolution in a previous section made mention of neighbor cache entries being labeled as Incomplete or Reachable. In fact, a neighbor cache entry can be in one of five states:

- `Incomplete` Neighbor address resolution is in progress. An NS has been sent to the solicited-node multicast address for the entry, but no NA has yet been received.
- `Reachable` The address has been recently confirmed as reachable. "Recently confirmed" means that some indication of its reachability has been received within the time specified in the Reachable Time field of the RAs. If no Reachable Time has been specified in RAs, a

default Reachable Time of 30 seconds is used.

- Stale The Reachable Time has elapsed since the last positive confirmation of reachability with the destination has been received.
- Probe A confirmation of reachability is being sought by sending NS to the destination every Retransmit Time or (if no Retransmit Time has been specified) every 1000 ms.
- Delay An address is put into this state when a packet is sent to a destination that was in the Stale state. It stays in the Delay state for 5 seconds, and if no confirmation of reachability is received within that time, the state is changed to Probe. This state is an optimization to give upper-layer protocols a chance to confirm reachability before a probe NS is sent.

Reachability of a neighbor is confirmed in one of two ways:

- "Hints" from an upper-layer protocol, such as an ACK of a TCP message.
- A response to a probe of the destination address by soliciting an RA or NA. This is necessary because some upper-layer protocols, such as UDP, do not actively acknowledge the receipt of transmitted messages.

Neighbor Unreachability Detection confirms not just reachability from the neighbor's perspective,

PREV NEXT

# Looking Ahead

The purpose of this and the previous chapter was to examine the basics of IP in both of its versions. Understanding the basics of IP addressing and the fundamental processes of IP provides the foundation for understanding IP routing. The next chapter delves into the information a router needs to successfully and accurately forward a packet toward its destination.

# Review Questions

1. What is the length of an IPv6 address?
2. How are IPv6 addresses represented?
3. What are the two rules for compacting IPv6 addresses?
4. Why is it illegal to use more than one double colon in an IPv6 address?
5. What is the difference between the IPv6 addresses ::/0 and ::/128?
6. What is the part of the unicast IPv6 address that specifies the host, and what is its length?
7. What is the length of the Subnet ID portion of the unicast IPv6 address?
8. If the first 10 bits of an IPv6 address are FF80::/10, what type of address is it?
9. What type of address is 3FFE:204:100:90::1?
10. What is an anycast address?
11. What is a multicast address?
12. What is the length of the IPv6 header?
13. What is the purpose of the Flow Label field in the IPv6 header?
14. To what field in the IPv4 header does the IPv6 Next Header field correspond?
15. To what field in the IPv4 header does the IPv6 Hop Limit field correspond?
16. In what way is the IPv6 Next Header field like the IPv4 Protocol Number field, and in what way is it different?
17. How do extension headers make IPv6 packets more efficient?
18. What is the Next Header value of ICMPv6?
19. What is the significant difference between IPv4 fragmentation and IPv6 fragmentation?
20. What are the five ICMPv6 messages used by the Neighbor Discovery Protocol?
21. What is the purpose of the M and O flags in the RA?
22. What is the purpose of the Reachable Time field of the RA?
23. What is the purpose of the Retransmit Timer field in the RA?
24. What is indicated if the Router Lifetime field in the RA is set to 0?
25. What is the purpose and effect of the S flag in the NA?

26 What is the difference between stateful and stateless address autoconfiguration?

27 What two steps does MAC-to-EUI64 conversion use to derive an Interface ID?

28 When a device acquires a unicast IPv6 address it must perform Duplicate Address Detection, with one exception. What is that exception?

29 What does the prefix FF02:0:0:0:0:1:FF00::/104 signify?

30 What does IPv6 use in place of ARP and an ARP cache?

31 What is a privacy address?

32 What does an Incomplete state of an entry in the neighbor cache signify?

33 What does a Probe state of an entry in the neighbor cache signify?

34 What two ways does Neighbor Unreachability Detection use to verify two-way reachability of a neighbor?

PREV NEXT

# Chapter 3. Static Routing

This chapter covers the following subjects:

- Route Table
- Configuring Static Routes
- Troubleshooting Static Routes

An important observation from Chapter 1, "TCP/IP Review," is that the data link/physical layers and the transport/network layers, as defined by the OSI model, perform very similar duties: They provide the means for conveying data from a source to a destination across some path. The difference is that the data link/physical layers provide communications across a physical path, whereas the transport/network layers provide communications across a logical or virtual path made up of a series of data links.

Further, Chapter 1 showed that for communications to take place across a physical path, certain information about data-link identifiers and encapsulations must be acquired and stored in a database such as the ARP cache. Similarly, information that the transport/network layers require to do their job must also be acquired and stored. This information is stored in the *route table*, also known as the *routing information database (RIB)*.

This chapter examines what sort of information is required to route a packet, how that information is stored in the route table, how to enter the information into the database, and some techniques for building a routed network by entering the proper information into the proper routers' route tables.

# Route Table

To understand the kind of information that exists in the route table, it is useful to begin with an examination of what happens when a framed packet arrives at one of a router's interfaces. The data-link identifier in the frame's destination address field is examined. If it contains either the identifier of the router's interface or a broadcast identifier, the router strips off the frame and passes the enclosed packet to the network layer. At the network layer, the destination address of the packet is examined. If the destination address is either the IP address of the router's interface or an all-hosts broadcast address, the protocol field of the packet is examined and the enclosed data is sent to the appropriate internal process.[1]

[1] There is also the special case of a multicast address, which is destined for a group of devices, but not for all devices. An example of a multicast address is the class D address 224.0.0.5, reserved for all OSPF-speaking routers.

Any other destination address calls for routing. The address might be for a host on another network to which the router is attached (including the router interface attached to that network) or for a host on a network not directly connected to the router. The address might also be a directed broadcast, in which there is a distinct network or subnet address, and the remaining host bits are all ones. These addresses are also routable.

If the packet is to be routed, the router will do a route table lookup to acquire the correct route. At a minimum, each route entry in the database must contain two items:

- Destination address This is the address of the network the router can reach. As this chapter explains, the router might have more than one route to the same address, or a group of subnets of the same or of varying lengths, grouped under the same major IP network address.
- Pointer to the destination This pointer either will indicate that the destination network is directly connected to the router or it will indicate the address of another router on a directly connected link or the local interface to that link. That router, which will be one router hop closer to the destination, is a *next-hop router.*

The router will match the most specific address it can.[2] In descending order of specificity, the address may be one of the following:

[2] There are two basic procedures for finding the best match, depending upon whether the router is behaving classfully or classlessly. Classful table lookups are explained in more detail in Chapter 5, "Routing Information Protocol (RIP)," and classless table lookups are explained in Chapter 6, "RIPv2, RIPng, and Classless Routing."

- Host address (a *host route*)
- Subnet
- Group of subnets (a *summary route*)
- Major network number
- Group of major network numbers (a *supernet*)
- Default address

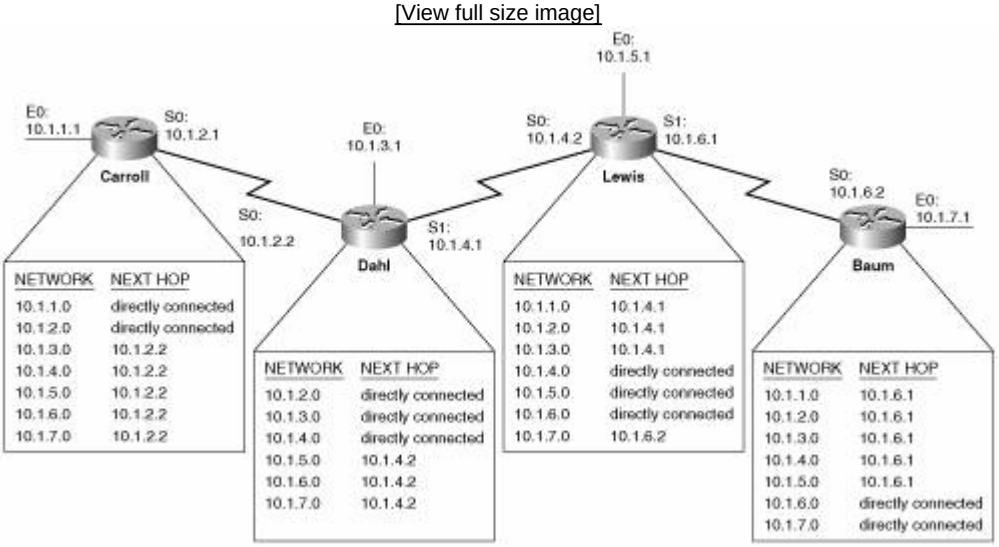
This chapter provides examples of the first four types. Supernets are covered in Chapter 6, "RIPv2, RIPng, and Classless Routing." A default address is considered a least-specific address and is matched only if no other match can be found. Default addressing is the topic of Chapter 12, "Default Routes and On-Demand Routing."

If the destination address of the packet cannot be matched to any route table entry, the packet is dropped and a Destination Unreachable ICMP message is sent to the source address.

Figure 3-1 shows a simple network and the route table entries required by each router. Of primary importance here is the "big picture," seeing how the route tables work as a whole to transport packets correctly and efficiently. The destination addresses that the router can reach are listed in the Network column of the route tables. The pointers to the destinations are in the Next Hop column.

Figure 3-1. The minimum information needed for each route table entry consists of the destination networks and the pointers to those networks.

[View full size image]




If router Carroll in Figure 3-1 receives a packet with a source address of 10.1.1.97 and a destination address of 10.1.7.35, a route table lookup determines that the best match for the destination address is subnet 10.1.7.0, reachable via next-hop address 10.1.2.2, on interface S0. The packet is sent to that next router (Dahl), which does a lookup in its own table and sees that network 10.1.7.0 is reachable via next-hop address 10.1.4.2, out interface S1. The process continues until the packet reaches router Baum. That router, receiving the packet on its interface S0, does a lookup, and sees that the destination is on one of its directly connected subnets, out E0. Routing is completed, and the packet is delivered to host 10.1.7.35 on the Ethernet link.

The routing process, as explained, assumes that the router can match its listed next-hop addresses to its interfaces. For example, router Dahl must know that Lewis's address 10.1.4.2 is reachable via interface S1. Dahl will know from the IP address and subnet mask assigned to S1 that S1 is directly connected to subnet 10.1.4.0. It then knows that 10.1.4.2, a member of the same subnet, must be connected to the same data link.

Notice that every router must have consistent and accurate information for correct packet switching to occur. For example, in Figure 3-1, an entry for network 10.1.1.0 is missing from Dahl's route table. A packet from 10.1.1.97 to 10.1.7.35 will be delivered, but when a reply is sent from 10.1.7.35 to 10.1.1.97, the packet is passed from Baum to Lewis to Dahl. Then, Dahl does a lookup and finds that it has no entry for subnet 10.1.1.0, so the packet is dropped, and an ICMP Destination Unreachable message is sent to host 10.1.7.35.

Example 3-1 shows the route table from router Lewis of Figure 3-1. The IOS command for examining the IP route table of a Cisco router is show ip route.

Example 3-1. The route table for router Lewis of Figure 3-1.

```
Lewis#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP,
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area,
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2,
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP,
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route, o - ODR
Gateway of last resort is not set
     10.0.0.0/24 is subnetted, 7 subnets
S      10.1.3.0 [1/0] via 10.1.4.1
S      10.1.2.0 [1/0] via 10.1.4.1
S      10.1.1.0 [1/0] via 10.1.4.1
S      10.1.7.0 [1/0] via 10.1.6.2
C      10.1.6.0 is directly connected, Serial1
C      10.1.5.0 is directly connected, Ethernet0
C      10.1.4.0 is directly connected, Serial0
Lewis#
```

Examine the contents of this database and compare it with the generic table shown for Lewis in Figure 3-1. A key at the top of the table explains the letters down the left side of the table. These letters indicate how each route entry was learned; in Example 3-1, all routes are tagged with either a C for "directly connected," or an S for "static entry." The statement "gateway of last resort is not set" refers to a default route.

At the top of the table is a statement indicating that the route table knows of seven subnets of the major network address 10.0.0.0, subnetted with a 24-bit mask. For each of the seven route entries, the destination subnet is shown; for the entries that are not directly connectedroutes for which the packet must be forwarded to a next-hop routera bracketed tuple indicates [administrative distance/metric] for that route. Administrative distances are introduced later in this chapter and are covered in detail in Chapter 11, "Route Redistribution."

Metrics, discussed in greater detail in Chapter 4, "Dynamic Routing Protocols," are a way for multiple routes to the same destination to be rated by preferencethe lower the metric, the "shorter" the path and so the more desirable the route. Notice that the static routes shown in Example 3-1 have a metric of 0. Finally, either the address of the directly connected interface of the next-hop router or the interface to which the destination is connected is shown.

PREV NEXT

# Configuring Static Routes

The route table acquires information in one of three ways:

- The information can be entered based on what the router knows about its directly connected subnets.
- The information can be entered manually, by means of a static route entry.
- The information can be entered automatically by one of several systems of automatic information discovery and sharing known as *dynamic routing protocols.*

The bulk of this book concerns dynamic IP routing protocols, but this discussion of static route configuration will prepare you for understanding the subsequent chapters.

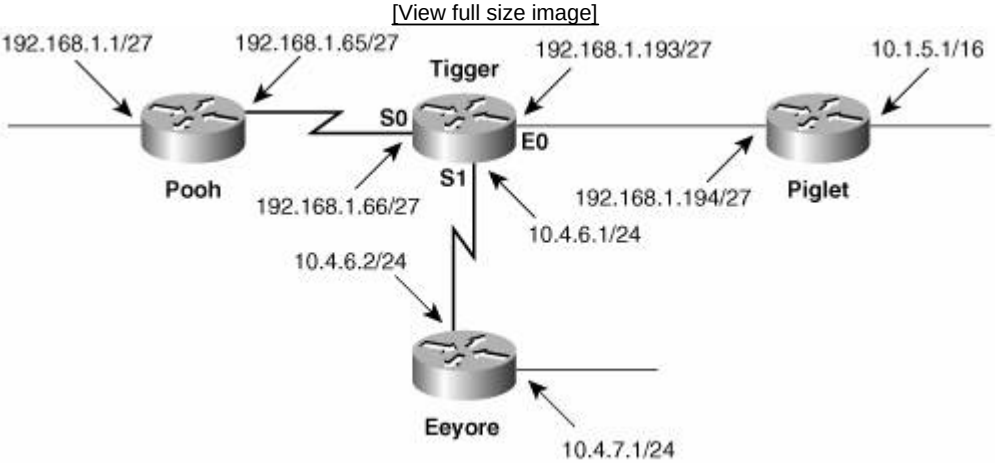
More to the point, static routing is preferred over dynamic routing in certain circumstances. As with any process, the more automatic it is, the less control you have over it. Although dynamic (automatic) routing requires much less human intervention, static routing allows very precise control over the routing behavior of a network. The price to be paid for this precision is the necessity of manual reconfiguration any time the topology of the network changes.

## Case Study: Simple IPv4 Static Routes

Figure 3-2 shows a network with four routers and six data links. Notice that the subnets of network 10.0.0.0 are *discontiguous*there is a different major network subnet (192.168.1.192, in the Tigger-to-Piglet link) separating 10.1.0.0 from the other 10.0.0.0 subnets. The subnets of 10.0.0.0 are also *variably subnetted*the subnet masks are not consistent throughout the network: Subnet 10.1.0.0 has a 16-bit mask, while 10.4.0.0 has a 24-bit mask. Finally, the subnet address of Pooh's Ethernet link is an all-zero subnet. Later chapters demonstrate that an addressing scheme with these characteristics causes problems for simpler, classful routing protocols such as RIP and IGRP; but static routes work fine here.

Figure 3-2. Routing protocols such as RIP and IGRP cannot easily route this discontiguous, variably subnetted network, but static routing will work.

[View full size image]



The procedure for statically routing a network has three steps:

1. For each data link within the network, identify all subnet or network addresses.

2. For each router, identify all data links not directly connected to that router.

3. For each router, write a route statement for each address not directly connected to it.

Writing route statements for a router's directly connected data links is unnecessary, because the addresses and masks configured on the router's interfaces cause those networks to be recorded in its route table.

For example, the network in Figure 3-2 has six subnets:

- 10.1.0.0/16
- 10.4.6.0/24
- 10.4.7.0/24
- 192.168.1.192/27
- 192.168.1.64/27
- 192.168.1.0/27

To configure static routes for Piglet, the subnets that are not directly connected are identified as follows:

- 10.4.6.0/24
- 10.4.7.0/24
- 192.168.1.64/27
- 192.168.1.0/27

These are the subnets for which static routes must be written. Example 3-2 shows the commands for entering Piglet's static routes.[3]

[3] For the static routes in this example and the subsequent examples in this chapter to work properly, two global commands must be added to the routers**: ip classless** and **ip subnet-zero**. In IOS 11.3 and later, **ip classless** is enabled by default. These commands are introduced in Chapter 6 and are mentioned here for readers who wish to try the configuration examples in a lab.

## Example 3-2. Configuring Piglet's static routes.

```
Piglet(config)# ip route 192.168.1.0 255.255.255.224 192.168.1.193
Piglet(config)# ip route 192.168.1.64 255.255.255.224 192.168.1.193
Piglet(config)# ip route 10.4.6.0 255.255.255.0 192.168.1.193
Piglet(config)# ip route 10.4.7.0 255.255.255.0 192.168.1.193
```

Following the same steps, Example 3-3 shows the route entries for the other three routers.

## Example 3-3. Route entries for Routers Pooh, Tigger, and Eeyore.

```
Pooh(config)# ip route 192.168.1.192 255.255.255.224 192.168.1.66
Pooh(config)# ip route 10.1.0.0 255.255.0.0 192.168.1.66
Pooh(config)# ip route 10.4.6.0 255.255.255.0 192.168.1.66
Pooh(config)# ip route 10.4.7.0 255.255.255.0 192.168.1.66
```

---

```
Tigger(config)# ip route 192.168.1.0 255.255.255.224 192.168.1.65
Tigger(config)# ip route 10.1.0.0 255.255.0.0 192.168.1.194
Tigger(config)# ip route 10.4.7.0 255.255.255.0 10.4.6.2
```

---

```
Eeyore(config)# ip route 192.168.1.0 255.255.255.224 10.4.6.1
Eeyore(config)# ip route 192.168.1.64 255.255.255.224 10.4.6.1
Eeyore(config)# ip route 192.168.1.192 255.255.255.224 10.4.6.1
Eeyore(config)# ip route 10.1.0.0 255.255.0.0 10.4.6.1
```

The routing commands themselves are easily read if the reader remembers that each command describes a route table entry. The command for IPv4 is ip route, followed by the address to be entered into the table, a mask for determining the network portion of the address, and the address of the directly connected interface of the next-hop router.

An alternative configuration command for IPv4 static routes specifies the interface out of which an address is reached instead of the interface address of the next-hop router. For example, Example 3-4 shows the possible route entries for Tigger.

## Example 3-4. Alternative route entries for Tigger.

```
ip route 192.168.1.0 255.255.255.224 S0
ip route 10.1.0.0 255.255.0.0 E0
ip route 10.4.7.0 255.255.255.0 S1
```

Certain conditions must be met before a static route is written into the route table. IP routing must be enabled, the next-hop address, if used, must be reachable, the exit interface must have an IP address configured on it, and the exit interface must be up.

Example 3-5 compares the route table resulting from this configuration with the route table resulting from entries pointing to a next-hop router. Notice that a certain inaccuracy is introduced: All addresses specified with a static route referring to an exit interface are entered into the table as if they are directly connected to that interface. The implications for route redistribution are discussed in Chapter 11.

A point of interest in Example 3-5 is that the header for the 10.0.0.0 subnets indicates the variable subnet masks used in the network. Variable-length subnet masking (VLSM) can be a useful tool and is discussed at length in Chapter 6.

## Example 3-5. The top route table is the result of static route entries pointing to the next-hop router. The bottom route table is the result of static routes that point to the interface a packet must exit to reach the destination network.[4]

```
Tigger#show ip route
Gateway of last resort is not set
     10.0.0.0 is variably subnetted, 3 subnets, 2 masks
C      10.4.6.0 255.255.255.0 is directly connected, Serial1
S      10.4.7.0 255.255.255.0 [1/0] via 10.4.6.2
```

```
S      10.1.0.0 255.255.0.0 [1/0] via 192.168.1.194
     192.168.1.0 255.255.255.224 is subnetted, 3 subnets
C      192.168.1.64 is directly connected, Serial0
S      192.168.1.0 [1/0] via 192.168.1.65
C      192.168.1.192 is directly connected, Ethernet0
Tigger#

Tigger#show ip route
Gateway of last resort is not set
     10.0.0.0 is variably subnetted, 3 subnets, 2 masks
C      10.4.6.0 255.255.255.0 is directly connected, Serial1
S      10.4.7.0 255.255.255.0 is directly connected, Serial1
S      10.1.0.0 255.255.0.0 is directly connected, Ethernet0
     192.168.1.0 255.255.255.224 is subnetted, 3 subnets
C      192.168.1.64 is directly connected, Serial0
S      192.168.1.0 is directly connected, Serial0
C      192.168.1.192 is directly connected, Ethernet0
Tigger#
```

[4] The key normally seen at the top of the route table (as in Figure 3-2) has been removed for clarity.

A third option for static routes is to use a combination of the outgoing interface and the next-hop address. The next-hop address is coupled with the specified exit interface. If the exit interface goes down, the route is removed from the route table, even if the next-hop address is recursively reachable via an alternate route. This minimizes table lookups associated with finding the outgoing interface associated with a next-hop address and the entry in the table appears as a route with a distance of 1, not a directly connected network.

Directing a static route to an exit broadcast interface without specifying the next-hop address can cause an excessive amount of traffic on the broadcast network, and also might eat up the router's memory. For example, look at Tigger's `ip route 10.1.0.0 255.255.0.0 E0` command. The router assumes 10.1.0.0 is directly connected, as we have seen from the route table. Therefore, when attempting to route to any address on the 10.1.0.0/16 subnet, the router sends an ARP request to find the MAC address to which to forward the packet. Each attempt to reach an address on the 10.1.0.0 network, whether the destination is valid or not, will result in an ARP request, an ARP response if a router on the broadcast network is responding on behalf of the 10.1.0.0 network (proxy ARP), and a potentially large ARP cache on the router. By appending the next-hop address to the static route entry, `ip route 10.1.0.0 255.255.0.0 E0 192.168.1.194`, the router no longer assumes that the destination is directly connected. The only ARP traffic is for the next-hop address, which only occurs for the first packet destined to a host on network 10.1.0.0, rather than for every packet destined to a new host on network 10.1.0.0.

Specify the exit interface and the next-hop address to minimize table lookups associated with finding the exit interface for a specified next-hop address, and to minimize traffic on the broadcast network.

Example 3-6 shows the difference in the static route entries in the route tables when the next-hop address is used with the exit interface.

## Example 3-6. Specifying an exit interface rather than the next-hop router address with static routing could generate excessive traffic on a broadcast network.

```
Tigger#show ip route static
     10.0.0.0/16 is subnetted, 1 subnets
S       10.1.0.0 is directly connected, Ethernet0

Tigger#show arp
```

```
Protocol   Address          Age (min)  Hardware Addr  Type   Interface
Internet   192.168.1.193          -   0004.c150.f1c0 ARPA   Ethernet0
Internet   10.1.8.1               0   0010.7b38.37d5 ARPA   Ethernet0
Internet   192.168.1.194         24   0010.7b38.37d5 ARPA   Ethernet0
Internet   10.1.5.5               0   0010.7b38.37d5 ARPA   Ethernet0
Internet   10.1.1.1               0   0010.7b38.37d5 ARPA   Ethernet0
Tigger#


Tigger#show ip route static
     10.0.0.0/16 is subnetted, 1 subnets
S       10.1.0.0 [1/0] via 192.168.1.194, Ethernet0

Tigger#show arp
Protocol  Address                Age (min)  Hardware Addr  Type  Interface
Internet  192.168.1.193                  -   0004.c150.f1c0 ARPA  Ethernet0
Internet  192.168.1.194                 22   0010.7b38.37d5 ARPA  Ethernet0
```

The first route table and ARP cache show that the static route entry was created with an exit interface and no next-hop address. The route is directly connected and there are multiple ARP cache entries for destinations on the 10.1.0.0 network. The MAC address for each entry is the same. It is the hardware address of the router with IP address 192.168.1.194. The router is sending ARP replies for all hosts on the 10.1.0.0 network. As discussed in Chapter 1, this proxy ARP is enabled by default in IOS.

The second set of tables shows the route table and ARP cache when the next-hop address is specified in addition to the exit interface. Notice the route is no longer directly connected. It is known via 192.168.1.194 and the exit interface is Ethernet 0. The ARP cache has no entries for the 10.1.0.0 network, only for the addresses that actually exist on the directly connected network, including 192.168.1.194.

## Case Study: Simple IPv6 Static Routes

IPv6 static routes are configured the same way as IPv4 static routes. The only difference is that the IPv6 prefix length of the destination network is entered rather than the dotted decimal form of the IPv4 network mask. Unlike IPv4, however, IPv6 routing is not enabled by default. Before entering a static route, IPv6 must be enabled using the `ipv6 unicast-routing` command. As with IPv4, an IPv6 address must be configured on the exit interface and the interface must be up before the static entry will be added to the route table. The command used to create a static route is `ipv6 route` followed by the network to be entered into the route table, the length, in bits of the prefix, and the address of the next-hop router, or the exit interface to be used to reach this destination.

To specify the next-hop address in the static route entry, you need to know what that address is. A detailed network drawing will help, but it may be out of date because of the dynamic nature of the Interface ID portion of the addresses. When addressing the IPv6 network, if you specify interface IDs manually rather than using the automatically constructed EUI-64 format addresses, the next-hop address will be predictable. However, if the interfaces on the data link are configured to use EUI-64 interface IDs, you only specify the first 64 bits of the address. The router determines the final 64 bits based on a MAC address. If a router is replaced, the new router will have different IPv6 addresses. (The first 64 bits will remain the same, but the final 64 bits will be different.) One way to identify the full 128-bit IPv6 address of a neighbor router is to use the Cisco Discovery Protocol (CDP) statistics. CDP displays information about neighboring routers, such as the router's hostname, router type, IOS, and the IP addresses configured on the remote end of the link. Example 3-7 displays one form of the `show cdp` command.

### Example 3-7. Cisco Discovery Protocol can tell you a lot of information

about a device's neighbors.

```
Honeybee#show cdp neighbor detail
-------------------------
Device ID: Honeytree
Entry address(es):
  IP address: 10.4.6.2
  IPv6 address: FE80::2B0:64FF:FE30:1DE0 (link-local)
  IPv6 address: FEC0::1:2B0:64FF:FE30:1DE0 (site-local)
Platform: cisco 2610, Capabilities: Router
Interface: Serial0/0.2, Port ID (outgoing port): Serial0/0.2
Holdtime : 146 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-J1S3-M), Version 12.3(6), RELEASE SOFTWARE (fc3)
Copyright (c) 1986-2004 by cisco Systems, Inc.
Compiled Wed 11-Feb-04 19:24 by kellythw

advertisement version: 2
```

Example 3-7 displays a lot of information about the neighbor router, including the router type, IOS, host name and IP addresses. The `detail` keyword is required to obtain all the information that is displayed.

Another way to determine the IPv6 address of a link is to issue the `show ipv6 interface` command. This command displays the IPv6 information relevant to an interface. Example 3-8 shows the output from the command issued on Honeybee.

Example 3-8. *show ipv6 interface* displays IPv6 information relevant to an interface, including the IPv6 EUI-64 formatted addresses.

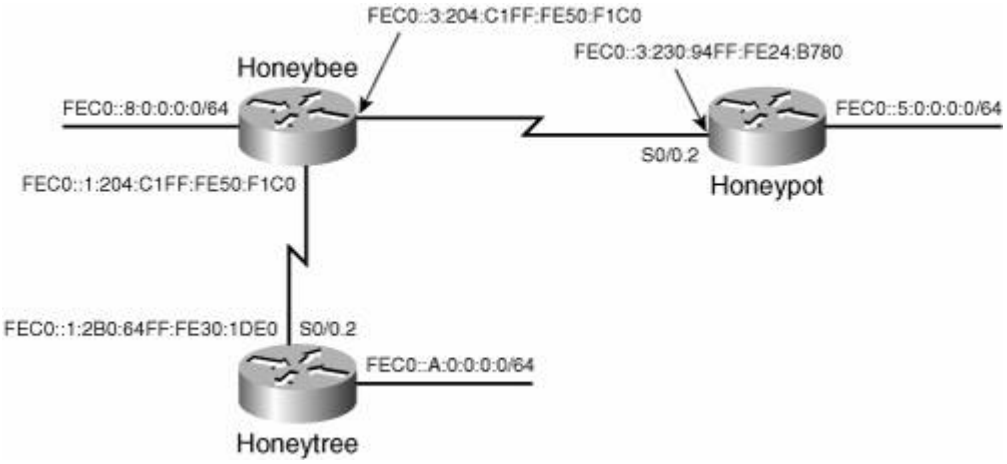
```
Honeybee#show ipv6 interface serial0/0.1
Serial0/0.1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::204:C1FF:FE50:F1C0
  Description: Link to Piglet
  Global unicast address(es):
    FEC0::3:204:C1FF:FE50:F1C0, subnet is FEC0:0:0:3::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF30:1DE0
```

Figure 3-3 shows a simple network with IPv6 addresses.[5]

[5] The interface addresses are configured with EUI-64 addresses. The addresses, therefore, are unique to each router based on MAC address. To reproduce the configuration, you'd have to determine your router's interface addresses to use as the next hop.

Figure 3-3. Static routing also works with IPv6.

[View full size image]




Example 3-9 shows the commands for entering Honeypot's IPv6 static routes.

## Example 3-9. Configuring Honeypot's IPv6 static routes.

```
ipv6 unicast-routing
interface serial 0/0.2 point-to-point
 ipv6 address fec0:0:0:3::/64 eui-64
ipv6 route fec0::1:0:0:0:0/64 fec0::3:204:c1ff:fe50:f1c0
ipv6 route fec0::a:0:0:0:0/64 fec0::3:204:c1ff:fe50:f1c0
ipv6 route fec0::8:0:0:0:0/64 fec0::3:204:c1ff:fe50:f1c0
```

Example 3-10 and Example 3-11 show the route entries for the other two routers, Honeytree and Honeybee, respectively.

## Example 3-10. Configuring IPv6 static routes for Honeytree.

```
ipv6 route fec0::8:0:0:0:0/64 fec0::1:204:c1ff:fe50:f1c0
ipv6 route fec0::3:0:0:0:0/64 fec0::1:204:c1ff:fe50:f1c0
ipv6 route fec0::5:0:0:0:0/64 fec0::1:204:c1ff:fe50:f1c0
```

## Example 3-11. Configuring IPv6 static routes for Honeybee.

```
ipv6 route fec0::a:0:0:0:0/64 fec0::1:2b0:64ff:fe30:1de0
ipv6 route fec0::5:0:0:0:0/64 fec0::3:230:94ff:fe24:b780
```

Look at the next-hop address used for Honeypot's routes, and the next-hop address used for Honeytree's routes. Honeypot's next-hop address for each route is fec0::3:204:c1ff:fe50:f1c0. The next-hop address used for Honeytree's routes is fec0::1:204:c1ff:fe50:f1c0. These addresses are those of Honeybee's interfaces to Honeypot and Honeytree, respectively. Notice that the last 64 bits of each of Honeybee's interface addresses are the same. The router uses its first encountered MAC address to form the last 64 bits of the EUI-64 formatted IPv6 addresses on each of its serial interfaces.

As with IPv4, IPv6 static routes can use the outbound interface rather than next-hop address. There is an option to enter an address after the interface as there is with IPv4. You can put either the link-local address here or a configured address. This next-hop address should be used when the exit interface is a broadcast interface, such as Ethernet.

Example 3-12 displays Honeypot's IPv6 route table with only the next-hop address specified in the ipv6 route statement. The command show ipv6 route displays the IPv6 route table. Prefixes, prefix lengths, and the next-hop address or outgoing interface are displayed, as are the administrative distance and route metric.

## Example 3-12. As with IPv4, the IPv6 static route table displays the destination network and the next-hop address used to reach the destination.

```
Honeypot#show ipv6 route
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2

L   FE80::/10 [0/0]
     via ::, Null0
C   FEC0:0:0:3::/64 [0/0]
     via ::, Serial0/0.2
L   FEC0::3:230:94FF:FE24:B780/128 [0/0]
     via ::, Serial0/0.2
S   FEC0:0:0:A::/64 [1/0]
     via FEC0::3:204:C1FF:FE50:F1C0
S   FEC0:0:0:8::/64 [1/0]
     via FEC0::3:204:C1FF:FE50:F1C0
S   FEC0:0:0:1::/64 [1/0]
     via FEC0::3:204:C1FF:FE50:F1C0
C   FEC0:0:0:5::/64 [0/0]
     via ::, Ethernet0/0
L   FEC0::5:230:94FF:FE24:B780/128 [0/0]
     via ::, Ethernet0/0
L   FF00::/8 [0/0]
     via ::, Null0
```

The static routes displayed in Example 3-12 were entered using an IPv6 next-hop address. The router must determine the exit interface associated with this IPv6 address recursively, as it does with IPv4. The entry for FEC0:0:0:A::/64 has a next-hop address of FEC0::3:204:C1FF:FE50:F1C0. Looking further into the route table, FEC0:0:0:3::/64 is connected on Serial0/0.2. Notice that the administrative distance of the static routes entered with the next-hop IPv6 address is 1 and the route metric is 0, the same as IPv4 static route entered in this way.

Routes can also be entered with the outgoing interface toward the destination network. The outgoing interface and the next-hop address can be entered together, too. Example 3-13 shows what Honeypot's static route configuration could be changed to.

## Example 3-13. Alternative static route configuration for Honeypot.

```
ipv6 route fec0::a:0:0:0:0/64 serial 0/0.2
ipv6 route fec0::8:0:0:0:0/64 serial 0/0.2
ipv6 route fec0::1:0:0:0:0/64 serial 0/0.2
ipv6 route fec0::20:0:0:0:0/62 Ethernet0/0 FE80::2B0:64FF:FE30:1DE0
```

The last entry, using the exit interface and the next-hop address will help to illustrate the difference in the route table between the two forms of the command. Example 3-14 displays

Honeypot's new route table.

### Example 3-14. Honeypot route table after changing the next hop to the exit interface.

```
Honeypot#show ipv6 route static

S   FEC0:0:0:A::/64 [1/0]
     via ::, Serial0/0.2
S   FEC0:0:0:8::/64 [1/0]
     via ::, Serial0/0.2
S   FEC0:0:0:1::/64 [1/0]
     via ::, Serial0/0.2
S   FEC0:0:0:20::/62 [1/0]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet0/0
```

One thing to notice in the route table is the administrative distance of the static route configured with an exit interface. The distance is 1, unlike IPv4 static routes configured the same way. The route does not appear to be directly connected as it does with IPv4.

The next-hop address is undetermined when you enter the outbound interface unless you specify the exit interface and the next-hop address. You can see this in the route table shown in Example 3-14. The first statement, for instance, says that FEC0:0:0:A::/64 is known via ::, Serial 0/0.2. The "::" means that the next hop is unspecified, but the outgoing interface is Serial 0/0.2. On a point-to-point serial interface, an unspecified next-hop address is not a problem. There is only one other device on that point-to-point network, and all packets are forwarded out the interface and reach the other device.

On a broadcast interface, the router must find a neighbor to which to send the packet. The router multicasts a neighbor solicitation message on the Ethernet and waits for a neighbor advertisement from the next-hop device. There is no defined proxy address resolution mechanism with IPv6, other than for mobile IPv6 nodes. A router on the Ethernet that has a route to the destination will not respond to a neighbor solicitation on behalf of another device.

For this reason, when using an exit interface to configure a static route on a broadcast network, a next-hop address must also be specified. The recommended address to use as the next-hop address is the link-local address of the next-hop router. One reason to use the link-local address is that it is not likely to change. A link-local address will only change if the interface card, or the entire router, is replaced. Even if the site is renumbered with a different IPv6 global prefix, the link-local address on the interface does not change. Another reason to use the link-local address as the next hop is to remain consistent with the addresses routers advertise in the router advertisement messages and so that processes using those addresses, such as ICMPv6 Redirect, will operate as expected.

Routers advertise their presence, along with their link-local addresses, to all IPv6 devices on broadcast networks. Hosts use the router list created from the router advertisement to determine how to forward packets off the network. If a host forwards a packet to a router, and that router knows that a second router on the network is a better choice for the host to use, the first router will send a redirect to the host. The redirect includes the link-local IPv6 address of the better choice router. When the host processes the redirect, if the better router is in its router list, the host will begin to forward packets to the better router. If the better router is not in the list (or it is listed by a different IPv6 address), the host will discard the redirect.

## Case Study: Summary Routes

A *summary route* is an address that encompasses several more specific addresses in a route table. It is the address mask used with a route entry that makes static routes as flexible as they

are; by using an appropriate address mask, it is sometimes possible to create a single summary route for several destination addresses.

For example, the preceding two case studies use a separate entry for each data link. The mask of each entry corresponds to the address mask used on the device interfaces connected to that data link. Looking again at Figure 3-2, you can see that subnets 10.4.6.0/24 and 10.4.7.0/24 could be specified to Piglet with a single entry of 10.4.0.0/16, reachable via Tigger. Likewise, subnets 192.168.1.0/27 and 192.168.1.64/27 could be accounted for in its route table with a single entry pointing to 192.168.1.0/24, also reachable via Tigger. These two route entries, 10.4.0.0/16 and 192.16.1.0/24, are summary routes.

Using summary routes, Piglet's static route entries are displayed in Example 3-15.

## Example 3-15. Piglet's static route entries are summarized into only two entries.

```
ip route 192.168.1.0 255.255.255.0 192.168.1.193
ip route 10.4.0.0 255.255.0.0 192.168.1.193
```

All subnets of network 10.0.0.0 are reachable from Pooh via Tigger, so a single entry to that major network address and a corresponding mask are all that is needed (see Example 3-16).

## Example 3-16. Pooh's static route entries for all of network 10.0.0.0 subnets are summarized into a single entry.

```
ip route 192.168.1.192 255.255.255.224 192.168.1.66
ip route 10.0.0.0 255.0.0.0 192.168.1.66
```

From Eeyore, all destination addresses beginning with 192 are reachable via Tigger. The single route entry does not even have to specify all of the Class C address bits[6], as displayed in Example 3-17.

[6] This method of summarizing a group of major network addresses with a mask shorter than the default address mask for that class is known as supernetting. This is introduced in Chapter 6.

## Example 3-17. Eeyore summarizes all routes beginning with 192 into a single entry.

```
ip route 192.0.0.0 255.0.0.0 10.4.6.1
ip route 10.1.0.0 255.255.0.0 10.4.6.1
```

Summary routes can also be applied to the IPv6 destination addresses in Figure 3-3.

Honeypot's two static routes can be summarized into a group consisting of fec0:0:0:8:: through fec0:0:0:b:: by changing the prefix length from 64 to 62, as in Example 3-18.

## Example 3-18. Honeypot summarizes IPv6 static routes.

```
ipv6 route fec0::8:0:0:0:0/62 fec0::3:204:c1ff:fe50:f1c0
```

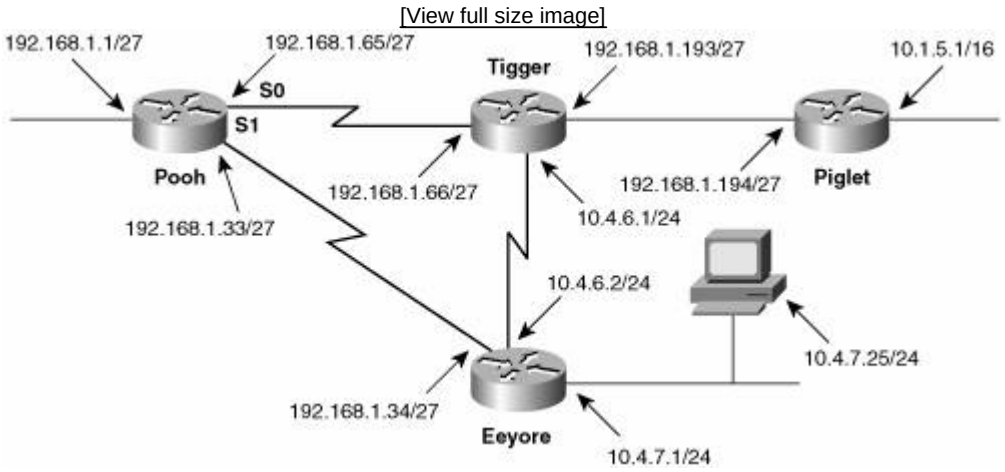
By summarizing a group of subnets or even major networks, the number of static route entries

may be reduced drasticallyin this example, by more than one-third. However, caution must be used when summarizing addresses; when done incorrectly, unexpected routing behavior may occur (see "Case Study: Tracing a Failed Route," later in this chapter). Summarization and the problems that can develop from incorrect summarization are examined in more depth in Chapters 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)," and 8, "OSPFv2."

## Case Study: Alternative Routes

In Figure 3-4, a new link has been added between Pooh and Eeyore. All packets from Pooh to the 10.0.0.0 networks will take this new path with the exception of packets destined for the host 10.4.7.25; a policy is in place stating that traffic to this host must go through Tigger. The static route commands at Pooh will be as displayed in Example 3-19.

Figure 3-4. A more direct path from Pooh to the 10.4.0.0 subnets is added to the network.




Example 3-19. Pooh's static route commands help implement a policy directing traffic through specific routers.

```
ip route 192.168.1.192 255.255.255.224 192.168.1.66
ip route 10.0.0.0 255.0.0.0 192.168.1.34
ip route 10.4.7.25 255.255.255.255 192.168.1.66
```

The first two route entries are the same as before except that the second path now points to the new interface 192.168.1.34 at Eeyore. The third entry is a *host route*, pointing to the single host 10.4.7.25 and made possible by setting the address mask to all ones. Notice that unlike the entry for the other 10.0.0.0 subnets, this host route points to Tigger's interface 192.168.1.66.

The debugging function `debug ip packet` is turned on in Pooh (see Example 3-20) to observe the paths packets take from the router as a result of the new route entries. A packet is sent from a host 192.168.1.15 to host 10.4.7.25. The first two debug trap messages show that the packet is routed from interface E0 to the next-hop router 192.168.1.66 (Tigger) out interface S0, as required, and that the reply packet was received on S0 and routed to the host 192.168.1.15 out E0.

Example 3-20. Debugging verifies that the new route entries at Pooh

are working correctly.

```
Pooh#debug ip packet
IP packet debugging is on
Pooh#
IP: s=192.168.1.15 (Ethernet0), d=10.4.7.25 (Serial0), g=192.168.1.66, forward
IP: s=10.4.7.25 (Serial0), d=192.168.1.15 (Ethernet0), g=192.168.1.15, forward
Pooh#
IP: s=192.168.1.15 (Ethernet0), d=10.4.7.100 (Serial1), g=192.168.1.34, forward
IP: s=10.4.7.100 (Serial0), d=192.168.1.15 (Ethernet0), g=192.168.1.15, forward
Pooh#
```

Next, a packet is sent from host 192.168.1.15 to host 10.4.7.100. Packets destined for any host on 10.0.0.0 subnets, other than host 10.4.7.25, should be routed across the new link to Eeyore's interface 192.186.1.34. The third debug message verifies that this is indeed happening. However, the fourth message shows something that at first might be surprising. The response from 10.4.7.100 to 192.168.1.15 arrived on Pooh's interface S0 from Tigger.

Remember that the route entries in the other routers have not changed from the original example. This result might or might not be desired, but it does illustrate two characteristics of static routes:

- First, if the network topology changes, the routers that are required to know about those changes must be reconfigured.
- Second, static routes can be used to create very specific routing behavior. In this example, perhaps it is desirable to have traffic taking one path in one direction and another path in the opposite direction.

A final observation about this example is that packets routed from Pooh to subnet 10.1.5.0 take a less-than-optimal route, from Pooh to Eeyore to Tigger instead of directly from Pooh to Tigger. Example 3-21 shows a more efficient configuration for Router Pooh.

### Example 3-21. Configuring a more efficient static route on Router Pooh.

```
ip route 192.168.1.192 255.255.255.224 192.168.1.66
ip route 10.0.0.0 255.0.0.0 192.168.1.34
ip route 10.1.0.0 255.255.0.0 192.168.1.66
ip route 10.4.7.25 255.255.255.255 192.168.1.66
```

The third entry will now send all packets for subnet 10.1.5.0 directly to Tigger.

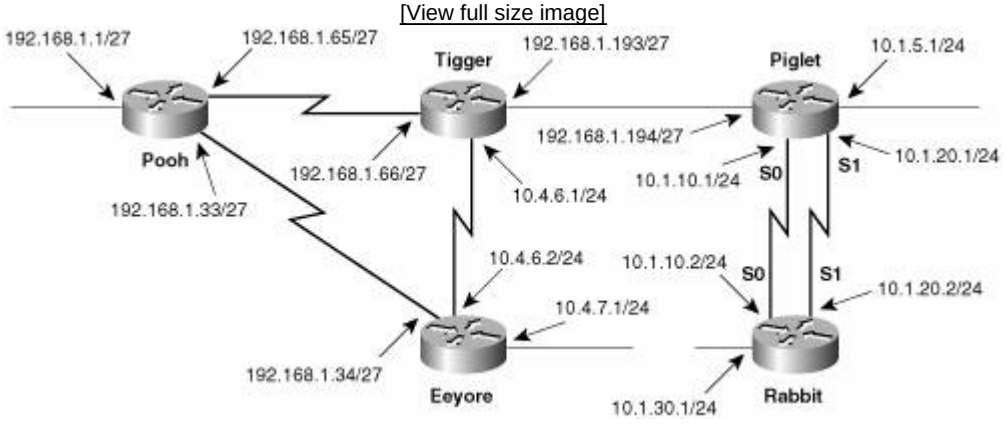
## Case Study: Floating Static Routes

Unlike other static routes, a floating static route is less preferred than other routes in the route table. It appears in the table only under the special circumstance of the failure of a more-preferred route.

In Figure 3-5, a new router (Rabbit) is connected to Piglet with two parallel links. One link connects their respective Serial 0 interfaces, and the second connection has been added between the two Serial 1 interfaces. This second link has been added for redundancy: If the primary link 10.1.10.0 fails, floating static routes will direct traffic across the backup link 10.1.20.0.

Figure 3-5. A new router has been connected to Piglet. Two serial links are used: one for the primary link and one for the backup link.

[View full size image]




Additionally, the mask on Piglet's Ethernet interface has changed from 10.1.5.1/16 to 10.1.5.1/24. This change allows the single route entry at Tigger

```
ip route 10.1.0.0 255.255.0.0 192.168.1.194
```

to point not only to 10.1.5.0 but also to all of the new subnets used in association with the new router.

To create the floating static route, Example 3-22 and Example 3-23 show the route entries for both Piglet and Rabbit, respectively.

## Example 3-22. Route entries for Piglet to create a floating static route.

```
ip route 192.168.1.0 255.255.255.0 192.168.1.193
ip route 10.4.0.0 255.255.0.0 192.168.1.193
ip route 10.1.30.0 255.255.255.0 10.1.10.2
ip route 10.1.30.0 255.255.255.0 10.1.20.2 50
```

## Example 3-23. Route entries for Rabbit to create a floating static route.

```
ip route 10.4.0.0 255.255.0.0 10.1.10.1
ip route 10.4.0.0 255.255.0.0 10.1.20.1 50
ip route 10.1.5.0 255.255.255.0 10.1.10.1
ip route 10.1.5.0 255.255.255.0 10.1.20.1 50
ip route 192.168.0.0 255.255.0.0 10.1.10.1
ip route 192.168.0.0 255.255.0.0 10.1.20.1 50
```

Two entries at Piglet point to Rabbit's network 10.1.30.0; one specifies a next-hop address of Rabbit's S0 interface, and the other specifies a next-hop address of Rabbit's S1 interface. Rabbit has similar double entries for every route.

Notice that all static routes using subnet 10.1.20.0 are followed by a 50. This number specifies an *administrative distance*, which is a measure of preferability; when duplicate paths to the same network are known, the router will prefer the path with the lower administrative distance. At first this idea sounds like a metric; however, a metric specifies the preferability of a route,

whereas an administrative distance specifies the preferability of the means by which the route was discovered.

For example, IPv4 static routes pointing to a next-hop address have an administrative distance of 1, and static routes referencing an exit interface have an administrative distance of 0. If two static routes point to the same destination, but one references a next-hop address and one references an exit interface, the latterwith the lower administrative distancewill be preferred.

By increasing the administrative distances of the static routes traversing subnet 10.1.20.0 to 50, they become less preferred than the routes traversing subnet 10.1.10.0. Example 3-24 shows three iterations of Rabbit's route table. In the first table, all routes to nonconnected networks use a next-hop address of 10.1.10.1. The bracketed numbers associated with each route indicate an administrative distance of 1 and a metric of 0 (because no metrics are associated with static routes).

## Example 3-24. When the primary link 10.1.10.0 fails, the backup link 10.1.20.0 is used. When the primary link is restored, it is again the preferred path.

```
Rabbit#show ip route
10.0.0.0 is variably subnetted, 5 subnets, 2 masks
C 10.1.10.0 255.255.255.0 is directly connected, Serial0
S 10.4.0.0 255.255.0.0 [1/0] via 10.1.10.1
S 10.1.5.0 255.255.255.0 [1/0] via 10.1.10.1
C 10.1.30.0 255.255.255.0 is directly connected, Ethernet0
C 10.1.20.0 255.255.255.0 is directly connected, Serial1
S 192.168.0.0 255.255.0.0 [1/0] via 10.1.10.1
Rabbit#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed state to down
%LINK-3-UPDOWN: Interface Serial0, changed state to down
Rabbit#show ip route
10.0.0.0 is variably subnetted, 4 subnets, 2 masks
S 10.4.0.0 255.255.0.0 [50/0] via 10.1.20.0
S 10.1.5.0 255.255.255.0 [50/0] via 10.1.20.1
C 10.1.30.0 255.255.255.0 is directly connected, Ethernet0
C 10.1.20.0 255.255.255.0 is directly connected, Serial1
S 192.168.0.0 255.255.0.0 [50/0] via 10.1.20.1
Rabbit#
%LINK-3-UPDOWN: Interface Serial0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed state to up
Rabbit#show ip route
10.0.0.0 is variably subnetted, 5 subnets, 2 masks
C 10.1.10.0 255.255.255.0 is directly connected, Serial0
S 10.4.0.0 255.255.0.0 [1/0] via 10.1.10.1
S 10.1.5.0 255.255.255.0 [1/0] via 10.1.10.1
C 10.1.30.0 255.255.255.0 is directly connected, Ethernet0
C 10.1.20.0 255.255.255.0 is directly connected, Serial1
S 192.168.0.0 255.255.0.0 [1/0] via 10.1.10.1
Rabbit#
```

Next, trap messages announce that the state of the primary link connected to Serial 0 has changed to "down," indicating a failure. A look at the second iteration of the route table shows that all nonconnected routes now point to a next-hop address of 10.1.20.1. Because the more-preferred entry is no longer available, the router has switched to the less-preferred backup link, with the administrative distance of 50 indicated in the brackets. And because subnet 10.1.10.0 has failed, it no longer shows up in the route table as a directly connected network.

Before the third iteration of the route table, trap messages indicate that the state of the primary

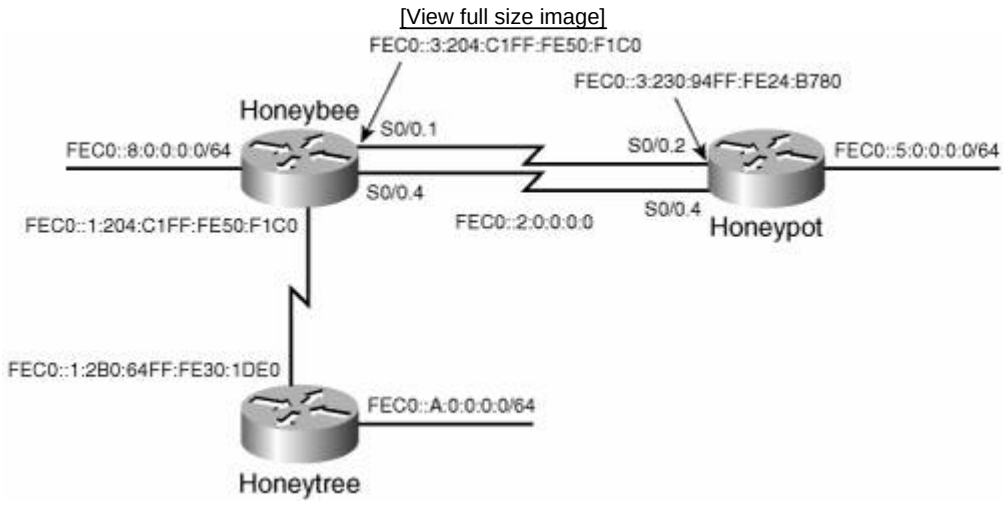
link has changed to "up." The route table then shows that subnet 10.1.10.0 is again in the table, and the router is again using the next-hop address of 10.1.10.1.

Chapter 11 discusses the administrative distances associated with the various dynamic routing protocols, but it can be said here that the administrative distances of all dynamic routing protocols are substantially higher than 1. Therefore, by default, a static route to a network will always be preferred over a dynamically discovered route to the same network.

## Case Study: IPv6 Floating Static Routes

IPv6 floating static route statements work the same way as IPv4. A second link has been added to the IPv6 network of Figure 3-3 between Honeypot and Honeybee, to route IPv6 traffic if the primary link fails (see Figure 3-6).

Figure 3-6. Backup link added between two IPv6 routers can be used to recover from a primary link failure with floating static routes.




Example 3-25 shows Honeypot's configuration with new static route entries, which have an administrative distance greater than 1. Similar static routes are entered on Honeybee, as shown in Example 3-26.

Example 3-25. Honeypot is configured with floating static routes to be used over the new redundant parallel link to Honeybee.

```
ipv6 route FEC0::/62 FEC0::3:204:C1FF:FE50:F1C0
ipv6 route FEC0::/62 FEC0::2:204:C1FF:FE50:F1C0 50
ipv6 route FEC0:0:0:8::/62 FEC0::3:204:C1FF:FE50:F1C0
ipv6 route FEC0:0:0:8::/62 FEC0::2:204:C1FF:FE50:F1C0 50
```

Example 3-26. Honeybee is configured with floating static routes to be used over the new redundant parallel link to Honeypot.

```
ipv6 route FEC0:0:0:5::/64 FEC0::3:230:94FF:FE24:B780
ipv6 route FEC0:0:0:5::/64 FEC0::2:230:94FF:FE24:B780 50
ipv6 route FEC0:0:0:A::/64 FEC0::1:2B0:64FF:FE30:1DE0
```

IPv6 traffic from Honeypot will continue to be forwarded out Serial0/0.2, unless this link goes down.

Example 3-27 shows Honeypot's route table with the routes known via the fec0::3:0:0:0:0/64 subnet installed. Both routes have an administrative distance of 1. Then, interface S0/0.2 goes down. The backup routes, with administrative distance of 50, get installed in the route table. After the interface S0/0.2 comes back up, the routing process learns of better routes to the destinations and installs the routes with an administrative distance of 1 back into the table.

### Example 3-27. Static routes with high administrative distances are installed in the IPv6 route table only when the lower administrative distance routes are deleted.

```
Honeypot#show ipv6 route static
S   FEC0::/62 [1/0]
     via FEC0::3:204:C1FF:FE50:F1C0
S   FEC0:0:0:8::/62 [1/0]
     via FEC0::3:204:C1FF:FE50:F1C0
Honeypot#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/2, changed state to down
%LINK-3-UPDOWN: Interface Serial0/2, changed state to down

Honeypot#show ipv6 route static
S   FEC0::/62 [50/0]
     via FEC0::2:204:C1FF:FE50:F1C0
S   FEC0:0:0:8::/62 [50/0]
     via FEC0::2:204:C1FF:FE50:F1C0
Honeypot#
%LINK-3-UPDOWN: Interface Serial0/2, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/2, changed state to up
```

The default administrative distance of IPv6 static routes, whether specified with an exit interface or a next-hop address, is 1. When static routes to a destination are specified both ways, the routes are considered equal, and load sharing (described in the next case study) occurs.

## Case Study: Load Sharing

The problem with the configuration used in the previous section is that under normal circumstances the second link is never utilized. The bandwidth available on the link is wasted. *Load sharing* allows routers to take advantage of multiple paths to the same destination by sending packets over all the available routes.

Load sharing can be equal cost or unequal cost, where *cost* is a generic term referring to whatever metric (if any) is associated with the route:

- Equal-cost load sharing distributes traffic equally among multiple paths with equal metrics. In this case, load sharing can also be called *load balancing.*
- Unequal-cost load sharing distributes packets among multiple paths with different metrics. The traffic is distributed in inverse proportion to the cost of the routes. That is, paths with lower costs are assigned more traffic, and paths with higher costs are assigned less traffic.

Some routing protocols support both equal-cost and unequal-cost load sharing, whereas others

support only equal cost. Static routes, which have no metric, support only equal-cost load sharing.

To configure the parallel links in Figure 3-5 for load sharing using static routes, Example 3-28 shows the route entries for Piglet and Example 3-29 shows the route entries for Rabbit.

## Example 3-28. Configuring parallel links for load sharing using static routes: route entries for Piglet.

```
ip route 192.168.1.0 255.255.255.0 192.168.1.193
ip route 10.4.0.0 255.255.0.0 192.168.1.193
ip route 10.1.30.0 255.255.255.0 10.1.10.2
ip route 10.1.30.0 255.255.255.0 10.1.20.2
```

## Example 3-29. Configuring parallel links for load sharing using static routes: route entries for Rabbit.

```
ip route 10.4.0.0 255.255.0.0 10.1.10.1
ip route 10.4.0.0 255.255.0.0 10.1.20.1
ip route 10.1.5.0 255.255.255.0 10.1.10.1
ip route 10.1.5.0 255.255.255.0 10.1.20.1
ip route 192.168.0.0 255.255.0.0 10.1.10.1
ip route 192.168.0.0 255.255.0.0 10.1.20.1
```

These entries were also used in the previous section for floating static routes, except both links now use the default administrative distance of 1. Rabbit's route table, shown in Example 3-30, now has two routes to each destination.

## Example 3-30. This route table indicates that there are two paths to the same destination networks. The router will balance the load across these multiple paths.

```
Rabbit#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route
Gateway of last resort is not set
     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C      10.1.10.0/24 is directly connected, Serial0
S      10.1.5.0/24 [1/0] via 10.1.10.1
                   [1/0] via 10.1.20.1
S      10.4.0.0/16 [1/0] via 10.1.10.1
                   [1/0] via 10.1.20.1
C      10.1.20.0/24 is directly connected, Serial1
S      192.168.0.0/16 [1/0] via 10.1.10.1
                      [1/0] via 10.1.20.1
Rabbit#
```

IPv6 works the same way as IPv4. The static routes in Honeypot's route table, shown in Example 3-31, will yield the route table shown in Example 3-32.

Example 3-31. Honeypot's IPv6 static routes are configured for load-balancing, with each destination prefix using two different next-hop addresses.

```
ipv6 route FEC0::/62 FEC0::2:204:C1FF:FE50:F1C0
ipv6 route FEC0::/62 FEC0::3:204:C1FF:FE50:F1C0
ipv6 route FEC0:0:0:8::/62 FEC0::2:204:C1FF:FE50:F1C0
ipv6 route FEC0:0:0:8::/62 FEC0::3:204:C1FF:FE50:F1C0
```

Example 3-32. The router will load balance over these two IPv6 paths to the same destination networks.

```
Honeypot#show ipv6 route static
S   FEC0::/62 [1/0]
     via FEC0::3:204:C1FF:FE50:F1C0
     via FEC0::2:204:C1FF:FE50:F1C0
S   FEC0:0:0:8::/62 [1/0]
     via FEC0::3:204:C1FF:FE50:F1C0
     via FEC0::2:204:C1FF:FE50:F1C0
```

Load sharing is also either per destination or per packet.

## Load Sharing and Cisco Express Forwarding

Per destination load sharing distributes the load according to destination address. Given two paths to the same network, all packets for one destination may travel over the first path, all packets for a second destination on the same network may travel over the second path, all packets for a third destination may again be sent over the first path, and so on. This is the default type of load sharing used by *Cisco Express Forwarding (CEF)*. On most platforms, CEF is the default switching mode for IPv4, but not IPv6.

CEF is a very efficient switching process. Its forwarding information is obtained and stored in tables before any packet needs to use the information. CEF builds a forwarding information base (FIB) with information obtained from the route table. All the destination networks entered in the route table are entered into the CEF FIB. If the route table is stable and not changing, the FIB will not change. CEF uses a separate table, the adjacency table, to maintain Layer 2 forwarding information for each entry in the FIB. The adjacency table is created as Layer 2 information is learned, through IP ARP or IPv6 Neighbor Discovery, for instance. Both the FIB and adjacency table are created before packets need to be forwarded. Not even the first packet to a destination is process switched, as is the case with other switching schemes.

CEF performs per-destination load sharing by default. This is actually per source-destination pair load sharing. All traffic that has a particular source address and is destined to a specific destination address will exit the same interface. Traffic with a different source/destination address pair may exit the next interface.

Per packet load sharing is another method available to CEF switched IPv4 packets. IPv6 CEF only supports per destination load sharing. Per packet load sharing means that one packet is sent over one link, the next packet is sent over the next link, even if this next packet is to the same destination as the first, and so on, given equal-cost paths. If the paths are unequal cost, the load sharing may be one packet over the higher-cost link for every three packets over the lower-cost link, or some other proportion depending upon the ratio of costs. Per packet load sharing may distribute the load more evenly than per destination load sharing, depending upon the number of different source-destination pairs, but because the packets to a given destination will be taking different paths, the packets are likely to arrive out of order, which is unacceptable for some applications, such as Voice over IP.

To determine if CEF is enabled globally on a router, use the commands show ip cef and show ipv6 cef. If it is not enabled by default, you can turn it on globally using the command ip cef for IPv4. To enable CEF for IPv6, first enable CEF for IPv4, then use the command ipv6 cef.

Per packet load sharing is enabled for IPv4 using the interface command ip load-sharing per-packet. The command ip load-sharing per-destination re-enables per destination load sharing. You can see which type of load sharing is enabled using the show cef interface command. This displays the CEF information that is configured on the interface.

The router determines whether to use CEF switching based on the ingress interface and the type of source and destination address. The ingress interface must be configured with CEF switching for the router to even consider using CEF. If CEF is configured on the ingress interface, CEF will attempt to switch the packet. If for some reason the packet cannot be CEF switched, CEF *punts* the packet down to the next-best and available switching method. For IPv4, this would be *fast switching*, if it is enabled on the interface. For IPv6, this would be *process switching.*

You can verify that CEF is enabled on an interface using the commands show cef interface {interface} and show ipv6 cef {interface} detail.

## Per Destination Load Sharing and Fast Switching

IOS performs per destination load sharing on exit interfaces configured with fast switching. Fast switching is the default IOS switching mode in some routers.

Fast switching works as follows:

1. When a router switches the first packet to a particular destination, a route table lookup is performed and an exit interface is selected.
2. The necessary data-link information to frame the packet for the selected interface is then retrieved (from the ARP cache, for instance), and the packet is encapsulated and transmitted.
3. The retrieved route and data-link information is then entered into a fast switching cache.
4. As subsequent packets to the same destination enter the router, the information in the fast cache allows the router to immediately switch the packet without performing another route table and ARP cache lookup.

While switching time and processor utilization are decreased, fast switching means that all packets to a specific destination, not source-destination pair, are routed out the same interface. When packets addressed to a different host on the same network enter the router and an alternate route exists, the router may send all packets for that destination on the alternate route. Therefore, the best the router can do is balance traffic on a per destination basis.

## Per Packet Load Sharing and Process Switching

Process switching simply means that for every packet, the router performs a route table lookup, selects an interface, and then looks up the data link information. Because each routing decision is independent for each packet, all packets to the same destination are not forced to use the same interface. To enable process switching on an interface, use the command no ip route-cache for IPv4. You don't have to do anything to enable process switching for IPv6. It is enabled by default.

In Example 3-33, host 192.168.1.15 has sent six pings to host 10.1.30.25. Using debug ip packet, the ICMP echo request and echo reply packets are observed at Piglet. Looking at the exit interfaces and the forwarding addresses, it can be observed that both Piglet and Rabbit are

using S0 and S1 alternately. Note that the command debug ip packet allows only process switched packets to be observed. Fast switched packets are not displayed.

### Note

The debug ip packet command displays only process switched packets.

### Example 3-33. This router is alternating between S0 and S1 to send packets to the same destination. Notice that the router on the other end of the two links is doing the same thing with the reply packets.

```
Piglet#debug ip packet
IP packet debugging is on
Piglet#
IP: s=192.168.1.15 (Ethernet0), d=10.1.30.25 (Serial0), g=10.1.10.2, forward
IP: s=10.1.30.25 (Serial0), d=192.168.1.15 (Ethernet0), g=192.168.1.193, forward
IP: s=192.168.1.15 (Ethernet0), d=10.1.30.25 (Serial1), g=10.1.20.2, forward
IP: s=10.1.30.25 (Serial1), d=192.168.1.15 (Ethernet0), g=192.168.1.193, forward
IP: s=192.168.1.15 (Ethernet0), d=10.1.30.25 (Serial0), g=10.1.10.2, forward
IP: s=10.1.30.25 (Serial0), d=192.168.1.15 (Ethernet0), g=192.168.1.193, forward
IP: s=192.168.1.15 (Ethernet0), d=10.1.30.25 (Serial1), g=10.1.20.2, forward
IP: s=10.1.30.25 (Serial1), d=192.168.1.15 (Ethernet0), g=192.168.1.193, forward
IP: s=192.168.1.15 (Ethernet0), d=10.1.30.25 (Serial0), g=10.1.10.2, forward
IP: s=10.1.30.25 (Serial0), d=192.168.1.15 (Ethernet0), g=192.168.1.193, forward
IP: s=192.168.1.15 (Ethernet0), d=10.1.30.25 (Serial1), g=10.1.20.2, forward
IP: s=10.1.30.25 (Serial1), d=192.168.1.15 (Ethernet0), g=192.168.1.193, forward
Piglet#
```

Like many design choices, per packet load balancing has a price. The traffic may be distributed more evenly among the various links than with per destination load balancing, but the lower switching time and processor utilization of fast switching are lost.

## Which Switching Method Will Be Used?

IOS makes switching decisions based on the configuration of the inbound interface first. If CEF is configured on an inbound interface, the packet will be CEF switched regardless of the configuration on the outbound interface.

If CEF is not enabled on the inbound interface, then IOS processes and forwards the packet, and based on the configuration of the outbound interface, subsequent packets will be fast-switched or process switched. Table 3-1 shows which switching method will be used based on configuration of inbound and outbound interfaces.

### Table 3-1. IOS switching determination is based on configuration of inbound and outbound interfaces.

| Inbound Configuration | Outbound Configuration | Switching Method Used |
|---|---|---|
| CEF | Process | CEF |

| Inbound Configuration | Outbound Configuration | Switching Method Used |
|---|---|---|
| CEF | Fast | CEF |
| Process | CEF | Fast (or process if IPv6) |
| Process | Fast | Fast |
| Fast | CEF | Fast (or process if IPv6) |
| Fast | Process | Process |

IOS will switch a packet using CEF only if CEF is enabled on the inbound interface. If CEF is not configured on the inbound interface, the configuration of the exit interface determines the switching method. Notice that when process or fast-switching is configured inbound and CEF is configured on the outbound interface, fast-switching is used. CEF is only used if it is configured on the ingress interface. For IPv4, fast-switching is enabled outbound, even if CEF is enabled on the interface.

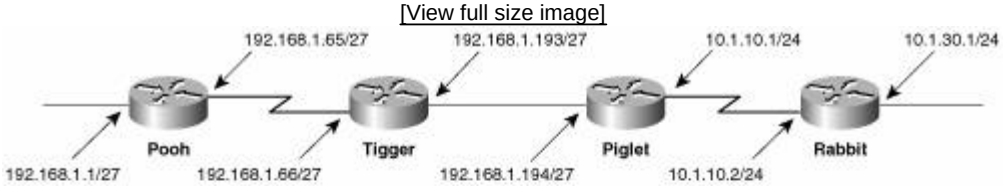
There are times when a packet will not be switched using CEF even if it is enabled (for example, if access-list logging is enabled and a packet will be logged). Packets will be punted down to the next fastest switching method. For IPv4, the next fastest switching method is fast-switching. For IPv6, this is process switching.

## Case Study: Recursive Table Lookups

All route entries do not necessarily need to point to the next-hop router. Figure 3-7 shows a simplified version of the network of Figure 3-5. In this network, Pooh is configured as displayed in Example 3-34.

Figure 3-7. To reach network 10.1.30.0, Pooh must perform three route table lookups.

[View full size image]




```
Pooh#show ip route
Codes:  C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
        i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
        U - per-user static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 2 subnets
S       10.1.10.0 [1/0] via 192.168.1.194
S       10.1.30.0 [1/0] via 10.1.10.2

     192.168.1.0/27 is subnetted, 3 subnets
C       192.168.1.64 is directly connected, Serial0
C       192.168.1.0 is directly connected, Loopback0
C       192.168.1.192 [1/0] via 192.168.1.66
Pooh#
```

Example 3-34. Pooh's static route entries use various addresses as the

next-hop parameter. The addresses are not necessarily the actual next-hop router interface.

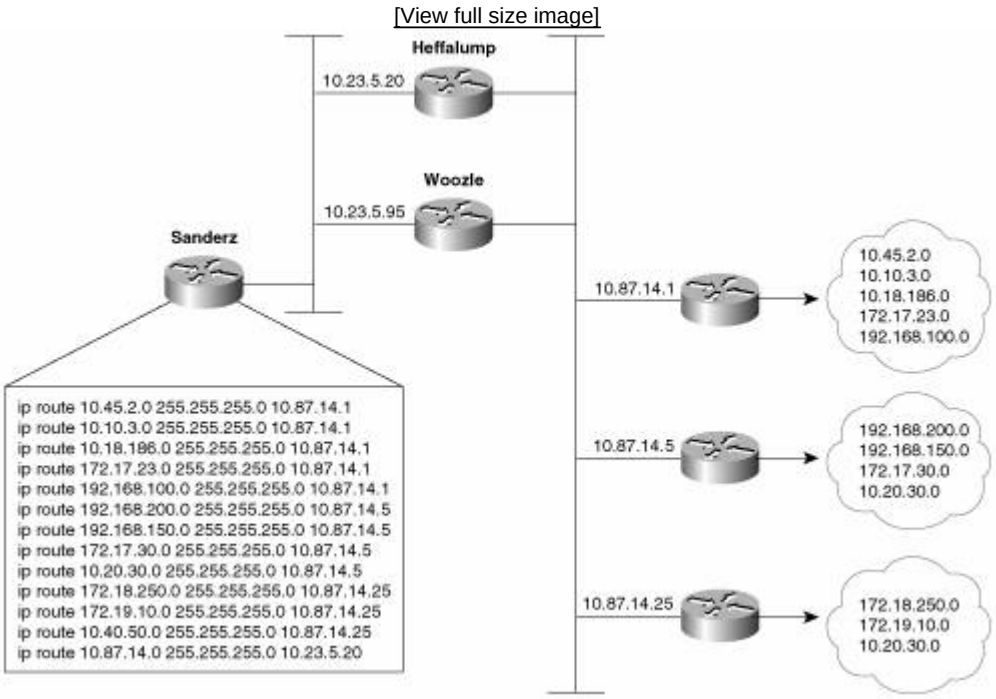
```
ip route 10.1.30.0 255.255.255.0 10.1.10.2
ip route 10.1.10.0 255.255.255.0 192.168.1.194
ip route 192.168.1.192 255.255.255.224 192.168.1.66
```

If Pooh needs to send a packet to host 10.1.30.25, it will look into its route table and find that the subnet is reachable via 10.1.10.2. Because that address is not on a directly connected network, Pooh must again consult the table to find that network 10.1.10.0 is reachable via 192.168.1.194. That subnet is also not directly connected, so a third table lookup is called for. Pooh will find that 192.168.1.192 is reachable via 192.168.1.66, which is on a directly connected subnet. The packet can now be forwarded.

Because each table lookup costs processor time, under normal circumstances forcing a router to perform multiple lookups is a poor design decision. Fast switching significantly reduces these adverse effects by limiting the recursive lookups to the first packet to each destination, but a justification should still be identified before using such a design.

Figure 3-8 shows an example of an instance in which recursive lookups might be useful. Here, Sanderz reaches all networks via Heffalump. However, the network administrator plans to eliminate Heffalump and repoint all of Sanderz's routes through Woozle. The first 12 entries point not to Heffalump, but to the appropriate router attached to the 10.87.14.0 subnet. The last entry specifies that the 10.87.14.0 subnet is reached via Heffalump.

Figure 3-8. Configuring Sanderz for recursive lookups enables the network administrator to redirect all of that router's exit traffic from Heffalump to Woozle by changing one route entry.




With this configuration, all of Sanderz's entries can be repointed through Woozle simply by changing the last static entry as in Example 3-35.

Example 3-35. Sanderz's routing can easily be modified to forward all routes through a different next-hop router simply by changing one static route entry.

```
Sanderz(config)# ip route 10.87.14.0 255.255.255.0 10.23.5.95
Sanderz(config)# no ip route 10.87.14.0 255.255.255.0 10.23.5.20
```

Had all the static routes referenced 10.23.5.20 as the next-hop address, it would have been necessary to delete all 13 lines and type 13 new lines. Nevertheless, the effort saved in retyping static routes must be weighed carefully against the extra processing burden that recursive

PREV NEXT

# Troubleshooting Static Routes

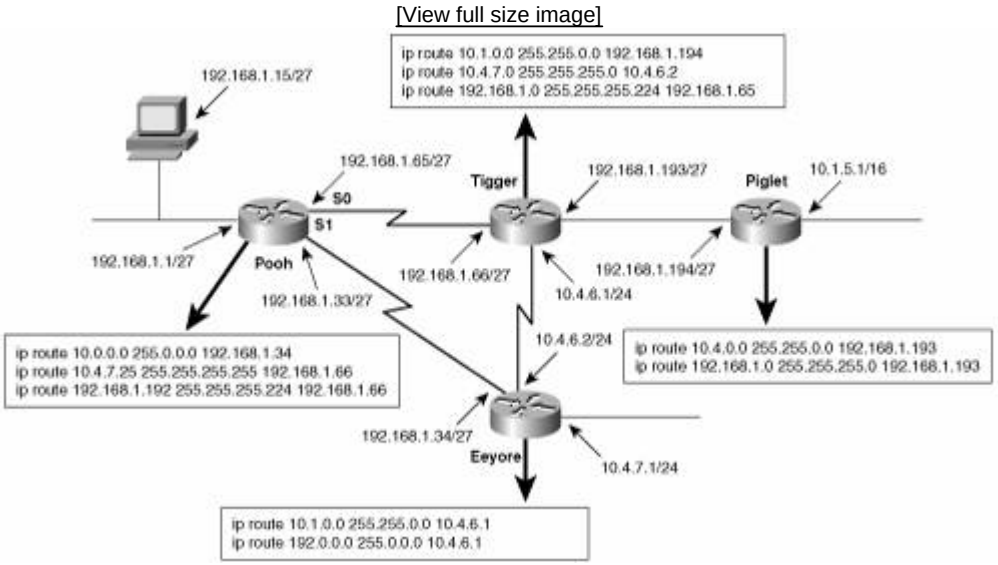
Followers of the assorted American political scandals of the past 30 or so years will have heard a congressional investigator ask the question, "What did he know and when did he know it?" The same question serves a networking investigator well. When troubleshooting routing problems, the first step should almost always be to examine the route table. What does the router know? How long has the information been in the route table? Does the router know how to reach the destination in question? Is the information in the route table accurate? Knowing how to trace a route is essential to successfully troubleshooting a network.

## Case Study: Tracing a Failed Route

Figure 3-9 shows a previously configured network, with each router's associated static routes. A problem has been discovered. Devices on subnet 192.168.1.0/27, connected to Pooh's Ethernet interface, can communicate with devices on subnet 10.1.0.0/16 just fine. However, when a ping is sent from Pooh itself to subnet 10.1.0.0/16, the ping fails (see Example 3-36). This seems strange If packets being routed by Pooh successfully reach their destinations, why do packets originated b the same router fail?

Figure 3-9. Packets from subnet 192.168.1.0/27 to subnet 10.1.0.0/16 are routed correctly, but Pooh itself cannot ping any device on 10.1.0.0/16.

[View full size image]




Example 3-36. A device on subnet 192.168.1.0/27 successfully pings Piglet's Ethernet interface, but pings from Pooh fail.

```
C:\WINDOWS>ping 10.1.5.1
Pinging 10.1.5.1 with 32 bytes of data:
Reply from 10.1.5.1: bytes=32 time=22ms TTL=253
Reply from 10.1.5.1: bytes=32 time=22ms TTL=253
```

```
Reply from 10.1.5.1: bytes=32 time=22ms TTL=253
Reply from 10.1.5.1: bytes=32 time=22ms TTL=253

Pooh#ping 10.1.5.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echoechoes to 10.1.5.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Pooh#
```

Addressing this problem requires tracing the route of the ping. First, Pooh's route table is examine (Example 3-37). The destination address of 10.1.5.1 matches the route entry for 10.0.0.0/8, whic (according to the table) is reached via the next-hop address 192.168.1.34one of Eeyore's interfaces.

## Example 3-37. A packet with a destination address of 10.1.5.1 matches the route entry for 10.0.0.0/8 and will be forwarded to the next-hop router at 192.168.1.34.

```
Pooh#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
     10.0.0.0 is variably subnetted, 2 subnets, 2 masks
S      10.0.0.0 255.0.0.0 [1/0] via 192.168.1.34
S      10.4.7.25 255.255.255.255 [1/0] via 192.168.1.66
     192.168.1.0 255.255.255.224 is subnetted, 4 subnets
C      192.168.1.64 is directly connected, Serial0
C      192.168.1.32 is directly connected, Serial1
C      192.168.1.0 is directly connected, Ethernet0
S      192.168.1.192 [1/0] via 192.168.1.66
Pooh#
```

Next the route table for Eeyore must be examined (Example 3-38). The destination address 10.1.5.1 matches the entry 10.1.0.0/16, with a next-hop address of 10.4.6.1. This address is one of Tigger's interfaces.

## Example 3-38. 10.1.5.1 matches the entry for 10.1.0.0/16 and will be forwarded to 10.4.6.1.

```
Eeyore#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
      10.0.0.0 is variably subnetted, 3 subnets, 2 masks
C       10.4.6.0 255.255.255.0 is directly connected, Serial1
C       10.4.7.0 255.255.255.0 is directly connected, Ethernet0
S       10.1.0.0 255.255.0.0 [1/0] via 10.4.6.1
      192.168.1.0 255.255.255.224 is subnetted, 1 subnets
C       192.168.1.32 is directly connected, Serial0
S       192.0.0.0 255.0.0.0 [1/0] via 10.4.6.1
```

```
Eeyore#
```

Example 3-39 shows Tigger's route table. The destination address matches the entry for 10.1.0.0/16 and will be forwarded to 192.168.1.194, which is at Piglet.

## Example 3-39. 10.1.5.1 matches the entry for 10.1.0.0/16 and will be forwarded to 192.168.1.194.

```
Tigger#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
     10.0.0.0 is variably subnetted, 3 subnets, 2 masks
C      10.4.6.0 255.255.255.0 is directly connected, Serial1
S      10.4.7.0 255.255.255.0 [1/0] via 10.4.6.2
S      10.1.0.0 255.255.0.0 [1/0] via 192.168.1.194
     192.168.1.0 255.255.255.224 is subnetted, 3 subnets
C      192.168.1.64 is directly connected, Serial0
S      192.168.1.0 [1/0] via 192.168.1.65
C      192.168.1.192 is directly connected, Ethernet0
Tigger#
```

Piglet's route table (Example 3-40) reveals that the target network, 10.1.0.0, is directly connectec In other words, the packet has arrived. The target address 10.1.5.1 is Piglet's own interface to tha network. Because the path to the address has just been verified as good, we can assume that the ICMP echo packets from Pooh are reaching the destination.

## Example 3-40. The destination network, 10.1.0.0, is directly connected to Piglet.

```
Piglet#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
     10.0.0.0 255.255.0.0 is subnetted, 2 subnets
C      10.1.0.0 is directly connected, Ethernet1
S      10.4.0.0 [1/0] via 192.168.1.193
     192.168.1.0 is variably subnetted, 2 subnets, 2 masks
S      192.168.1.0 255.255.255.0 [1/0] via 192.168.1.193
C      192.168.1.192 255.255.255.224 is directly connected, Ethernet0
Piglet#
```

The next step is to trace the path of the responding ICMP echo reply packets. To trace this path, you need to know the source address of the echo packet. That address will be the destination address of the echo reply packet. The source address of a packet that is originated from a router is the address of the interface from which the packet was transmitted.[7] In this example, Pooh originally forwarded the echo packet to 192.168.1.34. Figure 3-9 shows that the source address o that packet is 192.168.1.33. So this address is the destination address to which Piglet will send th echo reply.

[7] Unless the Extended Ping utility is used to set the source address to something different.

Referring again to Piglet's route table in Example 3-34, 192.168.1.33 will match the entry for 192.168.1.0/24 and be forwarded to 192.168.1.193, which is another of Tigger's interfaces. A re-examination of Tigger's route table in Example 3-39 at first suggests that there is an entry for 192.168.1.0. But take care to interpret correctly the information that is actually there.

Compare the entries in Tigger's route table for the 10.0.0.0 subnets to those of the 192.168.1.0 subnets. The heading for the former says that 10.0.0.0 is variably subnetted; in other words, Tigger's static route to subnet 10.4.7.0 uses a 24-bit mask, and the static route to subnet 10.1.0.( uses a 16-bit mask. The table records the correct mask at each subnet.

The heading for 192.168.1.0 is different. This heading states that Tigger knows of three subnets o 192.168.1.0 and that they all have a mask of 255.255.255.224. This mask will be used on the destination address of 192.168.1.33 to derive a destination network of 192.168.1.32/27. The rout table has entries for 192.168.1.64/27, 192.168.1.0/27, and 192.168.1.192/27. There is no entry for 192.168.1.32/27, so the router does not know how to reach this subnet.

The problem, then, is that the ICMP echo reply packet is being dropped at Tigger. One solution is t create another static route entry for network 192.168.1.32, with a mask of 255.255.255.224 and pointing to a next hop of either 192.168.1.65 or 10.4.6.2. Another solution would be to change the mask in the existing static route entry for 192.168.1.0 from 255.255.255.224 to 255.255.255.0.

The moral of this story is that when you are tracing a route, you must consider the complete communication process.

Note

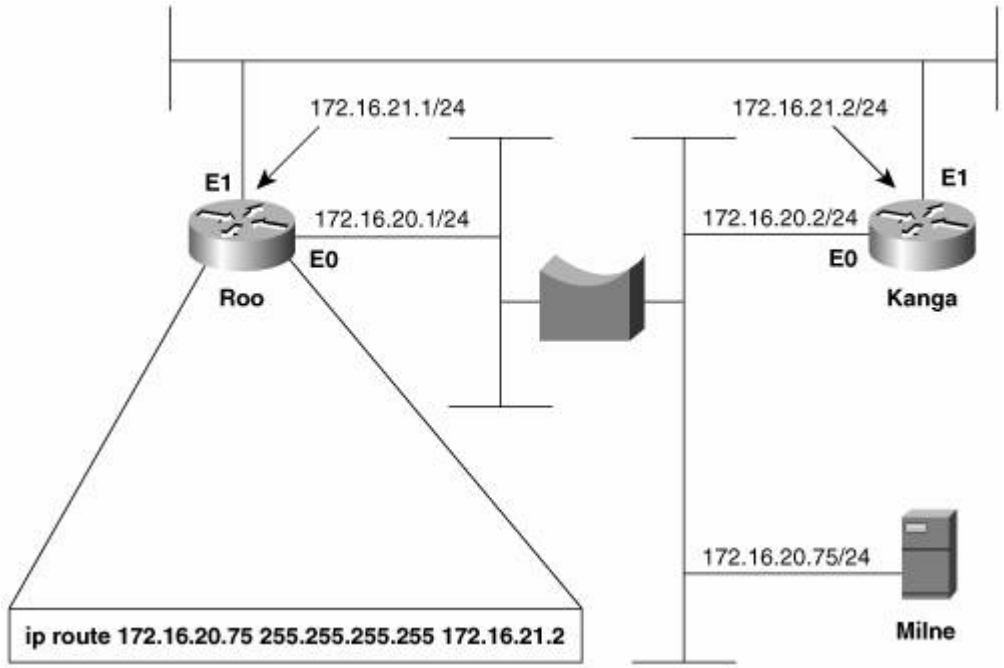
Check both the destination and return path when a route fails.

Verify not only that the path to a destination is good but also that the return path is good.

## Case Study: A Protocol Conflict

Figure 3-10 shows two routers connected by two Ethernet networks, one of which includes a simpl bridge. This bridge handles traffic for several other links not shown and occasionally becomes congested. The host Milne is a mission-critical server; the network administrator is worried about traffic to Milne being delayed by the bridge, so a static host route has been added in Roo that will direct packets destined for Milne across the top Ethernet, avoiding the bridge.

Figure 3-10. A host route directs packets from Roo to Milne across the top Ethernet, avoiding the occasionally congested bridge.




This solution seems to be logical, but it isn't working. After the static route was added, packets routed through Roo no longer reach the server. Not only that, but packets routed through Kanga r longer reach the server, although no changes were made to that router.

The first step, as always, is to check the route table. Roo's route table (Example 3-41) indicates that packets with a destination address of 172.16.20.75 will, in fact, be forwarded to Kanga's E1 interface, as desired. Kanga is directly connected to the destination network, so no more routing should be occurring; a quick check verifies that the Ethernet interfaces are functioning at both Kanga and at Milne.

## Example 3-41. Roo's route table, showing the static host route to Milne via Kanga's E1 interface.

```
Roo#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 3 subnets, 2 masks
C       172.16.20.0/24 is directly connected, Ethernet0
C       172.16.21.0/24 is directly connected, Ethernet1
S       172.16.20.75/32 [1/0] via 172.16.21.2
Roo#
```

In Example 3-42, a trace is performed from Roo to Milne, and a symptom is found. Instead of delivering the packets to Milne, Kanga is forwarding them to Roo's E0 interface. Roo forwards the packets to Kanga's E1 interface, and Kanga sends the packets right back to Roo. A routing loop seems to have occurred, but why?

## Example 3-42. A trace from Roo to Milne reveals that Kanga is forwarding the packets back to Roo instead of delivering them to the

correct destination.

```
Roo#trace 172.16.20.75
Type escape sequence to abort.
Tracing the route to 172.16.20.75
1 172.16.21.2 0 msec 0 msec 0 msec
2 172.16.20.1 4 msec 0 msec 0 msec
3 172.16.21.2 4 msec 0 msec 0 msec
4 172.16.20.1 0 msec 0 msec 4 msec
5 172.16.21.2 0 msec 0 msec 4 msec
6 172.16.20.1 0 msec 0 msec 4 msec
7 172.16.21.2 0 msec 0 msec 4 msec
8 172.16.20.1 0 msec 0 msec 4 msec
9 172.16.21.2 4 msec 0 msec 4 msec
10 172.16.20.1 4 msec 0 msec 4 msec
11 172.16.21.2 4 msec
Roo#
```

The suspicious aspect of all of this is that Kanga should not be routing the packet, which appears t be the case.

Kanga should recognize that the destination address of the packet is for its directly connected network 172.16.20.0 and should be using the data link to deliver the packet to the host. Therefore suspicion should fall on the data link. Just as the route table should be examined to see whether the router has the correct information for reaching a network across a logical path, the ARP cache should be examined to see whether the router has the correct information for reaching a host across a physical path.

Example 3-43 shows the ARP cache for Kanga. The IP address for Milne is in Kanga's cache as expected, with a MAC identifier of 00e0.1e58.dc39. When Milne's interface is checked, though, it shows that it has a MAC identifier of 0002.6779.0f4c; Kanga has somehow acquired incorrect information.

## Example 3-43. Kanga's ARP cache has an entry for Milne, but the associated data-link identifier is wrong.

```
Kanga#show arp
Protocol Address     Age (min)  Hardware Addr   Type  Interface
Internet 172.16.21.1         2  00e0.1e58.dc3c  ARPA  Ethernet1
Internet 172.16.20.2         -  00e0.1e58.dcb1  ARPA  Ethernet0
Internet 172.16.21.2         -  00e0.1e58.dcb4  ARPA  Ethernet1
Internet 172.16.20.75        2  00e0.1e58.dc39  ARPA  Ethernet0
Kanga#
```

Another look at Kanga's ARP table reveals that the MAC identifier associated with Milne is suspiciously similar to the MAC identifier of Kanga's own Cisco interfaces. (The MAC addresses wit no ages associated with them are for the router's interfaces.) Because Milne is not a Cisco product the first three octets of its MAC identifier should be different from the first three octets of Kanga's MAC identifier. The only other Cisco product in the network is Roo, so its ARP cache is examined (Example 3-44); 00e0.1e58.dc39 is the MAC identifier of Roo's E0 interface.

## Example 3-44. Roo's ARP cache shows that the MAC identifier that Kang has for Milne is actually Roo's E0 interface.

```
Roo#show arp
Protocol Address     Age (min)  Hardware Addr    Type   Interface
```

```
Internet 172.16.21.1        -   00e0.1e58.dc3c   ARPA   Ethernet0
Internet 172.16.20.1        -   00e0.1e58.dc39   ARPA   Ethernet0
Internet 172.16.20.2        7   00e0.1e58.dcb1   ARPA   Ethernet0
Internet 172.16.21.2        7   00e0.1e58.dcb4   ARPA   Ethernet1
Roo#
```

So Kanga mistakenly believes that the E0 interface of Roo is Milne. It frames packets destined for Milne with a destination identifier of 00e0.1e58.dc39; Roo accepts the frame, reads the destinatio address of the enclosed packet, and routes the packet back to Kanga.

But how did Kanga get the wrong information? The answer is proxy ARP. When Kanga first receive a packet for Milne, it will use ARP to determine that device's data-link identifier. Milne responds, b Roo also hears the ARP request on its E0 interface. Because Roo has a route to Milne on a differen network from the one on which it received the ARP Request, it issues a proxy ARP in reply. Kanga receives Milne's ARP reply and enters it into the ARP cache. The proxy ARP reply from Roo arrives later because of the delay of the bridge. The original ARP cache entry is overwritten by what Kang thinks is new information.

There are two solutions to the problem. The first is to turn off proxy ARP at Roo's E0 with the following command sequence:

```
Roo(config)#interface e0
Roo(config-if)#no ip proxy-arp
```

The second solution is to configure a static ARP entry for Milne at Kanga with the following command:

```
Kanga(config)#arp 172.16.20.75 0002.6779.0f4c arpa
```

This entry will not be overwritten by any ARP reply. Example 3-45 shows the static ARP entry bein entered and the resulting ARP cache at Kanga. Note that because the entry is static, no age is associated with it.

### Example 3-45. A static ARP entry on Kanga corrects the problem caused by proxy ARP.

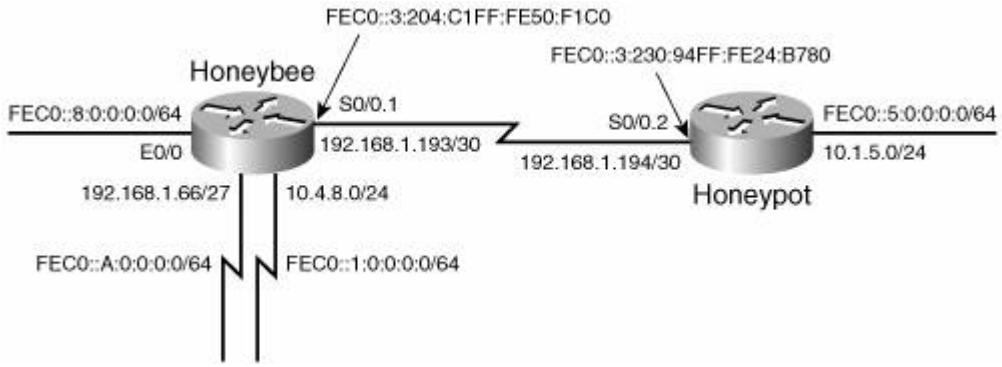
```
Kanga(config)#arp 172.16.20.75 0002.6779.0f4c arpa
Kanga(config)#^Z
Kanga#
%SYS-5-CONFIG_I: Configured from console by console
Kanga#show arp
Protocol Address     Age (min)   Hardware Addr    Type   Interface
Internet 172.16.21.1        10   00e0.1e58.dc3c   ARPA   Ethernet1
Internet 172.16.20.2         -   00e0.1e58.dcb1   ARPA   Ethernet0
Internet 172.16.21.2         -   00e0.1e58.dcb4   ARPA   Ethernet1
Internet 172.16.20.75        -   0002.6779.0f4c   ARPA
Kanga#
```

The circumstances of the network should help determine which of the two solutions is best. A stati ARP entry means that if the network interface at Milne is ever replaced, someone must remember to change the ARP entry to reflect the new MAC identifier. On the other hand, turning off proxy AR is a good solution only if no hosts make use of it.

## Case Study: A Replaced Router

Figure 3-11 shows two routers, Honeypot and Honeybee, connected via a Frame Relay link. The routers are configured for both IPv4 and IPv6, and use static routes to create the route tables.

Figure 3-11. A simple network with both IPv4 and IPv6.




Example 3-46 shows the route configurations for Honeypot and Example 3-47 shows the route configurations for Honeybee.

## Example 3-46. Route configurations for Honeypot.

```
ip route 10.4.0.0 255.255.0.0 192.168.1.193
ip route 192.168.1.0 255.255.255.0 192.168.1.193
ipv6 route FEC0::/62 Serial 0/0.2
ipv6 route FEC0:0:0:8::/62 FEC0::3:204:C1FF:FE50:F1C0
```

## Example 3-47. Route configurations for Honeybee.

```
ip route 10.1.0.0 255.255.0.0 192.168.1.194
ipv6 route FEC0:0:0:5::/64 Serial 0/0.1
```

Example 3-48 shows pings and traces between the two routers are working fine. Both IPv4 and IPv6 traffic is working fine between the two sites.

## Example 3-48. Pings and traces show two routers can successfully communicate with each other.

```
Honeypot#ping
Protocol [ip]:
Target IP address: 10.4.8.1
Extended commands [n]: y
Source address or interface: 10.1.5.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.4.8.1, timeout is 2 seconds:
Packet sent with a source address of 10.1.5.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/32/33 ms

Honeypot#ping fec0:0:0:8:204:c1ff:fe50:f1c0
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FEC0::8:204:C1FF:FE50:F1C0, timeout is 2 secon
ds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/32/33 ms

Honeypot#trace
Protocol [ip]: ipv6
Target IPv6 address: fec0:0:0:8:204:c1ff:fe50:f1c0
Source address: fec0:0:0:5:230:94ff:fe24:b780
Type escape sequence to abort.
Tracing the route to FEC0::8:204:C1FF:FE50:F1C0
  1 FEC0::3:204:C1FF:FE50:F1C0 24 msec 24 msec 24 msec

Honeypot#trace
Protocol [ip]: ipv6
Target IPv6 address: fec0::a:0:0:0:1
Source address: fec0::5:230:94ff:fe24:b780
Type escape sequence to abort.
Tracing the route to FEC0:0:0:A::1
 1 FEC0::3:204:C1FF:FE50:F1C0 24 msec 24 msec 24 msec
```

Honeypot can reach Honeybee's Ethernet address from its own Ethernet address, for both IPv4 an IPv6. Another address, FEC0:0:0:A::1, is also reachable from Honeypot, via Honeybee.

Honeybee, however, is due to be replaced with a new router. The router is replaced, Honeybee's configuration is loaded into the new router, and all the interfaces are up. Testing shows that IPv4 traffic is working between the two sites, but IPv6 traffic to some addresses fails (see Example 3-49).

## Example 3-49. After a router replacement, IPv4 works fine, while IPv6 fails.

```
Honeypot#ping
Protocol [ip]:
Target IP address: 10.4.8.1
Extended commands [n]: y
Source address or interface: 10.1.5.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.4.8.1, timeout is 2 seconds:
Packet sent with a source address of 10.1.5.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/30/32 ms

Honeypot#ping fec0:0:0:8:204:c1ff:fe50:f1c0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FEC0::8:204:C1FF:FE50:F1C0, timeout is 2 seconds
...H.
Success rate is 0 percent (0/5)

Honeypot#trace
Protocol [ip]: ipv6
Target IPv6 address: fec0::8:204:c1ff:fe50:f1c0
Source address: fec0::5:230:94ff:fe24:b780
Type escape sequence to abort.
Tracing the route to FEC0::8:204:C1FF:FE50:F1C0
  1 FEC0::3:2B0:64FF:FE30:1DE0 24 msec 24 msec 24 msec
  2  *  !H  *
```

```
Honeypot#trace
Protocol [ip]: ipv6
Target IPv6 address: fec0::a:0:0:0:1
Source address: fec0::5:230:94ff:fe24:b780
Type escape sequence to abort.
Tracing the route to FEC0:0:0:A::1
 1 FEC0::3:2B0:64FF:FE30:1DE0 24 msec 20 msec 20 msec
```

Pings and traces to the IPv6 address FEC0::8:204:c1FF:FE50:F1C0, Honeybee's Ethernet IPv6 address, fail. A trace to FEC0::A:0:0:0:1 is still successful.

Take a look at Honeypot's route tables in Example 3-50. Honeypot's IPv6 route table shows a stati route entry for FEC0:0:0:8::/62 via the next-hop address FEC0::3:204:C1FF:FE50:F1C0.

## Example 3-50. The IPv6 route table has not changed. Static route are installed.

```
Honeypot#show ipv6 route
IPv6 Routing Table - 8 entries
L   FE80::/10 [0/0]
     via ::, Null0
S   FEC0::/62 [1/0]
     via ::, Serial0/0.2
C   FEC0:0:0:3::/64 [0/0]
     via ::, Serial0/0.2
L   FEC0::3:230:94FF:FE24:B780/128 [0/0]
     via ::, Serial0/0.2
C   FEC0:0:0:5::/64 [0/0]
     via ::, Ethernet0/0
L   FEC0::5:230:94FF:FE24:B780/128 [0/0]
     via ::, Ethernet0/0
S   FEC0:0:0:8::/62 [1/0]
     via FEC0::3:204:C1FF:FE50:F1C0
```

FEC0:0:0:8::/62 is known via FEC0::3:204:C1FF:FE50:F1C0, and FEC0:0:0:3::/64 is connected t Serial0/0.2. Traffic for both FEC0:0:0:8::/64 and FEC0:0:0:A::/64 are forwarded out Serial 0/0.2 to Honeybee.

These route entries are exactly the same as they were before the router Honeybee was replaced. So what's the problem? Maybe Honeybee's Ethernet didn't come up after the router was replaced. Example 3-51 displays the state of Honeybee's Ethernet interface.

## Example 3-51. *show ipv6 interface ethernet 0/0* displays the status of the interface and some information about the IPv6 configuration.

```
Honeybee#show ipv6 interface e 0/0
Ethernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::2B0:64FF:FE30:1DE0
  Global unicast address(es):
    FEC0::8:2B0:64FF:FE30:1DE0, subnet is FEC0:0:0:8::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF30:1DE0
```

The interface is up and IPv6 is configured on it. According to the documentation and diagrams for the network, this is the interface to which a ping was successful before the router replacement, an now fails.

Look closer at the Ethernet output. The last 64 bits of the Global Unicast Address have changed. The address is now FEC0::8:2B0:64FF:FE30:1DE0. The IPv6 address is an EUI-64 address; therefore, the last 64 bits of the address are determined by the MAC address on the interface. When the router was replaced, the MAC address changed. FEC0::8:204:C1FF:FE50:F1C0 no longe exists on the router. This is why the address is no longer pingable.

But, Honeypot's static route points to the next-hop address FEC0::3:204:c1FF:FE50:F1C0 for the destination FEC0:0:0:8::/62, according to the route table (Example 3-50), and if the MAC addresses have changed, this address is no longer the serial interface's address. Example 3-52 displays Honeybee's new serial interface address.

### Example 3-52. All IPv6 serial interfaces on a router configured with EUI 64 addresses will have the same final 64 bits, which are derived from a MAC address on the router.

```
Honeybee#show ipv6 interface serial0/0.1
Serial0/0.1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::2B0:64FF:FE30:1DE0
  Description: Link to Piglet
  Global unicast address(es):
    FEC0::3:2B0:64FF:FE30:1DE0, subnet is FEC0:0:0:3::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF30:1DE0
```

Honeypot's IPv6 static route for addresses in the range FEC0:0:0:8::/62 directs traffic to the next hop address FEC0::3:204:C1FF:FE50:F1C0. The configured next-hop address is the MAC address the old router, yet traffic using this route (traffic destined to FEC0:0:0:A::1) is still routed successfully.

Even though a next-hop address is specified, and the next-hop address is not valid anymore, the old address and the new address of Honeybee's serial interface belong to the same subnet (FEC0:0:0:3::/64), and as seen in Honeypot's route table, this subnet is connected to Honeypot's interface Serial 0/0.2. Through recursive forwarding table lookups, the traffic destined for FEC0:0:0:A::1 is forwarded out Serial 0/0.2, even though the specified next-hop address does not exist.

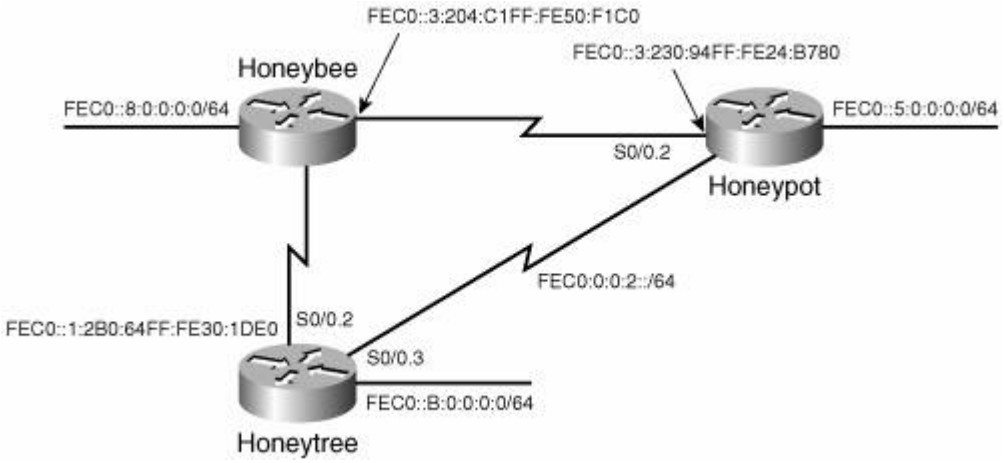
Whenever a router or interface card is replaced, remember to modify any static route entries that reference an EUI-64 IPv6 address on the old router.

## Case Study: Tracing An IPv6 Failed Route

A link is added to the network in Figure 3-3 between Honeypot and Honeytree to have an alternative route in case the primary route becomes unavailable. Figure 3-12 shows the new network. The IPv6 applications using the network are not sensitive to delay, but they are bandwidth-intensive, so the network administrator decides to use the new link also for load sharin Static routes are entered in each router to take advantage of the new link.

### Figure 3-12. Alternative route added to the IPv6 network creates a triangle, with alternate paths to reach each router.

[View full size image]




As Example 3-53 shows, routes are added into Honeypot to create a second route for each of the router's existing routes. Similar routes are entered for Honeytree and Honeybee in Example 3-54 and Example 3-55.

## Example 3-53. Honeypot's router configurations for the new network in Figure 3-12.

```
ipv6 route FEC0::/62 FEC0::2:2B0:64FF:FE30:1DE0
ipv6 route FEC0::/62 FEC0::3:204:C1FF:FE50:F1C0
ipv6 route FEC0:0:0:8::/62 FEC0::2:2B0:64FF:FE30:1DE0
ipv6 route FEC0:0:0:8::/62 FEC0::3:204:C1FF:FE50:F1C0
```

## Example 3-54. Honeytree's router configurations for the new network ir Figure 3-12.

```
ipv6 route FEC0:0:0:3::/64 FEC0::2:230:94FF:FE24:B780
ipv6 route FEC0:0:0:3::/64 FEC0::1:204:C1FF:FE50:F1C0
ipv6 route FEC0:0:0:5::/64 FEC0::2:230:94FF:FE24:B780
ipv6 route FEC0:0:0:5::/64 FEC0::1:204:C1FF:FE50:F1C0
ipv6 route FEC0:0:0:8::/64 FEC0::1:204:C1FF:FE50:F1C0
ipv6 route FEC0:0:0:8::/64 FEC0::2:230:94FF:FE24:B780
```

## Example 3-55. Honeybee's router configurations for the new network in Figure 3-12.

```
ipv6 route FEC0:0:0:2::/64 FEC0::1:2B0:64FF:FE30:1DE0
ipv6 route FEC0:0:0:2::/64 FEC0::3:230:94FF:FE24:B780
ipv6 route FEC0:0:0:5::/64 FEC0::1:2B0:64FF:FE30:1DE0
ipv6 route FEC0:0:0:5::/64 FEC0::3:230:94FF:FE24:B780
ipv6 route FEC0:0:0:B::/64 FEC0::3:230:94FF:FE24:B780
ipv6 route FEC0:0:0:B::/64 FEC0::1:2B0:64FF:FE30:1DE0
```

IPv6 routing is now intermittently failing. Sometimes pings work, sometimes not. A look at the route tables shows all static routes in place as designed. Pings from Honeypot to Honeybee's Ethernet interface address result in success sometimes, and host unreachable at other times (see

Example 3-56).

## Example 3-56. IPv6 pings are sometimes successful.

```
Honeypot#ping fec0::8:204:c1ff:fe50:f1c0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FEC0::8:204:C1FF:FE50:F1C0, timeout is 2 secon
ds:
!!HHH
Success rate is 40 percent (2/5), round-trip min/avg/max = 60/90/120 ms
Honeypot#
```

Debugging IPv6 ICMP packets on Honeypot (see Example 3-57) reveal some reply packets receive successfully, and some ICMP destination unreachable messages coming from Honeytree.

## Example 3-57. Debugging on Honeypot shows some successful packets; others fail.

```
ICMPv6: Sending echo request to FEC0::8:204:C1FF:FE50:F1C0
ICMPv6: Received ICMPv6 packet from FEC0::8:204:C1FF:FE50:F1C0, type 129
ICMPv6: Received echo reply from FEC0::8:204:C1FF:FE50:F1C0
ICMPv6: Sending echo request to FEC0::8:204:C1FF:FE50:F1C0
ICMPv6: Received ICMPv6 packet from FEC0::2:2B0:64FF:FE30:1DE0, type 1
ICMPv6: Received ICMP unreachable code 3 from FEC0::2:2B0:64FF:FE30:1DE0
```

Debugging ICMPv6 on Honeytree (see Example 3-58) shows that IPv6 packets are coming in S0/0.3 and S0/0.2.

## Example 3-58. Debugging shows packets arriving on two different interfaces.

```
Honeytree#
 IPV6: source FEC0::2:230:94FF:FE24:B780 (Serial0/0.3)
       dest FEC0::8:204:C1FF:FE50:F1C0 (Serial0/0.2)
       traffic class 0, flow 0x0, len 100+4, prot 58, hops 63, forwarding
 IPV6: source FEC0::8:204:C1FF:FE50:F1C0 (Serial0/0.2)
       dest FEC0::2:230:94FF:FE24:B780 (Serial0/0.3)
       traffic class 0, flow 0x0, len 100+4, prot 58, hops 63, forwarding
 IPV6: source FEC0::2:230:94FF:FE24:B780 (Serial0/0.3)
       dest FEC0::8:204:C1FF:FE50:F1C0 (Serial0/0.3)
       traffic class 0, flow 0x0, len 100+4, prot 58, hops 63, destination is not
connected
 IPv6: SAS picked source FEC0::2:2B0:64FF:FE30:1DE0 for FEC0::2:230:94FF:FE24:B780
(Serial0/0.3)
 ICMPv6: Sending ICMP unreachable code 3 to FEC0::2:230:94FF:FE24:B780 about
FEC0::8:204:C1FF:FE50:F1C0
```

Packets with a source address fec0::2:230:94ff:fe24:b780 are arriving at Honeytree, interface S0/0.3 (from Honeypot over the link connecting Honeypot to Honeytree). The exit interface, for destination fec0::8:204:c1ff:fe50:f1c0, is either S0/0.2 or S0/0.3. Honeytree is performing per-packet load-sharing, alternating forwarding packets between S0/0.2 and S0/0.3. When the outgoing interface to the destination is S0/0.2 and the incoming interface is S0/0.3, the packet is

forwarded. When IOS forwards the packet to S0/0.3, the same interface on which the packet arrived, pings fail, the router generates an ICMP error message.

A packet destined out the same serial interface on which it arrived indicates a routing loop. Since the routers are process switching and thus per packet load sharing, each router alternates forwarding packets using each of its two route entries. Some packets will thus arrive on the same interface to which the packet is destined.

PREV NEXT

# Looking Ahead

For precise control over routing behavior in a network, static routing is a powerful tool. However, if topology changes are prevalent, the demands of manual reconfiguration may make static routing administratively unfeasible. Dynamic routing protocols enable a network to respond quickly and automatically to topology changes. Before examining the details of specific IP routing protocols, the general issues surrounding all dynamic protocols must be examined. The next chapter introduces dynamic routing protocols.

# Summary Table: Chapter 3 Command Review

| Command | Description |
|---|---|
| arp *ip-address hardware-address* | Statically maps an IP *type* [*alias*] address to a hardware address. |
| debug ip packet | Displays information on IP packets received, generated, and forwarded. Information on fast-switched packets will not be displayed. |
| debug ipv6 packet | Displays information on IPv6 packets received, generated, and forwarded. |
| ip cef | Enables Cisco Express Forwarding for IPv4. |
| ip load-sharing {per-packet \| per-destination} | Configures the type of load-sharing on an outbound interface. |
| ip proxy-arp | Enables proxy ARP. |
| ip route *prefix mask {address \| interface [next-hop-address]} [distance]* [permanent] [name name] [tag tag-number] | Statically adds a route entry to the route table. |
| ip route-cache | Configures the type of switching cache an interface will use for IPv4. |
| ipv6 cef | Enables Cisco Express Forwarding for IPv6. IPv4 CEF must be enabled before IPv6 CEF can be enabled. |
| ipv6 unicast-routing | Enables IPv6 routing. IPv6 routing is not enabled by default. |
| ipv6 route prefix/prefix length {address \| interface [next-hop-address]} [distance] [permanent] [name name] [tag tag-number] | Statically adds an IPv6 route. |
| show cdp neighbor detail | Displays information about a neighbor router, including IOS version, and IPv4 and IPv6 interface configuration. |
| show ip cef | Displays CEF forwarding cache or a message indicating CEF is not enabled. |
| show ipv6 cef | |
| show ipv6 cef *{interface}* detail | Displays whether CEF is enabled on an interface, among other things. |
| show ipv6 interface *{interface}* | Displays interface and IPv6 specific interface information. |
| show ip route | Displays the IP route table. |
| show ipv6 route | Displays the IPv6 route table. |

# Review Questions

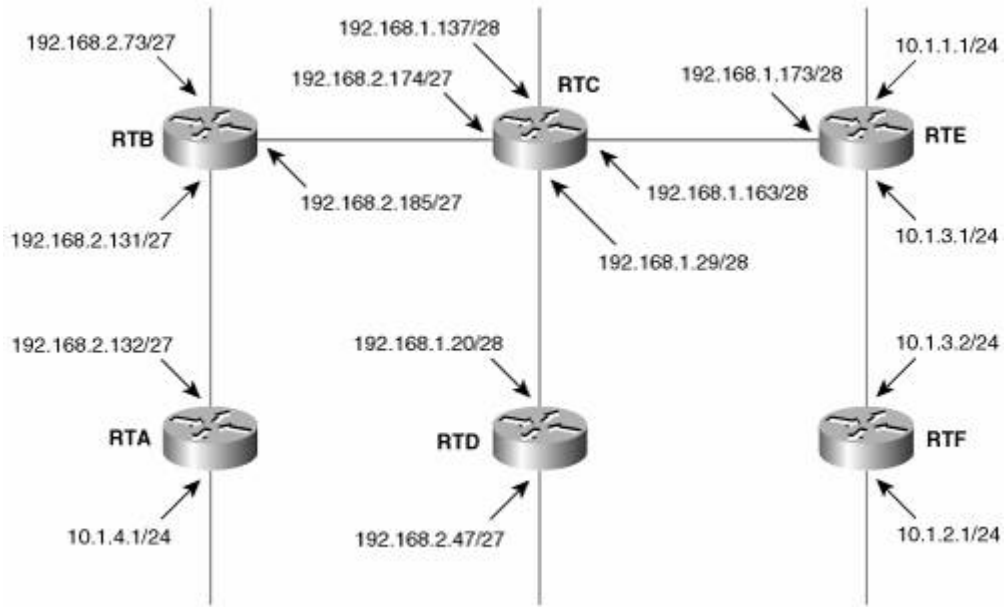
1. What information must be stored in the route table?
2. What does it mean when a route table says that an address is variably subnetted?
3. What are discontiguous subnets?
4. What IOS command is used to examine the IPv4 route table?
5. What IOS command is used to examine the IPv6 route table?
6. What are the two bracketed numbers associated with the nondirectly connected routes in the route table?
7. When static routes are configured to reference an exit interface instead of a next-hop address, how will the route table be different?
8. What is a summary route? In the context of static routing, how are summary routes useful?
9. What is an administrative distance?
10. What is a floating static route?
11. What is the difference between equal-cost and unequal-cost load sharing?
12. How does the switching mode at an interface affect load sharing?
13. What is a recursive table lookup?

# Configuration Exercises

1 Configure static routes for each router of the network shown in Figure 3-13. Write the routes so that every subnet of the Internet has an individual entry.

Figure 3-13. The network for Configuration Exercises 1 and 2.

[View full size image]



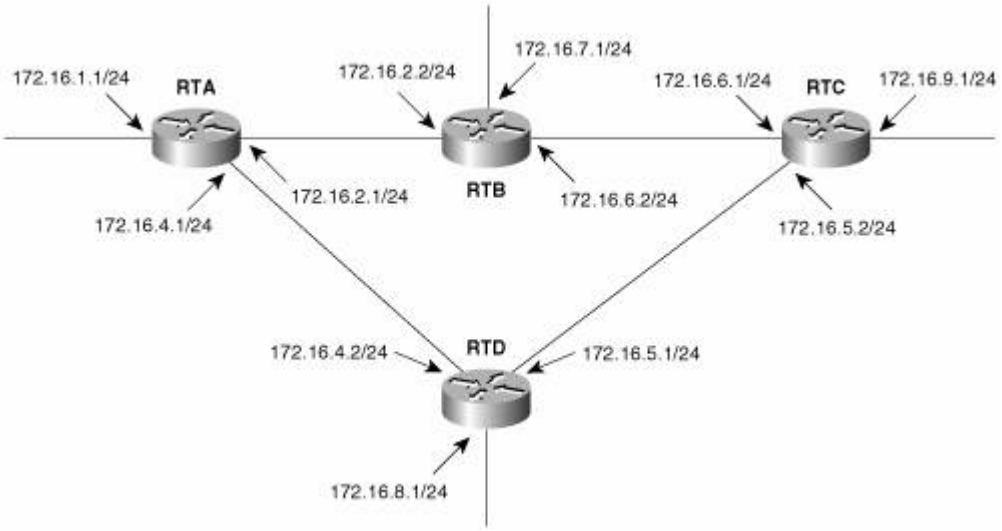

2 Rewrite the static routes created in Configuration Exercise 1 to use the minimum possible route entries.[8] (Hint: RTA will have only two static route entries.)

[8] If this exercise is being implemented in a lab, be sure to add the command **ip classless** to all six routers.

3 For the network shown in Figure 3-14, write static routes for each router. Assume that all links are identical media. Use load balancing on equal cost paths and floating static routes for maximum efficiency and redundancy.

Figure 3-14. The network for Configuration Exercise 3.

[View full size image]


172.16.1.1/24
RTA
172.16.2.2/24
172.16.7.1/24
172.16.6.1/24
RTC
172.16.9.1/24
172.16.2.1/24
RTB
172.16.6.2/24
172.16.4.1/24
172.16.5.2/24
172.16.4.2/24
RTD
172.16.5.1/24
172.16.8.1/24

.

PREV NEXT

# Troubleshooting Exercises

1 In the network of Figure 3-2 and the associated configurations, the route entries for Piglet a changed from

```
Piglet(config)# ip route 192.168.1.0 255.255.255.0 192.168.1.193
Piglet(config)# ip route 10.4.0.0 255.255.0.0 192.168.1.193
```

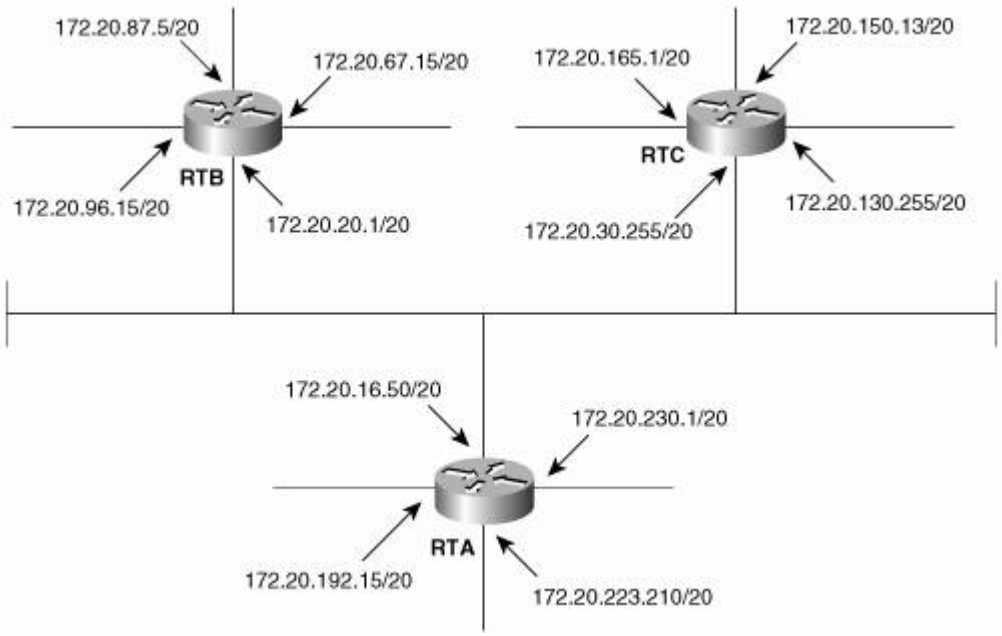
to

```
Piglet(config)# ip route 192.168.1.0 255.255.255.224 192.168.1.193
Piglet(config)# ip route 10.0.0.0 255.255.0.0 192.168.1.193
```

What is the result?

2 Example 3-59 shows the static route configurations for the routers in Figure 3-15.

Figure 3-15. The network for Troubleshooting Exercise 2.




Example 3-59. Static route configurations for routers in Figure 3-15

```
!RTA
ip route 172.20.96.0 255.255.240.0 172.20.20.1
ip route 172.20.82.0 255.255.240.0 172.20.20.1
ip route 172.20.64.0 255.255.240.0 172.20.20.1
ip route 172.20.160.0 255.255.240.0 172.20.30.255
ip route 172.20.144.0 255.255.240.0 172.20.30.255
```

```
ip route 172.20.128.0 255.255.240.0 172.20.30.255
```

---

```
!RTB
ip route 172.20.192.0 255.255.240.0 172.20.16.50
ip route 172.20.224.0 255.255.240.0 172.20.16.50
ip route 172.20.128.0 255.255.240.0 172.20.16.50
ip route 172.20.160.0 255.255.240.0 172.20.30.255
ip route 172.20.144.0 255.255.240.0 172.20.30.255
ip route 172.20.128.0 255.255.240.0 172.20.30.255
```

---

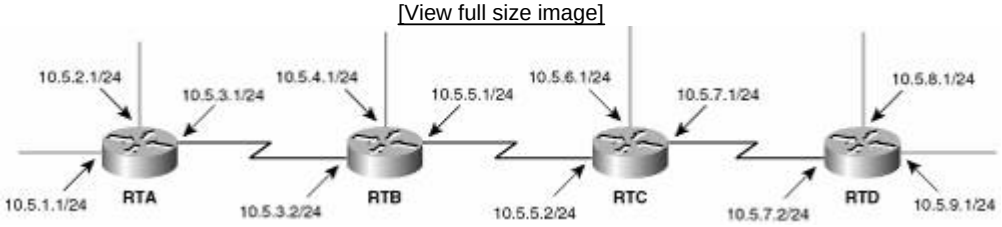
```
!RTC
ip route 172.20.192.0 255.255.240.0 172.20.16.50
ip route 172.20.208.0 255.255.255.0 172.20.16.50
ip route 172.20.224.0 255.255.240.0 172.20.16.50
ip route 172.20.96.0 255.255.240.0 172.20.20.1
ip route 172.20.82.0 255.255.240.0 172.20.20.1
ip route 172.20.64.0 255.255.240.0 172.20.20.1
```

Users are complaining of several connectivity problems in this internet. Find the mistakes in static route configurations.

3 Figure 3-16 shows another network in which users are complaining of connectivity problems Example 3-60 through Example 3-63 show the route tables of the four routers.

Figure 3-16. The network for Troubleshooting Exercise 3.

[View full size image]




Find the mistakes in the static route configurations.

Example 3-60. The route table of RTA in Figure 3-16.

```
RTA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate defau
       U  per-user static route
Gateway of last resort is not set
     10.0.0.0/8 is subnetted, 9 subnets
S      10.5.9.0 [1/0] via 10.5.3.2
S      10.5.8.0 [1/0] via 10.5.3.2
S      10.5.7.0 [1/0] via 10.5.3.2
S      10.5.6.0 [1/0] via 10.5.3.2
S      10.5.5.0 [1/0] via 10.5.3.2
S      10.5.4.0 [1/0] via 10.5.3.2
C      10.5.3.0 is directly connected, Serial0
C      10.5.2.0 is directly connected, TokenRing1
```

```
C      10.5.1.0 is directly connected, TokenRing0
RTA#
```

Example 3-61. The route table of RTB in Figure 3-16.

```
RTB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate defau
       U  per-user static route
Gateway of last resort is not set
     10.0.0.0/8 is subnetted, 9 subnets
S      10.5.9.0 [1/0] via 10.5.5.2
S      10.5.8.0 [1/0] via 10.5.5.2
S      10.5.7.0 [1/0] via 10.5.5.2
S      10.5.6.0 [1/0] via 10.5.5.2
C      10.5.5.0 is directly connected, Serial1
C      10.5.4.0 is directly connected, TokenRing0
C      10.5.3.0 is directly connected, Serial0
S      10.5.2.0 [1/0] via 10.5.3.1
S      10.5.1.0 [1/0] via 10.5.3.1
RTB#
```

Example 3-62. The route table of RTC in Figure 3-16.

```
RTC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate defau
       U  per-user static route, o - ODR
Gateway of last resort is not set
     10.0.0.0/24 is subnetted, 8 subnets
S      10.5.9.0 [1/0] via 10.5.7.2
S      10.5.8.0 [1/0] via 10.5.5.1
C      10.5.7.0 is directly connected, Serial1
C      10.5.6.0 is directly connected, Ethernet0
S      10.1.1.0 [1/0] via 10.5.5.1
C      10.5.5.0 is directly connected, Serial0
S      10.5.3.0 [1/0] via 10.5.5.1
S      10.5.2.0 [1/0] via 10.5.5.1
RTC#
```

Example 3-63. The route table of RTD in Figure 3-16.

```
RTD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate defau
       U  per-user static route, o - ODR
Gateway of last resort is not set
     10.0.0.0/24 is subnetted, 9 subnets
```

```
C       10.5.9.0 is directly connected, Ethernet1
C       10.5.8.0 is directly connected, Ethernet0
C       10.5.7.0 is directly connected, Serial0
S       10.5.6.0 [1/0] via 10.5.7.1
S       10.5.5.0 [1/0] via 10.5.7.1
S       10.4.5.0 [1/0] via 10.5.7.1
S       10.5.3.0 [1/0] via 10.5.7.1
S       10.5.2.0 [1/0] via 10.5.7.1
S       10.5.1.0 [1/0] via 10.5.7.1
RTD#
```

PREV NEXT

# Chapter 4. Dynamic Routing Protocols

This chapter covers the following subjects:


- Routing Protocol Basics
- Distance Vector Routing Protocols
- Link State Routing Protocols
- Interior and Exterior Gateway Protocols
- Static or Dynamic Routing?


The previous chapter explains what a router needs to recognize to correctly switch packets to their respective destinations, and how that information is put into the route table manually. This chapter shows how routers can discover this information automatically and share that information with other routers via dynamic routing protocols. A *routing protocol* is the language a router speaks with other routers to share information about the reachability and status of networks.

Dynamic routing protocols not only perform these path-determination and route-table-update functions, but also determine the next-best path if the best path to a destination becomes unusable. The capability to compensate for topology changes is the most important advantage dynamic routing offers over static routing.

Obviously, for communication to occur, the communicators must speak the same language. Since the advent of IP routing, there have been eight major IP routing protocols from which to choose;[1] if one router speaks RIP and another speaks OSPF, they cannot share routing information because they are not speaking the same language. Subsequent chapters examine all the IP routing protocols in current use, and even consider how to make a router "bilingual," but first it is necessary to explore some characteristics and issues common to all routing protocolsIP or otherwise.

[1] Of these eight protocols, BGP has obsoleted EGP, the Cisco Systems EIGRP obsoleted its IGRP, and RIPv2 is quickly replacing RIPv1.

# Routing Protocol Basics

All dynamic routing protocols are built around an algorithm. Generally, an *algorithm* is a step-by-step procedure for solving a problem. A routing algorithm must, at a minimum, specify the following:

- A procedure for passing reachability information about networks to other routers
- A procedure for receiving reachability information from other routers
- A procedure for determining optimal routes based on the reachability information it has and for recording this information in a route table
- A procedure for reacting to, compensating for, and advertising topology changes in a network

A few issues common to any routing protocol are path determination, metrics, convergence, and load balancing.

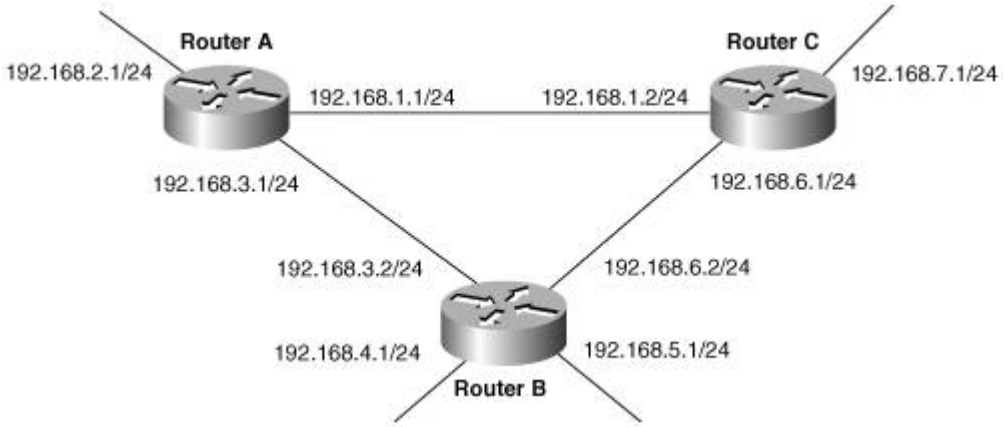
## Path Determination

All subnets within a network must be connected to a router, and wherever a router has an interface on a network, that interface must have an address on the network.[2] This address is the originating point for reachability information.

[2] Many point-to-point links are configured as "unnumbered" linksthat is, there is no address assigned to the connected point-to-point interfaceso as to conserve addresses. But unnumbered links do not violate the rule that every interface must have an address; they use another address on the router, usually the loopback address, as a proxy address.

Figure 4-1 shows a simple three-router network. Router A recognizes networks 192.168.1.0, 192.168.2.0, and 192.168.3.0 because it has interfaces on those networks with corresponding addresses and appropriate address masks. Likewise, Router B recognizes 192.168.3.0, 192.168.4.0, 192.168.5.0, and 192.186.6.0; Router C recognizes 192.168.6.0, 192.168.7.0, and 198.168.1.0. Each interface implements the data link and physical protocols of the network to which it is attached, so the router also recognizes the state of the network (up or down).

Figure 4-1. Each router knows about its directly connected networks from its assigned addresses and masks.




At first glance, the information-sharing procedure seems simple. Look at Router A:

1. Router A examines its IP addresses and associated masks and deduces that it is attached to networks 192.168.1.0, 192.186.2.0, and 192.168.3.0.
2. Router A enters these networks into its route table, along with some sort of flag indicating that the networks are directly connected.
3. Router A places the information into a packet: "My directly connected networks are 192.168.1.0, 192.186.2.0, and 192.168.3.0."
4. Router A transmits copies of these route information packets, or *routing updates*, to Routers B and C.

Routers B and C, having performed the same steps, have sent updates with their directly connected networks to A. Router A enters the received information into its route table, along with the source address of the router that sent the update packet. Router A now recognizes all the networks and the addresses of the routers to which they are attached.

This procedure does seem quite simple. So why are routing protocols so much more complicated than this? Look again at Figure 4-1:

- What should Router A do with the updates from B and C after it has recorded the information in the route table? Should it, for instance, pass B's routing information packet to C and pass C's packet to B?
- If Router A does not forward the updates, information sharing might not be complete. For instance, if the link between B and C does not exist, those two routers would not recognize each other's networks. Router A must forward the update information, but this step opens a new set of problems.
- If Router A hears about network 192.168.4.0 from both Router B and Router C, which router should be used to reach that network? Are they both valid? Which one is the best path?
- What mechanism will be used to ensure that all routers receive all routing information while preventing update packets from circulating endlessly through the network?
- The routers share certain directly connected networks (192.168.1.0, 192.168.3.0, and 192.168.6.0). Should the routers still advertise these networks?

These questions are almost as simplistic as the preceding preliminary explanation of routing protocols, but they should give you an indication for some of the issues that contribute to the

complexity of the protocols. Each routing protocol addresses these questions one way or another, which will become clear in following sections and chapters.

## Metrics

When there are multiple routes to the same destination, a router must have a mechanism for calculating the best path. A *metric* is a variable assigned to routes as a means of ranking them from best to worst or from most preferred to least preferred. Consider the following example of why metrics are needed.

Assuming that information sharing has properly occurred in the network of Figure 4-1, Router A might have a route table that looks like Table 4-1.

Table 4-1. Rudimentary route table for Router A of Figure 4-1.

| Network | Next-Hop Router |
|---|---|
| 192.168.1.0 | Directly connected |
| 192.168.2.0 | Directly connected |
| 192.168.3.0 | Directly connected |
| 192.168.4.0 | B, C |
| 192.168.5.0 | B, C |
| 192.168.6.0 | B, C |
| 192.168.7.0 | B, C |

This route table says that the first three networks are directly connected and that no routing is needed from Router A to reach them, which is correct. The last four networks, according to this table, can be reached via Router B or Router C. This information is also correct. But if network 192.168.7.0 can be reached via either Router B or Router C, which path is the preferable path? Metrics are needed to rank the alternatives.

Different routing protocols use different metrics. For example, RIP defines the "best" route as the one with the least number of router hops; EIGRP defines the "best" route based on a combination of the lowest bandwidth along the route and the total delay of the route. The following sections provide basic definitions of these and other commonly used metrics. Further complexitiessuch as how some routing protocols such as EIGRP use multiple parameters to compute a metric and deal with routes that have identical metric valuesare covered later, in the protocol-specific chapters of this book.

### Hop Count

A hop-count metric simply counts router hops. For instance, from Router A, it is one hop to network 192.168.5.0 if packets are sent out interface 192.168.3.1 (through Router B) and two

hops if packets are sent out 192.168.1.1 (through Routers C and B). Assuming hop count is the only metric being applied, the best route is the one with the fewest hops, in this case, A-B.

But is the A-B link really the best path? If the A-B link is a DS-0 link and the A-C and C-B links are T-1 links, the two-hop route might actually be best, because bandwidth plays a role in how efficiently traffic travels through the network.

## Bandwidth

A bandwidth metric would choose a higher-bandwidth path over a lower-bandwidth link. However, bandwidth by itself still might not be a good metric. What if one or both of the T1 links are heavily loaded with other traffic and the 56K link is lightly loaded? Or what if the higher-bandwidth link also has a higher delay?

## Load

This metric reflects the amount of traffic utilizing the links along the path. The best path is the one with the lowest load.

Unlike hop count and bandwidth, the load on a route changes, and, therefore, the metric will change. Care must be taken here. If the metric changes too frequently, route *flapping*the frequent change of preferred routesmight occur. Route flaps can have adverse effects on the router's CPU, the bandwidth of the data links, and the overall stability of the network.

## Delay

*Delay* is a measure of the time a packet takes to traverse a route. A routing protocol using delay as a metric would choose the path with the least delay as the best path. There might be many ways to measure delay. Delay might take into account not only the delay of the links along the route, but also such factors as router latency and queuing delay. On the other hand, the delay of a route might not be measured at all; it might be a sum of static quantities defined for each interface along the path. Each individual delay quantity would be an estimate based on the type of link to which the interface is connected.

## Reliability

*Reliability* measures the likelihood that the link will fail in some way and can be either variable or fixed. Examples of variable-reliability metrics are the number of times a link has failed, or the number of errors it has received within a certain time period. Fixed-reliability metrics are based on known qualities of a link as determined by the network administrator. The path with highest reliability would be selected as best.

## Cost

This metric is configured by a network administrator to reflect more- or less-preferred routes. Cost might be defined by any policy or link characteristic or might reflect the arbitrary judgment of the network administrator. Therefore, "cost" is a term of convenience describing a dimensionless metric.

The term *cost* is often used as a generic term when speaking of route choices. For example, "RIP chooses the lowest-cost path based on hop count." Another generic term is *shortest*, as in "RIP chooses the shortest path based on hop count." When used in this context, either *lowest-cost* (or *highest-cost*) and *shortest* (or *longest*) merely refer to a routing protocol's view of paths based on its specific metrics.

## Convergence

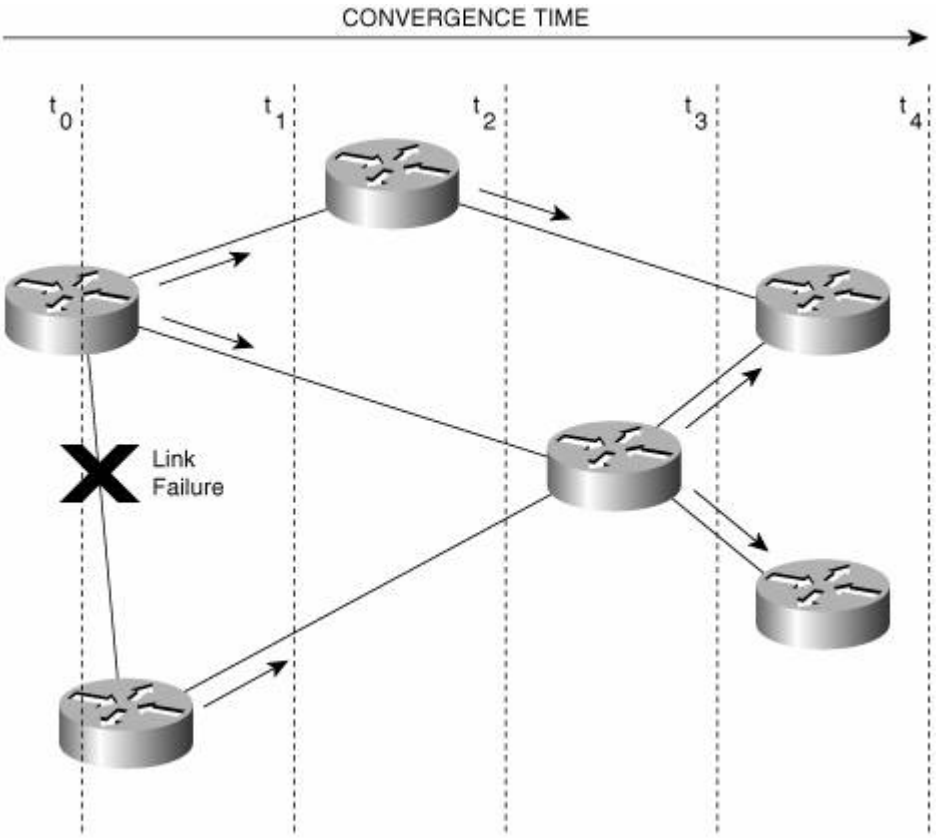
A dynamic routing protocol must include a set of procedures for a router to inform other routers about its directly connected networks, to receive and process the same information from other routers, and to pass along the information it receives from other routers. Further, a routing protocol must define a metric by which best paths might be determined.

A further criterion for routing protocols is that the reachability information in the route tables of all routers in the network must be consistent. If Router A in Figure 4-1 determines that the best path to network 192.168.5.0 is via Router C and if Router C determines that the best path to the same network is through Router A, Router A will send packets destined for 192.168.5.0 to C, C will send them back to A, A will again send them to C, and so on. This continuous circling of traffic between two or more destinations is referred to as a *routing loop*.

The process of bringing all route tables to a state of consistency is called *convergence*. The time it takes to share information across a network and for all routers to calculate best paths is the *convergence time*.

Figure 4-2 shows a network that was converged, but now a topology change has occurred. The link between the two left-most routers has failed; both routers, being directly connected, know about the failure from the data link protocol and proceed to inform their neighbors of the unavailable link. The neighbors update their route tables accordingly and inform their neighbors, and the process continues until all routers know about the change.

Figure 4-2. Reconvergence after a topology change takes time. While the network is in an unconverged state, routers are susceptible to bad routing information.



Notice that at time $t_2$ the three left-most routers recognize the topology change but the three right-most routers have not yet received that information. Those three have old information and will continue to switch packets accordingly. It is during this intermediate time, when the network is in an unconverged state, that routing errors might occur. Therefore, convergence time is an important factor in any routing protocol. The faster a network can reconverge after a topology change, the better.

## Load Balancing

Recall from Chapter 3, "Static Routing," that load balancing is the practice of distributing traffic among multiple paths to the same destination, so as to use bandwidth efficiently. As an example of the usefulness of load balancing, consider Figure 4-1 again. All the networks in Figure 4-1 are reachable from two paths. If a device on 192.168.2.0 sends a stream of packets to a device on

PREV NEXT

# Distance Vector Routing Protocols

Most routing protocols fall into one of two classes: *distance vector* or *link state*. The basics of distance vector routing protocols are examined here; the next section covers link state routing protocols. Most distance vector algorithms are based on the work done of R. E. Bellman,[3] L. R. Ford, and D. R. Fulkerson,[4] and for this reason occasionally are referred to as *Bellman-Ford or Ford-Fulkerson algorithms*. A notable exception is EIGRP, which is based on an algorithm developed by J. J. Garcia Luna Aceves.

[3] R. E. Bellman. *Dynamic Programming.* Princeton, New Jersey: Princeton University Press; 1957.

[4] L. R. Ford Jr. and D. R. Fulkerson. *Flows in Networks.* Princeton, New Jersey: Princeton University Press; 1962.

The name distance vector is derived from the fact that routes are advertised as vectors of (distance, direction), where distance is defined in terms of a metric and direction is defined in terms of the next-hop router. For example, "Destination A is a distance of five hops away, in the direction of next-hop Router X." As that statement implies, each router learns routes from its neighboring routers' perspectives and then advertises the routes from its own perspective. Because each router depends on its neighbors for information, which the neighbors in turn might have learned from their neighbors, and so on, distance vector routing is sometimes facetiously referred to as "routing by rumor."

Distance vector routing protocols include the following:

- Routing Information Protocol (RIP) for IP
- Xerox Networking System's XNS RIP
- Novell's IPX RIP
- The Cisco Systems Internet Gateway Routing Protocol (IGRP) and Enhanced Internet Gateway Routing Protocol (EIGRP)
- DEC's DNA Phase IV
- AppleTalk's Routing Table Maintenance Protocol (RTMP)

## Common Characteristics

A typical distance vector routing protocol uses a routing algorithm in which routers periodically send routing updates to all neighbors by broadcasting their entire route tables.[5]

[5] A notable exception to this convention is the Cisco Enhanced IGRP. EIGRP is a distance vector protocol, but its updates are not periodic, are not broadcasted, and do not contain the full route table. EIGRP is covered in Chapter 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)."

The preceding statement contains a lot of information. Following sections consider it in more detail.

### Periodic Updates

*Periodic updates* means that at the end of a certain time period, updates will be transmitted. This period typically ranges from 10 seconds for AppleTalk's RTMP to 90 seconds for the Cisco IGRP. At issue here is the fact that if updates are sent too frequently, congestion and router CPU

overloading might occur; if updates are sent too infrequently, convergence time might be unacceptably high.

### Neighbors

In the context of routers, *neighbors* means routers sharing a common data link or some higher-level logical adjacency. A distance vector routing protocol sends its updates to neighboring routers[6] and depends on them to pass the update information along to their neighbors. For this reason, distance vector routing is said to use hop-by-hop updates.

[6] This statement is not entirely true. Hosts also can listen to routing updates in some implementations; but all that is important for this discussion is how routers work.

### Broadcast Updates

When a router first becomes active on a network, how does it find other routers and how does it announce its own presence? Several methods are available. The simplest is to send the updates to the broadcast address (in the case of IP, 255.255.255.255). Neighboring routers speaking the same routing protocol will hear the broadcasts and take appropriate action. Hosts and other devices uninterested in the routing updates will simply drop the packets.

### Full Routing Table Updates

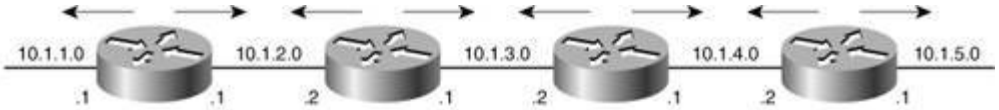
Most distance vector routing protocols take the very simple approach of telling their neighbors everything they know by broadcasting their entire route table, with some exceptions that are covered in following sections. Neighbors receiving these updates glean the information they need and discard everything else.

## Routing by Rumor

Figure 4-3 shows a distance vector algorithm in action. In this example, the metric is hop count. At time $t_0$, Routers A through D have just become active. Looking at the route tables across the top row, at $t_0$ the only information any of the four routers has is its own directly connected networks. The tables identify these networks and indicate that they are directly connected by having no next-hop router and by having a hop count of 0. Each of the four routers will broadcast this information on all links.

Figure 4-3. Distance vector protocols converge hop-by-hop.

[View full size image]




**Router A**

| | NET | VIA | HOPS |
|---|---|---|---|
| $t_0$ | 10.1.1.0 | -- | 0 |
| | 10.1.2.0 | -- | 0 |
| $t_1$ | 10.1.1.0 | -- | 0 |
| | 10.1.2.0 | -- | 0 |
| | 10.1.3.0 | 10.1.2.2 | 1 |
| $t_2$ | 10.1.1.0 | -- | 0 |
| | 10.1.2.0 | -- | 0 |
| | 10.1.3.0 | 10.1.2.2 | 1 |
| | 10.1.4.0 | 10.1.2.2 | 2 |
| $t_3$ | 10.1.1.0 | -- | 0 |
| | 10.1.2.0 | -- | 0 |
| | 10.1.3.0 | 10.1.2.2 | 1 |
| | 10.1.4.0 | 10.1.2.2 | 2 |
| | 10.1.5.0 | 10.1.2.2 | 3 |

**Router B**

| | NET | VIA | HOPS |
|---|---|---|---|
| $t_0$ | 10.1.2.0 | -- | 0 |
| | 10.1.3.0 | -- | 0 |
| $t_1$ | 10.1.2.0 | -- | 0 |
| | 10.1.3.0 | -- | 0 |
| | 10.1.1.0 | 10.1.2.1 | 1 |
| | 10.1.4.0 | 10.1.3.2 | 1 |
| $t_2$ | 10.1.2.0 | -- | 0 |
| | 10.1.3.0 | -- | 0 |
| | 10.1.1.0 | 10.1.2.1 | 1 |
| | 10.1.4.0 | 10.1.3.2 | 1 |
| | 10.1.5.0 | 10.1.3.2 | 2 |
| $t_3$ | 10.1.2.0 | -- | 0 |
| | 10.1.3.0 | -- | 0 |
| | 10.1.1.0 | 10.1.2.1 | 1 |
| | 10.1.4.0 | 10.1.3.2 | 1 |
| | 10.1.5.0 | 10.1.3.2 | 2 |

**Router C**

| | NET | VIA | HOPS |
|---|---|---|---|
| $t_0$ | 10.1.3.0 | -- | 0 |
| | 10.1.4.0 | -- | 0 |
| $t_1$ | 10.1.3.0 | -- | 0 |
| | 10.1.4.0 | -- | 0 |
| | 10.1.2.0 | 10.1.3.1 | 1 |
| | 10.1.5.0 | 10.1.4.2 | 1 |
| $t_2$ | 10.1.3.0 | -- | 0 |
| | 10.1.4.0 | -- | 0 |
| | 10.1.2.0 | 10.1.3.1 | 1 |
| | 10.1.5.0 | 10.1.4.2 | 1 |
| | 10.1.1.0 | 10.1.3.1 | 2 |
| $t_3$ | 10.1.3.0 | -- | 0 |
| | 10.1.4.0 | -- | 0 |
| | 10.1.2.0 | 10.1.3.1 | 1 |
| | 10.1.5.0 | 10.1.4.2 | 1 |
| | 10.1.1.0 | 10.1.3.1 | 2 |

**Router D**

| | NET | VIA | HOPS |
|---|---|---|---|
| $t_0$ | 10.1.4.0 | -- | 0 |
| | 10.1.5.0 | -- | 0 |
| $t_1$ | 10.1.4.0 | -- | 0 |
| | 10.1.5.0 | -- | 0 |
| | 10.1.3.0 | 10.1.4.1 | 1 |
| $t_2$ | 10.1.4.0 | -- | 0 |
| | 10.1.5.0 | -- | 0 |
| | 10.1.3.0 | 10.1.4.1 | 1 |
| | 10.1.2.0 | 10.1.4.1 | 2 |
| $t_3$ | 10.1.4.0 | -- | 0 |
| | 10.1.5.0 | -- | 0 |
| | 10.1.3.0 | 10.1.4.1 | 1 |
| | 10.1.2.0 | 10.1.4.1 | 2 |
| | 10.1.1.0 | 10.1.4.1 | 3 |

At time $t_1$, the first updates have been received and processed by the routers. Look at Router A's table at $t_1$. Router B's update to Router A said that Router B can reach networks 10.1.2.0 and 10.1.3.0, both zero hops away. If the networks are zero hops from B, they must be one hop from A. Router A incremented the hop count by one and then examined its route table. It already recognized 10.1.2.0, and the hop count (zero) was less than the hop count B advertised, (one), so A disregarded that information.

Network 10.1.3.0 was new information, however, so A entered this in the route table. The source address of the update packet was Router B's interface (10.1.2.2) so that information is entered along with the calculated hop count.

Notice that the other routers performed similar operations at the same time $t_1$. Router C, for instance, disregarded the information about 10.1.3.0 from B and 10.1.4.0 from C but entered information about 10.1.2.0, reachable via B's interface address 10.1.3.1, and 10.1.5.0, reachable via C's interface 10.1.4.2. Both networks were calculated as one hop away.

At time $t_2$, the update period has again expired and another set of updates has been broadcast. Router B sent its latest table; Router A again incremented B's advertised hop counts by one and compared. The information about 10.1.2.0 is again discarded for the same reason as before. 10.1.3.0 is already known, and the hop count hasn't changed, so that information is also discarded. 10.1.4.0 is new information and is entered into the route table.

The network is converged at time $t_3$. Every router recognizes every network, the address of the next-hop router for every network, and the distance in hops to every network.

It is time for an analogy. You are wandering in the Sangre de Cristo mountains of northern New Mexicoa wonderful place to wander if you aren't lost. But you are lost. You come upon a fork in the trail and a sign pointing west, reading "Taos, 15 miles." You have no choice but to trust the sign. You have no clue what the terrain is like over that 15 miles; you don't know whether there is a better route or even whether the sign is correct. Someone could have turned it around, in which case you will travel deeper into the forest instead of to safety!

Distance vector algorithms provide road signs to networks.[7] They provide the direction and the distance, but no details about what lies along the route. And like the sign at the fork in the trail, they are vulnerable to accidental or intentional misdirection. Following are some of the difficulties and refinements associated with distance vector algorithms.

[7] The road sign analogy is commonly used. You can find a good presentation in Radia Perlman's *Interconnections*, pp.

205210.

## Route Invalidation Timers

Now that the network in Figure 4-3 is fully converged, how will it handle reconvergence when some part of the topology changes? If network 10.1.5.0 goes down, the answer is simple enoughRouter D, in its next scheduled update, flags the network as unreachable and passes the information along.

But what if, instead of 10.1.5.0 going down, Router D fails? Routers A, B, and C still have entries in their route tables about 10.1.5.0; the information is no longer valid, but there's no router to inform them of this fact. They will unknowingly forward packets to an unreachable destinationa black hole has opened in the network.

This problem is handled by setting a route invalidation timer for each entry in the route table. For example, when Router C first hears about 10.1.5.0 and enters the information into its route table, C sets a timer for that route. At every regularly scheduled update from Router D, C discards the update's already-known information about 10.1.5.0 as described in "Routing by Rumor." But as C does so, it resets the timer on that route.

If Router D goes down, C will no longer hear updates about 10.1.5.0. The timer will expire, C will flag the route as unreachable and will pass the information along in the next update.

Typical periods for route timeouts range from three to six update periods. A router would not want to invalidate a route after a single update has been missed, because this event might be the result of a corrupted or lost packet or some sort of network delay. At the same time, if the period is too long, reconvergence will be excessively slow.

## Split Horizon

According to the distance vector algorithm as it has been described so far, at every update period each router broadcasts its entire route table to every neighbor. But is this really necessary? Every network known by Router A in Figure 4-3, with a hop count higher than zero, has been learned from Router B. Common sense suggests that for Router A to broadcast the networks it has learned from Router B back to Router B is a waste of resources. Obviously, B already "knows" about those networks.

A route pointing back to the router from which packets were received is called a reverse route. *Split horizon* is a technique for preventing reverse routes between two routers.

Besides not wasting resources, there is a more important reason for not sending reachability information back to the router from which the information was learned. The most important function of a dynamic routing protocol is to detect and compensate for topology changesif the best path to a network becomes unreachable, the protocol must look for a next-best path.

Look yet again at the converged network of Figure 4-3 and suppose that network 10.1.5.0 goes down. Router D will detect the failure, flag the network as unreachable, and pass the information along to Router C at the next update interval. However, before D's update timer triggers an update, something unexpected happens. C's update arrives, claiming that it can reach 10.1.5.0, one hop away! Remember the road sign analogy? Router D has no way of recognizing that C is not advertising a legitimate next-best path. It will increment the hop count and make an entry into its route table indicating that 10.1.5.0 is reachable via Router C's interface 10.1.4.1, just two hops away.

Now a packet with a destination address of 10.1.5.3 arrives at Router C, which consults its route table and forwards the packet to D. Router D consults its route table and forwards the packet to C, C forwards it back to D, *ad infinitum*. A routing loop has occurred.

Implementing split horizon prevents the possibility of such a routing loop. There are two

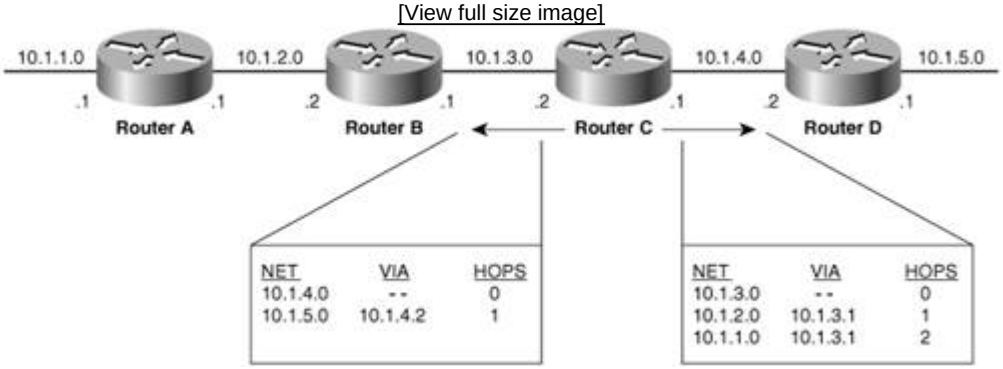
categories of split horizon: simple split horizon and split horizon with poisoned reverse.

The rule for simple split horizon is, when sending updates out a particular interface, do not include networks that were learned from updates received on that interface.

The routers in Figure 4-4 implement simple split horizon. Router C sends an update to Router D for networks 10.1.1.0, 10.1.2.0, and 10.1.3.0. Networks 10.1.4.0 and 10.1.5.0 are not included because they were learned from Router D. Likewise, updates to Router B include 10.1.4.0 and 10.1.5.0 with no mention of 10.1.1.0, 10.1.2.0, and 10.1.3.0.

Figure 4-4. Simple split horizon does not advertise routes back to the neighbors from whom the routes were learned.

[View full size image]



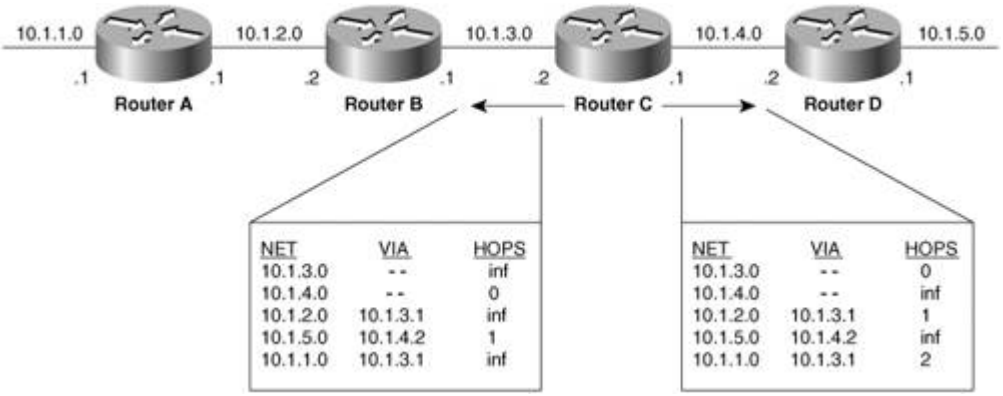

Simple split horizon works by suppressing information. Split horizon with poisoned reverse is a modification that provides more positive information.

The rule for split horizon with poisoned reverse is, when sending updates out a particular interface, designate any networks that were learned from updates received on that interface as unreachable.

In the scenario of Figure 4-4, Router C would in fact advertise 10.1.4.0 and 10.1.5.0 to Router D, but the network would be marked as unreachable. Figure 4-5 shows what the route tables from C to B and D would look like. Notice that a route is marked as unreachable by setting the metric to infinity; in other words, the network is an infinite distance away. Coverage of a routing protocol's concept of infinity continues in the next section.

Figure 4-5. Split horizon with poisoned reverse advertises reverse routes but with an unreachable (infinite) metric.

[View full size image]

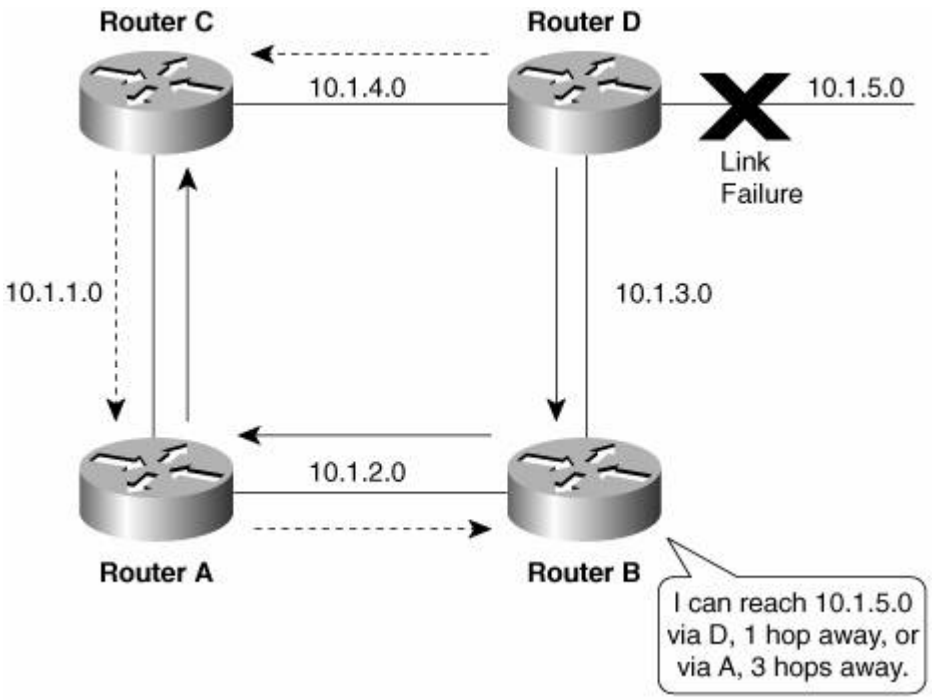



Split horizon with poisoned reverse is considered safer and stronger than simple split horizona sort of "bad news is better than no news at all" approach. For example, suppose that Router B in Figure 4-5 receives corrupted information, causing it to proceed as if subnet 10.1.1.0 is reachable via Router C. Simple split horizon would do nothing to correct this misperception, whereas a poisoned reverse update from Router C would immediately stop the potential loop. For this reason, most modern distance vector implementations use split horizon with poisoned reverse. The trade-off is that routing update packets are larger, which might exacerbate any congestion problems on a link.

## Counting to Infinity

Split horizon will break loops between neighbors, but it will not stop loops in a network such as the one in Figure 4-6. Again, 10.1.5.0 has failed. Router D sends the appropriate updates to its neighbors, Router C (the dashed arrows), and Router B (the solid arrows). Router B marks the route via D as unreachable, but Router A is advertising a next-best path to 10.1.5.0, which is three hops away. B posts that route in its route table.

Figure 4-6. Split horizon will not prevent routing loops here.



B now informs D that it has an alternative route to 10.1.5.0. D posts this information and updates C, saying that it has a four-hop route to the network. C tells A that 10.1.5.0 is five hops away. A tells B that the network is now six hops away.

"Ah," Router B thinks, "Router A's path to 10.1.5.0 has increased in length. Nonetheless, it's the only route I've got, so I'll use it!"

B changes the hop count to seven, updates D, and around it goes again. This situation is the *counting-to-infinity* problem because the hop count to 10.1.5.0 will continue to increase to infinity. All routers are implementing split horizon, but it doesn't help.

The way to alleviate the effects of counting to infinity is to define infinity. Most distance vector protocols define infinity to be 16 hops. As the updates continue to loop among the routers in Figure 4-6, the hop count to 10.1.5.0 in all routers will eventually increment to 16. At that time, the network will be considered unreachable.

This method is also how routers advertise a network as unreachable. Whether it is a poisoned reverse route, a network that has failed, or a network beyond the maximum network diameter of 15 hops, a router will recognize any 16-hop route as unreachable.

Setting a maximum hop count of 15 helps solve the counting-to-infinity problem, but convergence will still be very slow. Given an update period of 30 seconds, a network could take up to 7.5 minutes to reconverge and is susceptible to routing errors during this time. Triggered updates can be used to reduce this convergence time.

## Triggered Updates

*Triggered updates*, also known as *flash updates*, are very simple: If a metric changes for better or for worse, a router will immediately send out an update without waiting for its update timer to expire. Reconvergence will occur far more quickly than if every router had to wait for regularly scheduled updates, and the problem of counting to infinity is greatly reduced, although not completely eliminated. Regular updates might still occur along with triggered updates. Thus a router might receive bad information about a route from a not-yet-reconverged router after having received correct information from a triggered update. Such a situation shows that confusion and routing errors might still occur while a network is reconverging, but triggered updates will help to iron things out more quickly.

A further refinement is to include in the update only the networks that actually triggered it, rather than the entire route table. This technique reduces the processing time and the impact on network bandwidth.

## Holddown Timers

Triggered updates add responsiveness to a reconverging network. *Holddown timers* introduce a certain amount of skepticism to reduce the acceptance of bad routing information.

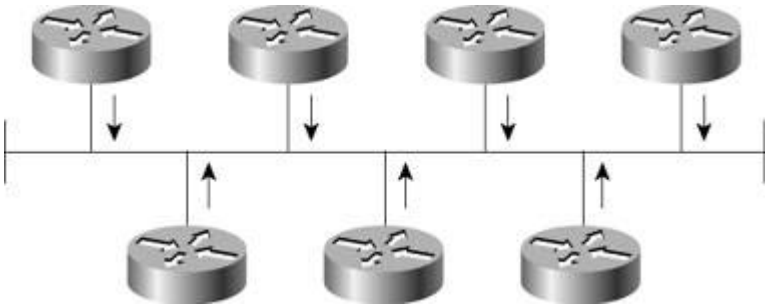
If the distance to a destination increases (for example, the hop count increases from two to four), the router sets a holddown timer for that route. Until the timer expires, the router will not accept any new updates for the route.

Obviously, a trade-off is involved here. The likelihood of bad routing information getting into a table is reduced but at the expense of the reconvergence time. Like other timers, holddown timers must be set with care. If the holddown period is too short, it will be ineffective, and if it is too long, normal routing will be adversely affected.

## Asynchronous Updates

Figure 4-7 shows a group of routers connected to an Ethernet backbone. The routers should not broadcast their updates at the same time; if they do, the update packets will collide. Yet this situation is exactly what can happen when several routers share a broadcast network. System delays related to the processing of updates in the routers tend to cause the update timers to become synchronized. As a few routers become synchronized, collisions will begin to occur, further contributing to system delays, and eventually all routers sharing the broadcast network might become synchronized.

Figure 4-7. If update timers become synchronized, collisions might occur.



Asynchronous updates might be maintained by one of two methods:

- Each router's update timer is independent of the routing process and is, therefore, not affected by processing loads on the router.
- A small random time, or *timing jitter*, is added to each update period as an offset.

If routers implement the method of rigid, system-independent timers, all routers sharing a broadcast network must be brought online in a random fashion. Rebooting the entire group of routers simultaneouslysuch as might happen during a widespread power outage, for examplecould result in all the timers attempting to update at the same time.

Adding randomness to the update period is effective if the variable is large enough in proportion to the number of routers sharing the broadcast network. Sally Floyd and Van Jacobson[8] have calculated that a too-small randomization will be overcome by a large enough network of routers, and that to be effective the update timer should range up to 50 percent of the median update period.

[8] S. Floyd and V. Jacobson. "The Synchronisation of Periodic Routing Messages." ACM Sigcomm '93 Symposium, September 1993.

PREV NEXT

# Link State Routing Protocols

The information available to a distance vector router has been compared to the information available from a road sign. Link state routing protocols are like a road map. A link state router cannot be fooled as easily into making bad routing decisions, because it has a complete picture of the network. The reason is that unlike the routing-by-rumor approach of distance vector, link state routers have firsthand information from all their peer[9] routers. Each router originates information about itself, its directly connected links, the state of those links (hence the name), and any directly connected neighbors. This information is passed around from router to router, each router making a copy of it, but never changing it. The ultimate objective is that every router has identical information about the network, and each router will independently calculate its own best paths.

[9] That is, all routers speaking the same routing protocol.

Link state protocols, sometimes called *shortest path* first or *distributed database* protocols, are built around a well-known algorithm from graph theory, E. W. Dijkstra's shortest path algorithm. Examples of link state routing protocols are

- Open Shortest Path First (OSPF) for IP
- The ISO's Intermediate System-to-Intermediate System (IS-IS) for CLNS and IP
- DEC's DNA Phase V
- Novell's NetWare Link Services Protocol (NLSP)

Although link-state protocols are rightly considered more complex than distance vector protocols, the basic functionality is not complex at all:

1. Each router establishes a relationshipan adjacencywith each of its neighbors.
2. Each router sends a data unit we will call *link state advertisements* (LSAs) to each neighbor.[10] The LSA lists each of the router's links, and for each link, it identifies the link, the state of the link, the metric cost of the router's interface to the link, and any neighbors that might be connected to the link. Each neighbor receiving an advertisement in turn forwards (*floods*) the advertisement to its own neighbors.

   [10] LSA is an OSPF term. The corresponding data unit created by IS-IS is called link state PDU (LSP). But for this general discussion, "LSA" fits our needs.

3. Each router stores a copy of all the LSAs it has seen in a database. If all works well, the databases in all routers should be identical.
4. The completed *topological database*, also called the *link state database*, is used by the Dijkstra algorithm to compute a graph of the network indicating the shortest path to each router. The link state protocol can then consult the link state database to find what subnets are attached to each router, and enter this information into the routing table.

## Neighbors

Neighbor discovery is the first step in getting a link state environment up and running. In keeping with the friendly neighbor terminology, a Hello protocol is used for this step. The

protocol will define a Hello packet format and a procedure for exchanging the packets and processing the information the packets contain.

At a minimum, the Hello packet will contain a *router ID* (RID) and the address of the network on which the packet is being sent. The router ID is something by which the router originating the packet can be uniquely distinguished from all other routers; for instance, an IP address from one of the router's interfaces. Other fields of the packet might carry a subnet mask, Hello intervals, a specified maximum period the router will wait to hear a Hello before declaring the neighbor "dead," a descriptor of the circuit type, and flags to help in bringing up adjacencies.

When two routers have discovered each other as neighbors, they go through a process of synchronizing their databases in which they exchange and verify database information until their databases are identical. The details of database synchronization are described in Chapters 8, "OSPFv2," and 10, "Integrated IS-IS." To perform this database synchronization, the neighbors must be *adjacent*that is, they must agree on certain protocol-specific parameters such as timers and support of optional capabilities. By using Hello packets to build adjacencies, link state protocols can exchange information in a controlled fashion. Contrast this approach with distance vector, which simply broadcasts updates out any interface configured for that routing protocol.

Beyond building adjacencies, Hello packets serve as keepalives to monitor the adjacency. If Hellos are not heard from an adjacent neighbor within a certain established time, the neighbor is considered unreachable and the adjacency is broken. A typical interval for the exchange of Hello packets is ten seconds, and a typical dead period is four times that.

## Link State Flooding

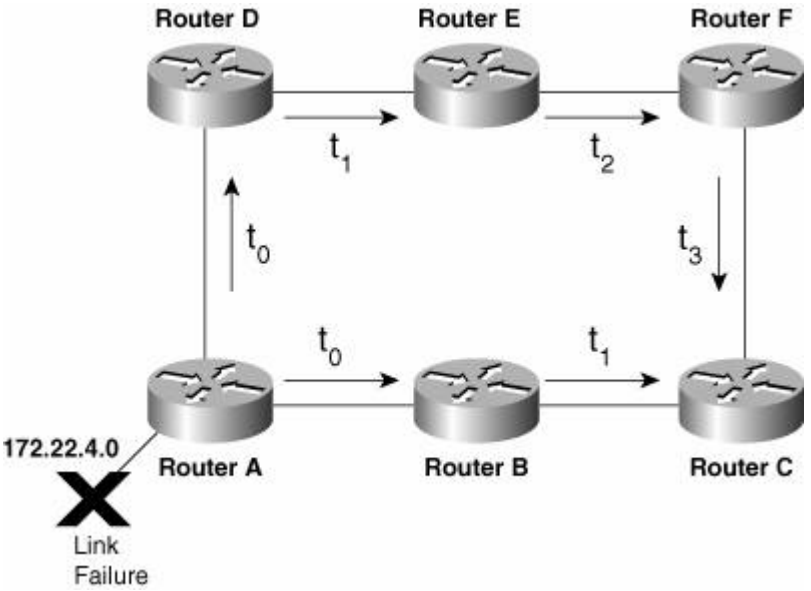
After the adjacencies are established, the routers might begin sending out LSAs. As the term *flooding* implies, the advertisements are sent to every neighbor. In turn, each received LSA is copied and forwarded to every neighbor except the one that sent the LSA. This process is the source of one of link state's advantages over distance vector. LSAs are forwarded almost immediately, whereas distance vector protocols based on the Bellman-Ford algorithm must run its algorithm and update its route table before routing updates, even the triggered ones, can be forwarded. As a result, link state protocols converge much faster than distance vector protocols converge when the topology changes.

The flooding process is the most complex piece of a link state protocol. There are several ways to make flooding more efficient and more reliable, such as using unicast and multicast addresses, checksums, and positive acknowledgments. These topics are examined in the protocol-specific chapters, but two procedures are vitally important to the flooding process: sequencing and aging.

### Sequence Numbers

A difficulty with flooding, as described so far, is that when all routers have received all LSAs, the flooding must stop. A time-to-live value in the packets could simply be relied on to expire, but it is hardly efficient to permit LSAs to wander the network until they expire. Take the network in Figure 4-8. Subnet 172.22.4.0 at Router A has failed, and A has flooded an LSA to its neighbors B and D, advertising the new state of the link. B and D dutifully flood to their neighbors, and so on.

Figure 4-8. When a topology change occurs, LSAs advertising the change will be flooded throughout the network.




Look next at what happens at Router C. An LSA arrives from Router B at time $t_1$, is entered into C's topological database, and is forwarded to Router F. At some later time $t_3$, another copy of the same LSA arrives from the longer A-D-E-F-C route. Router C sees that it already has the LSA in its database; the question is, should C forward this LSA to Router B? The answer is no because B has already received the advertisement. Router C knows this because the sequence number of the LSA it received from Router F is the same as the sequence number of the LSA it received earlier from Router B.

When Router A sent out the LSA, it included an identical sequence number in each copy. This sequence number is recorded in the routers' topological databases along with the rest of the LSA; when a router receives an LSA that is already in the database and its sequence number is the same, the received information is discarded. If the information is the same but the sequence number is greater, the received information and new sequence number are entered into the database and the LSA is flooded. In this way, flooding is abated when all routers have seen a copy of the most recent LSA.

As described so far, it seems that routers could merely verify that their link state databases contain the same LSA as the newly received LSA and make a flood/discard decision based on that information, without needing a sequence number. But imagine that immediately after Figure 4-8's network 172.22.4.0 failed, it came back up. Router A might send out an LSA advertising the network as down, with a sequence number of 166; then it sends out a new LSA announcing the same network as up, with a sequence number of 167. Router C receives the down LSA and then the up LSA from the A-B-C path, but then it receives a delayed down LSA from the A-D-E-F-C path. Without the sequence numbers, C would not know whether or not to believe the delayed down LSA. With sequence numbers, C's database will indicate that the information from Router A has a sequence number of 167; the late LSA has a sequence number of 166 and is therefore recognized as old information and discarded.

Because the sequence numbers are carried in a set field within the LSAs, the numbers must have some upper boundary. What happens when this maximum sequence number is reached?

## Linear Sequence Number Spaces

One approach is to use a linear sequence number space so large that it is unlikely the upper limit will ever be reached. If, for instance, a 32-bit field is used, there are $2^{32}$ = 4,294,967,296 available sequence numbers starting with zero. Even if a router was creating a new link state packet every 10 seconds, it would take 1361 years to exhaust the sequence number supply; few routers are expected to last so long.

In this imperfect world, unfortunately, malfunctions occur. If a link state routing process somehow runs out of sequence numbers, it must shut itself down and stay down long enough for its LSAs to age out of all databases before starting over at the lowest sequence number (see the section "Aging," later in this chapter).

A more common difficulty presents itself during router restarts. If Router A restarts, it probably will have no way of remembering the sequence number it last used and must begin again at, say, one. But if its neighbors still have Router A's previous sequence numbers in their databases, the lower sequence numbers will be interpreted as older sequence numbers and will be ignored. Again, the routing process must stay down until all old LSAs are aged out of the network. Given that a maximum age might be an hour or more, this solution is not very attractive.

A better solution is to add a new rule to the flooding behavior described thus far: If a restarted router issues to a neighbor an LSA with a sequence number that appears to be older than the neighbor's stored sequence number, the neighbor will send its own stored LSA and sequence number back to the router. The router will thus learn the sequence number it was using before it restarted and can adjust accordingly.

Care must be taken, however, that the last-used sequence number was not close to the maximum; otherwise, the restarting router will simply have to restart again. A rule must be set limiting the "jump" the router might make in sequence numbersfor instance, a rule might say that the sequence numbers cannot make a single increase more than one-half the total sequence number space. (The actual formulas are more complex than this example, taking into account age constraints.)

IS-IS uses a 32-bit linear sequence number space.

## Circular Sequence Number Spaces

Another approach is to use a circular sequence number space, where the numbers "wrap"that is, in a 32-bit space, the number following 4,294,967,295 is 0. Malfunctions can cause interesting dilemmas here, too. A restarting router might encounter the same believability problem as discussed for linear sequence numbers.

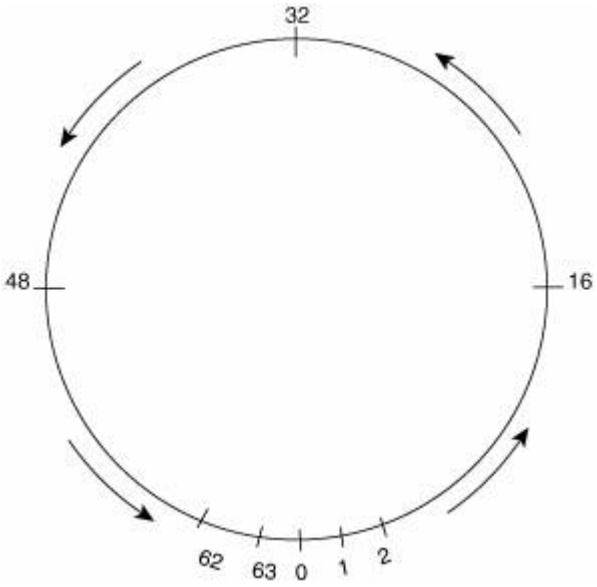
Circular sequence numbering creates a curious bit of illogic. If $x$ is any number between 1 and 4,294,967,295 inclusive, then $0 < x < 0$. This situation can be managed in well-behaved networks by asserting two rules for determining when a sequence number is greater than or less than another sequence number. Given a sequence number space $n$ and two sequence numbers $a$ and $b$, $a$ is considered more recent (of larger magnitude) in either of the following situations:

$$a > b, \text{ and } (a - b) \leq n/2$$
$$a < b, \text{ and } (b - a) > n/2$$

For the sake of simplicity, take a sequence number space of six bits, shown in Figure 4-9:

$$n = 2^6 = 64$$
$$n/2 = 32.$$

Figure 4-9. Six-Bit Circular Address Space




Given two sequence numbers 48 and 18, 48 is more recent because by rule (1)

$$48 > 18 \text{ and } (48 - 18) = 30 \text{ and } 30 < 32.$$

Given two sequence numbers 3 and 48, 3 is more recent because by rule (2)

$$3 < 48 \text{ and } (48 - 3) = 45 \text{ and } 45 > 32.$$

Given two sequence numbers 3 and 18, 18 is more recent because by rule (1)

$$18 > 3 \text{ and } (18 - 3) = 15 \text{ and } 15 < 32.$$

So the rules seem to enforce circularity.

But what about a not-so-well-behaved network? Imagine a network running a six-bit sequence number space. Now imagine that one of the routers on the network malfunctions, but as it does so, it blurts out three identical LSAs with a sequence number of 44 (101100). Unfortunately, a neighboring router is also malfunctioningit is dropping bits. The neighbor drops a bit in the sequence number field of the second LSA, drops yet another bit in the third LSA, and floods all three. The result is three identical LSAs with three different sequence numbers:

| | |
|---|---|
| 44 | (101100) |
| 40 | (101000) |
| 8 | (001000) |

Applying the circularity rules reveals that 44 is more recent than 40, which is more recent than 8, which is more recent than 44! The result is that every LSA will be continuously flooded, and databases will be continually overwritten with the "latest" LSA, until finally buffers become clogged with LSAs, CPUs become overloaded, and the entire network comes crashing down.

This chain of events sounds quite far-fetched. It is, however, factual. The ARPANET, the precursor of the modern Internet, ran an early link state protocol with a six-bit circular sequence number space; on October 27, 1980, two routers experiencing the malfunctions just described brought the entire ARPANET to a standstill.[11]

[11] E. C. Rosen. "Vulnerabilities of Network Control Protocols: An Example." Computer Communication Review, July 1981.

## Lollipop-Shaped Sequence Number Spaces

This whimsically-named construct was proposed by Dr. Radia Perlman.[12] Lollipop-shaped sequence number spaces are a hybrid of linear and circular sequence number spaces; if you think about it, a lollipop has a linear component and a circular component. The problem with circular spaces is that there is no number less than all other numbers. The problem with linear spaces is that they arewellnot circular. That is, their set of sequence numbers is finite.

[12] R. Perlman. "Fault-Tolerant Broadcasting of Routing Information." Computer Networks, Vol. 7, December 1983, pp. 395405.

When Router A restarts, it would be nice to begin with a number *a* that is less than all other numbers. Neighbors will recognize this number for what it is, and if they have a pre-restart number *b* in their databases from Router A, they can send that number to Router A and Router A will jump to that sequence number. Router A might be able to send more than one LSA before hearing about the sequence number it had been using before restarting. Therefore, it is important to have enough restart numbers so that A cannot use them all before neighbors either inform it of a previously used number, or the previously used number ages out of all databases.

These linear restart numbers form the stick of the lollipop. When they have been used up, or after a neighbor has provided a sequence number to which A can jump, A enters a circular number space, the candy part of the lollipop.

One way of designing a lollipop address space is to use signed sequence numbers, where $k < 0 < k$. The negative numbers counting up from $k$ to 1 form the stick, and the positive numbers from 0 to $k$ are the circular space. Perlman's rules for the sequence numbers are as follows. Given two numbers $a$ and $b$ and a sequence number space $n$, $b$ is more recent than $a$ if and only if

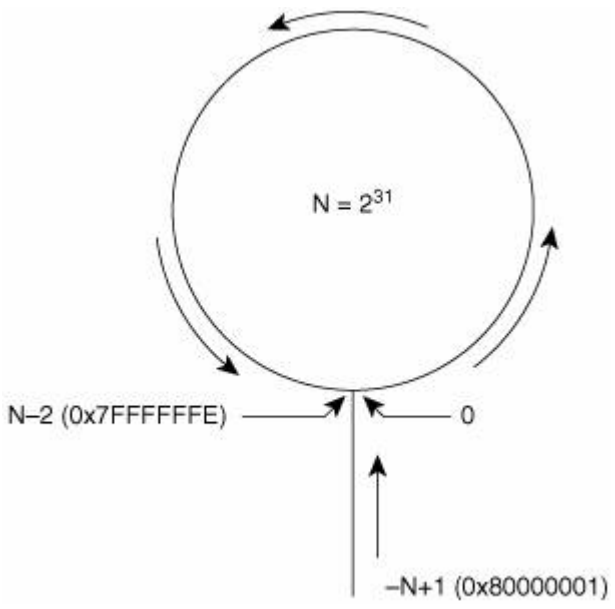
$a < 0$ and $a < b$, or

$a > 0$, $a < b$, and $(b\ a) < n/2$, or

$a > 0$, $b > 0$, $a > b$, and $(a\ b) > n/2$

Figure 4-10 shows an implementation of the lollipop-shaped sequence number space. A 32-bit signed number space N is used, yielding $2^{31}$ positive numbers and $2^{31}$ negative numbers. N ($2^{31}$, or 0x80000000) and N 1 ($2^{31}$ 1, or 0x7FFFFFFF) are not used. A router coming online will begin its sequence numbers at N + 1 (0x80000001) and increment up to zero, at which time it has entered the circular number space. When the sequence reaches N 2 (0x7FFFFFFE), the sequence wraps back to zero (again, N 1 is unused).

Figure 4-10. Lollipop-shaped sequence number space.




Next, suppose the router restarts. The sequence number of the last LSA sent before the restart is 0x00005de3 (part of the circular sequence space). As it synchronizes its database with its neighbor after the restart, the router sends an LSA with a sequence number of 0x80000001 (N + 1). The neighbor looks into its own database and finds the pre-restart LSA with a sequence number of 0x00005de3. The neighbor sends this LSA to the restarted router, essentially saying, "This is where you left off." The restarted router then records the LSA with the positive sequence number. If it needs to send a new copy of the LSA at some future time, the new sequence number will be 0x00005de6.

Lollipop sequence spaces were used with the original version of OSPF, OSPFv1 (RFC 1131). Although the use of signed numbers was an improvement over the linear number space, the circular part was found to be vulnerable to the same ambiguities as a purely circular space. The deployment of OSPFv1 never progressed beyond the experimental stage. The current version of OSPF, OSPFv2 (originally specified in RFC 1247), adopts the best features of linear and lollipop sequence number spaces. It uses a signed number space like lollipop sequence numbers, beginning with 0x80000001. However, when the sequence number goes positive, the sequence space continues to be linear until it reaches the maximum of 0x7FFFFFFF. At that point, the OSPF process must flush the LSA from all link state databases before restarting.

## Aging

The LSA format should include a field for the age of the advertisement. When an LSA is created, the router sets this field to one. As the packet is flooded, each router increments the age of the advertisement.[13]

[13] Of course, another option would be to start with some maximum age and decrement. OSPF increments; IS-IS decrements.

This process of aging adds another layer of reliability to the flooding process. The protocol defines a maximum age difference (MaxAgeDiff) value for the network. A router might receive multiple copies of the same LSA with identical sequence numbers but different ages. If the difference in the ages is lower than the MaxAgeDiff, it is assumed that the age difference was the result of normal network latencies; the original LSA in the database is retained, and the newer LSA (with the greater age) is not flooded. If the difference is greater than the MaxAgeDiff value, it is assumed that an anomaly has occurred in the network in which a new LSA was sent without incrementing the sequence number. In this case, the newer LSA will be recorded, and the packet will be flooded. A typical MaxAgeDiff value is 15 minutes (used by OSPF).

The age of an LSA continues to be incremented as it resides in a link state database. If the age for a link state record is incremented up to some maximum age (MaxAge)again defined by the specific routing protocolthe LSA, with age field set to the MaxAge value, is flooded to all neighbors and the record is deleted from the databases.

If the LSA is to be flushed from all databases when MaxAge is reached, there must be a mechanism to periodically validate the LSA and reset its timer before MaxAge is reached. A link state refresh time (LSRefreshTime)[14] is established; when this time expires, a router floods a new LSA to all its neighbors, who will reset the age of the sending router's records to the new received age. OSPF defines a MaxAge of 1 hour and an LSRefreshTime of 30 minutes.

[14] LSRefreshTime, MaxAge, and MaxAgeDiff are OSPF architectural constants.

## Link State Database

In addition to flooding LSAs and discovering neighbors, a third major task of the link state routing protocol is establishing the link state database. The link state or topological database stores the LSAs as a series of records. Although a sequence number and age and possibly other information are included in the LSA, these variables exist mainly to manage the flooding process. The important information for the shortest path determination process is the advertising router's ID, its attached networks and neighboring routers, and the cost associated with those networks or neighbors. As the previous sentence implies, LSAs might include two types of generic information:[15]

[15] Actually, there can be more than two types of information and multiple types of link state packets. They are covered in the chapters on specific link state protocols.

- *Router link information* advertises a router's adjacent neighbors with a triple of (Router ID, Neighbor ID, Cost), where cost is the cost of the link to the neighbor.
- *Stub network information* advertises a router's directly connected stub subnets (subnets with no neighbors) with a triple of (Router ID, Network ID, Cost).

The shortest path first (SPF) algorithm is run once for the router link information to establish shortest paths to each router, and then stub network information is used to add these networks to the routers. Figure 4-11 shows a network of routers and the links between them; stub networks are not shown for the sake of simplicity. Notice that several links have different costs associated with them at each end. A cost is associated with the outgoing direction of an interface. For instance, the link from RB to RC has a cost of 1, but the same link has a cost of 5 in the RC to RB direction.

Figure 4-11. Link costs are calculated for the outgoing direction from an interface and do not necessarily have to be the same at all interfaces on a link.

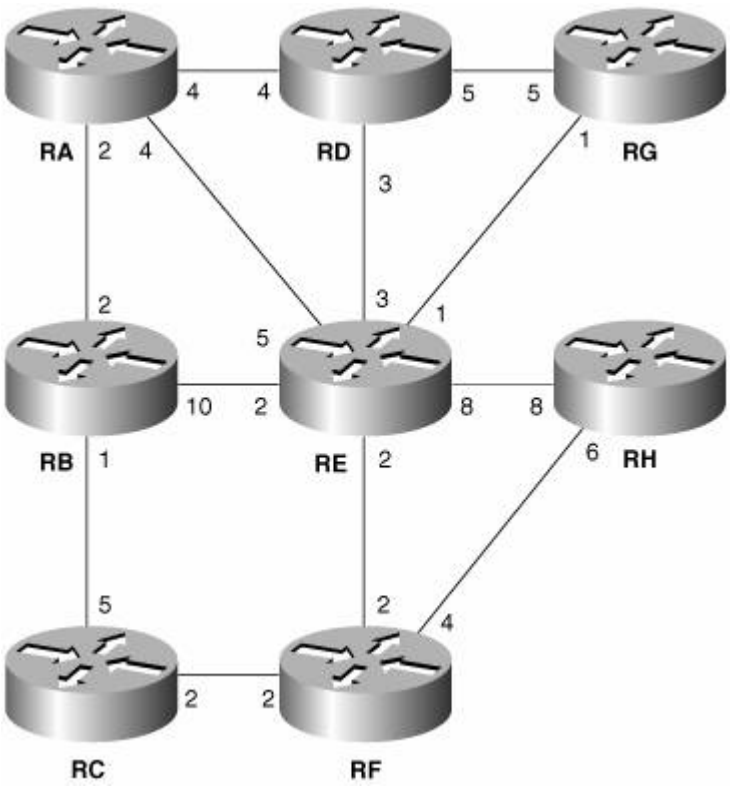



Table 4-2 shows a generic link state database for the network of Figure 4-11, a copy of which is stored in every router. As you read through this database, you will see that it completely describes the network. Now it is possible to compute a tree that describes the shortest path to each router by running the SPF algorithm.

Table 4-2. Topological database for the network in Figure 4-11.

| Router ID | Neighbor | Cost |
|---|---|---|
| RA | RB | 2 |
| RA | RD | 4 |
| RA | RE | 4 |
| RB | RA | 2 |
| RB | RC | 1 |
| RB | RE | 10 |
| RC | RB | 5 |
| RC | RF | 2 |
| RD | RA | 4 |
| RD | RE | 3 |

| Router ID | Neighbor | Cost |
|---|---|---|
| RD | RG | 5 |
| RE | RA | 5 |
| RE | RB | 2 |
| RE | RD | 3 |
| RE | RF | 2 |
| RE | RG | 1 |
| RE | RH | 8 |
| RF | RC | 2 |
| RF | RE | 2 |
| RF | RH | 4 |
| RG | RD | 5 |
| RG | RE | 1 |
| RH | RE | 8 |
| RH | RF | 6 |

## SPF Algorithm

It is unfortunate that Dijkstra's algorithm is so commonly referred to in the routing world as the shortest path first algorithm. After all, the objective of every routing protocol is to calculate shortest paths. It is also unfortunate that Dijkstra's algorithm is often made to appear more esoteric than it really is; many writers just can't resist putting it in set theory notation. The clearest description of the algorithm comes from E. W. Dijkstra's original paper. Here it is in his own words, followed by a "translation" for the link state routing protocol:

Construct [a] tree of minimum total length between the *n* nodes. (The tree is a graph with one and only one path between every two nodes.)

In the course of the construction that we present here, the branches are divided into three sets:

I. the branches definitely assigned to the tree under construction (they will be in a subtree);

II. the branches from which the next branch to be added to set I, will be selected;

III. the remaining branches (rejected or not considered).

The nodes are divided into two sets:

A. the nodes connected by the branches of set I,

B. the remaining nodes (one and only one branch of set II will lead to each of these nodes).

We start the construction by choosing an arbitrary node as the only member of set A, and by placing all branches that end in this node in set II. To start with, set I is empty. From then

onwards we perform the following two steps repeatedly.

Step 1. The shortest branch of set II is removed from this set and added to set I. As a result, one node is transferred from set B to set A.

Step 2. Consider the branches leading from the node, which has just been transferred to set A, to the nodes that are still in set B. If the branch under construction is longer than the corresponding branch in set II, it is rejected; if it is shorter, it replaces the corresponding branch in set II, and the latter is rejected.

We then return to step 1 and repeat the process until sets II and B are empty. The branches in set I form the tree required.[16]

[16] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs." Numerische Mathematik, Vol. 1, 1959, pp. 269271.

Adapting the algorithm for routers, first note that Dijkstra describes three sets of branches: I, II, and III. In the router, three databases represent the three sets:

- *The Tree Database* This database represents set I. Links (branches) are added to the shortest path tree by adding them here. When the algorithm is finished, this database will describe the shortest path tree.
- *The Candidate Database* This database corresponds to set II. Links are copied from the link state database to this list in a prescribed order, where they become candidates to be added to the tree.
- *The Link State Database* The repository of all links, as has been previously described. This topological database corresponds to set III.

Dijkstra also specifies two sets of nodes, set A and set B. Here the nodes are routers. Specifically, they are the routers represented by Neighbor ID in the Router Links triples (Router ID, Neighbor ID, Cost). Set A comprises the routers connected by the links in the Tree database. Set B is all other routers. Since the whole point is to find a shortest path to every router, set B should be empty when the algorithm is finished.

Here's a version of Dijkstra's algorithm adapted for routers:

1. A router initializes the Tree database by adding itself as the root. This entry shows the router as its own neighbor, with a cost of 0.
2. All triples in the link state database describing links to the root router's neighbors are added to the Candidate database.
3. The cost from the root to each link in the Candidate database is calculated. The link in the Candidate database with the lowest cost is moved to the Tree database. If two or more links are an equally low cost from the root, choose one.
4. The Neighbor ID of the link just added to the Tree database is examined. With the exception of any triple whose Neighbor ID is already in the Tree database, triples in the link state database describing that router's neighbors are added to the Candidate database.
5. If entries remain in the Candidate database, return to step 3. If the Candidate database is empty, terminate the algorithm. At termination, a single Neighbor ID entry in the Tree database should represent every router, and the shortest path tree is complete.

Table 4-3 summarizes the process and results of applying Dijkstra's algorithm to build a shortest path tree for the network in Figure 4-11. Router RA from Figure 4-11 is running the algorithm, using the link state database of Table 4-2. Figure 4-12 shows the shortest path tree constructed

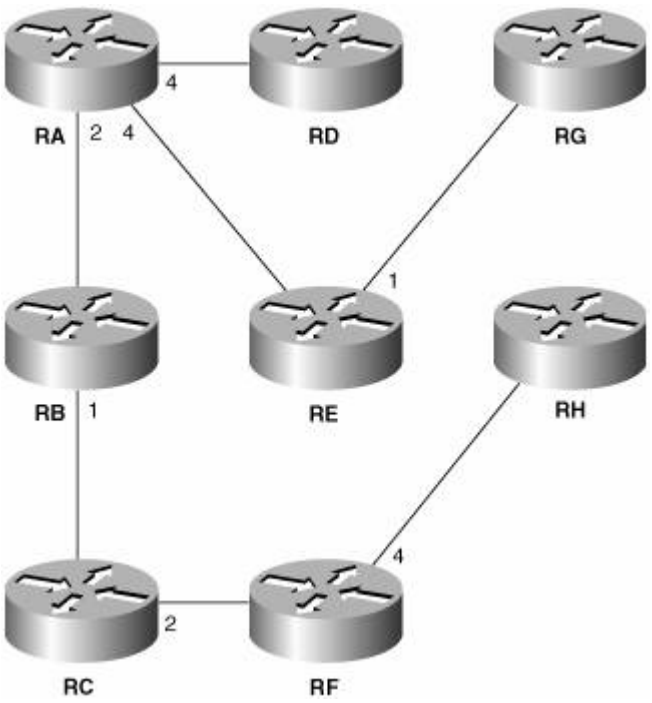
for Router RA by this algorithm. After each router calculates its own tree, it can examine the other routers' network link information and add the stub networks to the tree fairly easily. From this information, entries might be made into the routing table.

Table 4-3. Dijkstra's Algorithm Applied to the Database of Table 4-2

| Candidate | Cost to Root | Tree | Description |
|---|---|---|---|
| | | RA,RA,0 | Router A adds itself to the tree as root. |
| RA,RB,2<br>RA,RD,4<br>RA,RE,4 | 2<br>4<br>4 | RA,RA,0 | The links to all of RA's neighbors are added to the candidate list. |
| RA,RD,4<br>RA,RE,4<br>RB,RC,1<br>~~RB,RE,10~~ | 4<br>4<br>3<br>12 | RA,RA,0<br>RA,RB,2 | (RA,RB,2) is the lowest-cost link on the candidate list, so it is added to the tree. All of RB's neighbors except those already in the tree are added to the candidate list. (RA,RE,4) is a lower-cost link to RE than (RB,RE,10), so the latter is dropped from the candidate list. |
| RA,RD,4<br>RA,RE,4<br>RC,RF,2 | 4<br>4<br>5 | RA,RA,0<br>RA,RB,2<br>RB,RC,1 | (RB,RC,1) is the lowest-cost link on the candidate list, so it is added to the tree. All of RC's neighbors except those already on the tree become candidates. |
| RA,RE,4<br>RC,RF,2<br>RD,RE,3<br>RD,RG,5 | 4<br>5<br>7<br>9 | RA,RA,0<br>RA,RB,2<br>RB,RC,1<br>RA,RD,4 | (RA,RD,4) and (RA,RE,4) are both a cost of 4 from RA; (RC,RF,2) is a cost of 5. (RA,RD,4) is added to the tree and its neighbors become candidates. Two paths to RE are on the candidate list; (RD,RE,3)is a higher cost from RA and is dropped. |
| RC,RF,2<br>~~RD,RG,5~~<br>RE,RF,2<br>RE,RG,1<br>RE,RH,8 | 5<br>9<br>6<br>5<br>12 | RA,RA,0<br>RA,RB,2<br>RB,RC,1<br>RA,RD,4<br>RA,RE,4 | (RF,RE,1) is added to the tree. All of RE's neighbors not already on the tree are added to the candidate list. The higher-cost link to RG is dropped. |
| RE,RF,2<br>RE,RG,1<br>~~RE,RH,8~~<br>RF,RH,4 | 6<br>5<br>12<br>9 | RA,RA,0<br>RA,RB,2<br>RB,RC,1<br>RA,RD,4<br>RA,RE,4<br>RC,RF,2 | (RC,RF,2) is added to the tree, and its neighbors are added to the candidate list. (RE,RG,1) could have been selected instead because it has the same cost (5) from RA. The higher-cost path to RH is dropped. |
| RF,RH,4 | 9 | RA,RA,0 | (RE,RG,1) is added to the tree. RG has no |

| Candidate | Cost to Root | Tree | Description |
|---|---|---|---|
| | | RA,RB,2<br>RB,RC,1<br>RA,RD,4<br>RA,RE,4<br>RC,RF,2<br>RE,RG,1 | neighbors that are not already on the tree, so nothing is added to the candidate list. |
| | | RA,RA,0<br>RA,RB,2<br>RB,RC,1<br>RA,RD,4<br>RA,RE,4<br>RC,RF,2<br>RE,RG,1<br>RF,RH,4 | (RF,RH,4) is the lowest-cost link on the candidate list, so it is added to the tree. No candidates remain on the list, so the algorithm is terminated. The shortest path tree is complete. |

Figure 4-12. Shortest path tree derived by the algorithm in Table 4-3.




## Areas

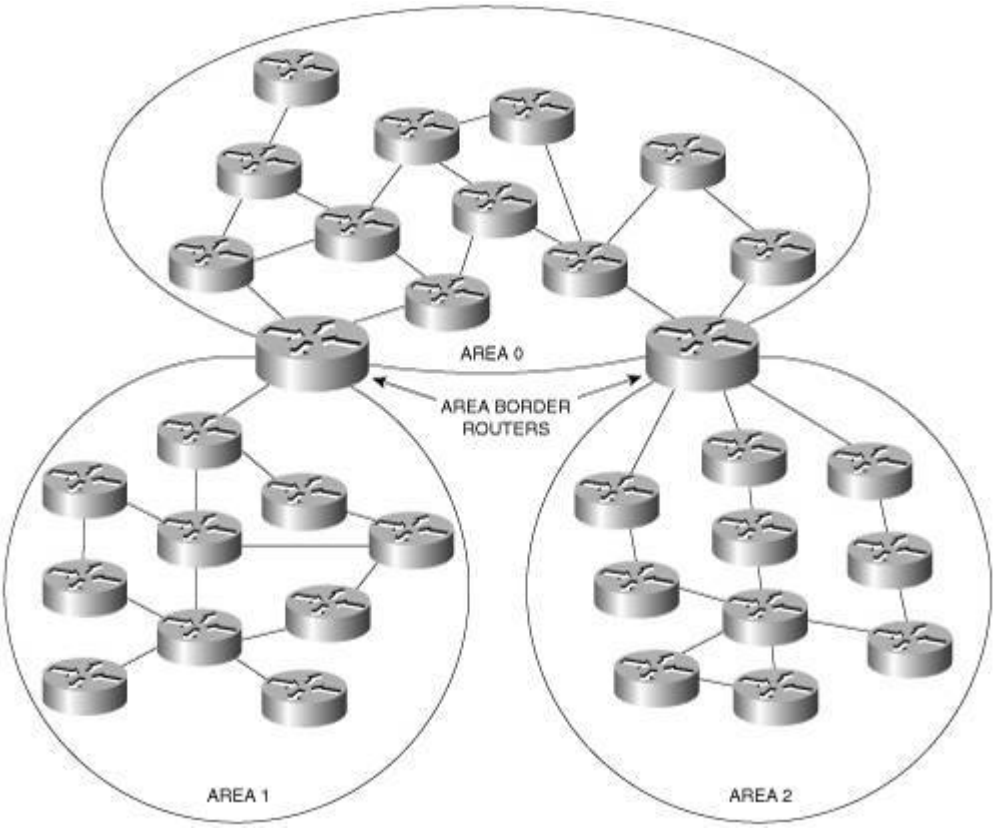
An *area* is a subset of the routers that make up a network. Dividing a network into areas is a response to three concerns commonly expressed about link state protocols:

- The necessary databases require more memory than a distance vector protocol requires.
- The complex algorithm requires more CPU time than a distance vector protocol requires.
- The flooding of link state packets adversely affects available bandwidth, particularly in unstable networks.

Modern link state protocols and the routers that run them are designed to reduce these effects, but cannot eliminate them. The last section examined what the link state database might look like, and how an SPF algorithm might work, for a small eight-router network. Remember that the stub networks that would be connected to those eight routers and that would form the leaves of the SPF tree were not taken into consideration. Now, imagine an 8000-router network, and you can understand the concern about the impact on memory, CPU, and bandwidth.

This impact can be greatly reduced by the use of areas, as in Figure 4-13. When a network is subdivided into areas, the routers within an area need to flood LSAs only within that area and therefore need to maintain a link state database only for that area. The smaller database means less required memory in each router and fewer CPU cycles to run the SPF algorithm on that database. If frequent topology changes occur, the resulting flooding will be confined to the area of the instability.

Figure 4-13. Use of areas reduces link state's demand for system resources.




The routers connecting two areas (*Area Border Routers*, in OSPF terminology) belong to both areas and must maintain separate topological databases for each. Just as a host on one network that wants to send a packet to another network only needs to know how to find its local router, a router in one area that wants to send a packet to another area only needs to know how to find its local Area Border Router. In other words, the intra-area router/inter-area router relationship is the same as the host/router relationship but at a higher hierarchical level.

Distance vector protocols, such as RIP and IGRP, do not use areas. Given that these protocols have no recourse but to see a large network as a single entity, must calculate a route to every network, and must broadcast the resulting huge route table every 30 or 90 seconds, it becomes

# Interior and Exterior Gateway Protocols

Areas introduce a hierarchy to the network architecture. Another layer is added to this hierarchical structure by grouping areas into larger areas. These higher-level areas are called *autonomous* systems in the IP world and *routing domains* in the ISO world.

An autonomous system was once defined as a group of routers under a common administrative domain running a common routing protocol. Given the fluidity of modern networking life, the latter part of the definition is no longer very accurate. Departments, divisions, and even entire companies frequently merge, and networks that were designed with different routing protocols merge along with them. The result is that many networks nowadays combine multiple routing protocols with multiple degrees of inelegance, all under common administrations. So a contemporary definition of an autonomous system is a network under a common administration.

The routing protocols that run within an autonomous system are referred to as *Interior Gateway Protocols* (IGP). All the protocols given in this chapter as examples of distance vector or link state protocols are IGPs.

Routing protocols that route between autonomous systems or routing domains are referred to as *Exterior Gateway Protocols* (EGP). Whereas IGPs discover paths between networks, EGPs discover paths between autonomous systems. Examples of EGPs include the following:

- Border Gateway Protocol (BGP) for IP
- Exterior Gateway Protocol (EGP) for IP (yes, an EGP named EGP)
- The ISO's InterDomain Routing Protocol (IDRP)

Novell also incorporates an EGP functionality, called Level 3 Routing, into NLSP.

Having given these definitions, it must be said that the common usage of the term *autonomous system* is not so absolute. Various standards documents, literature, and people tend to give various meanings to the term. As a result, it is important to understand the context in which one is reading or hearing the term.

This book uses *autonomous system* in one of two contexts:

- *Autonomous system* might refer to a routing domain, as defined at the beginning of this section. In this context, an autonomous system is a system of one or more IGPs that is autonomous from other systems of IGPs. An EGP is used to route between these autonomous systems.
- *Autonomous system* might also refer to a *process domain*, or a single IGP process that is autonomous from other IGP processes. For example, a system of OSPF-speaking routers might be referred to as an OSPF autonomous system. EIGRP also uses autonomous system in this context. Redistribution is used to route between these autonomous systems.

The context will indicate which form of autonomous system is under discussion at different points throughout this book.

# Static or Dynamic Routing?

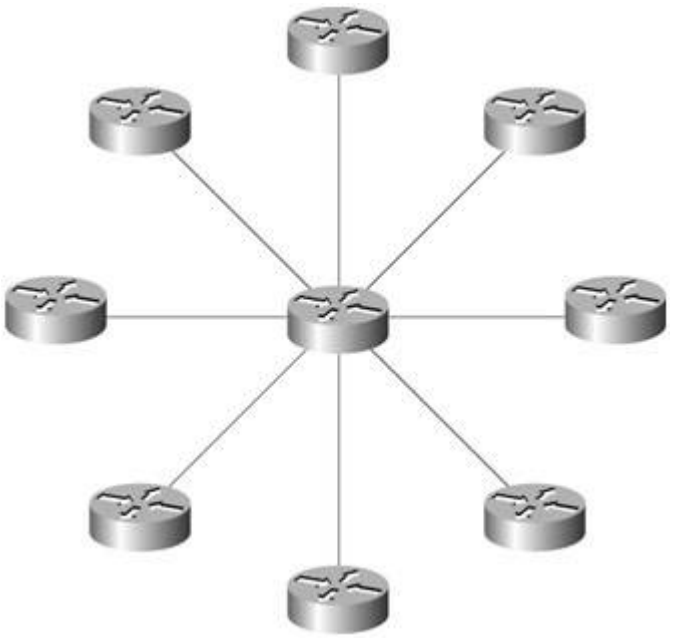
When reading (or being lectured about) all the glorious details of dynamic routing protocols, it's hard not to come away with the impression that dynamic routing is always better than static routing. It's important to keep in mind that the primary duty of a dynamic routing protocol is to automatically detect and adapt to topological changes in the network. The price of this "automation" is paid in bandwidth and maybe queue space, in memory, and in processing time.

Another price of dynamic routing is a reduced control of routing choices. The routing protocol decides what the best path is to a given destination, you don't. If precise control of path selection is important, particularly when the path you want is different from the path a routing protocol would choose, static routing is a better choice.

A frequent objection to static routing is that it is hard to administer. This criticism might be true of medium to large topologies with many alternative routes, but it is certainly not true of small networks with few or no alternative routes.

The network in Figure 4-14 has a hub-and-spoke topology popular in smaller networks. If a spoke to any router breaks, is there another route for a dynamic routing protocol to choose? This network is an ideal candidate for static routing. Configure one static route in the hub router for each spoke router and a single default route in each spoke router pointing to the hub, and the network is ready to go. (Default routes are covered in Chapter 12, "Default Routes and On-Demand Routing.")

Figure 4-14. This hub-and-spoke network is ideal for static routing.



When designing a network, the simplest solution is almost always the best solution. It is good practice to choose a dynamic routing protocol only after determining that static routing is not a practical solution for the design at hand.

# Looking Ahead

Now that the basics of dynamic routing protocols have been examined, it is time to examine specific routing protocols. The following chapter looks at RIP, the oldest and simplest of the dynamic routing protocols.

# Recommended Reading

Perlman, R. *Interconnections: Bridges and Routers*. Reading, Massachusetts: Addison-Wesley; 1992.

# Review Questions

1. What is a routing protocol?
2. What basic procedures should a routing algorithm perform?
3. Why do routing protocols use metrics?
4. What is convergence time?
5. What is load balancing? Name four different types of load balancing.
6. What is a distance vector routing protocol?
7. Name several problems associated with distance vector protocols.
8. What are neighbors?
9. What is the purpose of route invalidation timers?
10. Explain the difference between simple split horizon and split horizon with poisoned reverse.
11. What is the counting-to-infinity problem, and how can it be controlled?
12. What are holddown timers, and how do they work?
13. What are the differences between distance vector and link state routing protocols?
14. What is the purpose of a topological database?
15. Explain the basic steps involved in converging a link state network.
16. Why are sequence numbers important in link state protocols?
17. What purpose does aging serve in a link state protocol?
18. Explain how an SPF algorithm works.
19. How do areas benefit a link state network?
20. What is an autonomous system?
21. What is the difference between an IGP and an EGP?

# Part II: Interior Routing Protocols

Chapter 5 Routing Information Protocol (RIP)

Chapter 6 RIPv2, RIPng, and Classless Routing

Chapter 7 Enhanced Interior Gateway Routing Protocol (EIGRP)

Chapter 8 OSPFv2

Chapter 9 OSPFv3

Chapter 10 Integrated IS-IS

# Chapter 5. Routing Information Protocol (RIP)

This chapter covers the following subjects:

- Operation of RIP
- Configuring RIP
- Troubleshooting RIP

The oldest of the distance vector IP routing protocols still in widespread use, RIP currently exists in two versions. This chapter deals with version 1 of RIP. Chapter 6, "RIPv2, RIPng, and Classless Routing," covers Version 2, which adds several enhancements to RIPv1. Most notably, RIPv1 is a classful routing protocol, whereas RIPv2 is classless. This chapter introduces classful routing, and Chapter 6 introduces classless routing. Chapter 6 also introduces RIPng, which is an adaptation of RIPv2 for support of IPv6.

Distance vector protocols, based on the algorithms developed by Bellman,[1] Ford, and Fulkerson,[2] were implemented beginning in 1969 in networks such as ARPANET and CYCLADES. In the mid-1970s Xerox developed a protocol called PARC[3] Universal Protocol, or PUP, to run on its 3-Mbps experimental predecessor to modern Ethernet. PUP was routed by the Gateway Information Protocol (GWINFO). PUP evolved into the Xerox Network Systems (XNS) protocol suite; concurrently, the Gateway Information Protocol became the XNS Routing Information Protocol. In turn, XNS RIP has become the precursor of such common routing protocols as Novell's IPX RIP, AppleTalk's Routing Table Maintenance Protocol (RTMP), and, of course, IP RIP.

[1] R. E. Bellman. *Dynamic Programming*. Princeton, New Jersey: Princeton University Press; 1957.

[2] L. R. Ford Jr. and D. R. Fulkerson. *Flows in Networks*. Princeton, New Jersey: Princeton University Press; 1962.

[3] Palo Alto Research Center.

The 4.2 Berkeley Software Distribution of UNIX, released in 1982, implemented RIP in a daemon called *routed*; many more recent versions of UNIX are based on the popular 4.2BSD and implement RIP in either *routed* or *gated*.[4] Oddly enough, a standard for RIP was not released until 1988, after the protocol was in extensive deployment. That was RFC 1058, written by Charles Hedrick, and it remains the only formal standard of RIPv1.

[4] Pronounced "route-dee" and "gate-dee."

Depending on the literature you reads, RIP is either unjustly maligned or undeservedly popular. Although it lacks the capabilities of many of its successors, its simplicity and widespread use mean that compatibility problems between implementations are rare. RIP was designed for smaller networks in which the data links are fairly homogeneous. Within these constraints, and especially within many UNIX environments, RIP continues to be a popular routing protocol.

# Operation of RIP

The RIP process operates from UDP port 520; all RIP messages are encapsulated in a UDP (User Datagram Protocol) segment with both the Source and Destination Port fields set to that value. RIP defines two message types: *Request messages* and *Response messages*. A Request message is used to ask neighboring routers to send an update. A Response message carries the update. The metric used by RIP is hop count, with 1 signifying a directly connected network of the advertising router and 16 signifying an unreachable network.

On startup, RIP broadcasts a packet carrying a Request message out each RIP-enabled interface. The RIP process then enters a loop, listening for RIP Request or Response messages from other routers. Neighbors receiving the Request send a Response containing their route table.

When the requesting router receives the Response messages, it processes the enclosed information. If a particular route entry included in the update is new, it is entered into the route table along with the address of the advertising router, which is read from the source address field of the update packet. If the route is for a network that RIP has already entered in the table, the existing entry will be replaced only if the new route has a lower hop count. If the advertised hop count is higher than the recorded hop count and the update was originated by the recorded next-hop router, the route will be marked as unreachable for a specified holddown period. If at the end of that time the same neighbor is still advertising the higher hop count, the new metric will be accepted.[5]

[5] Holddowns are used by Cisco IOS, but are not part of the stability features specified in RFC 1058.

## RIP Timers and Stability Features

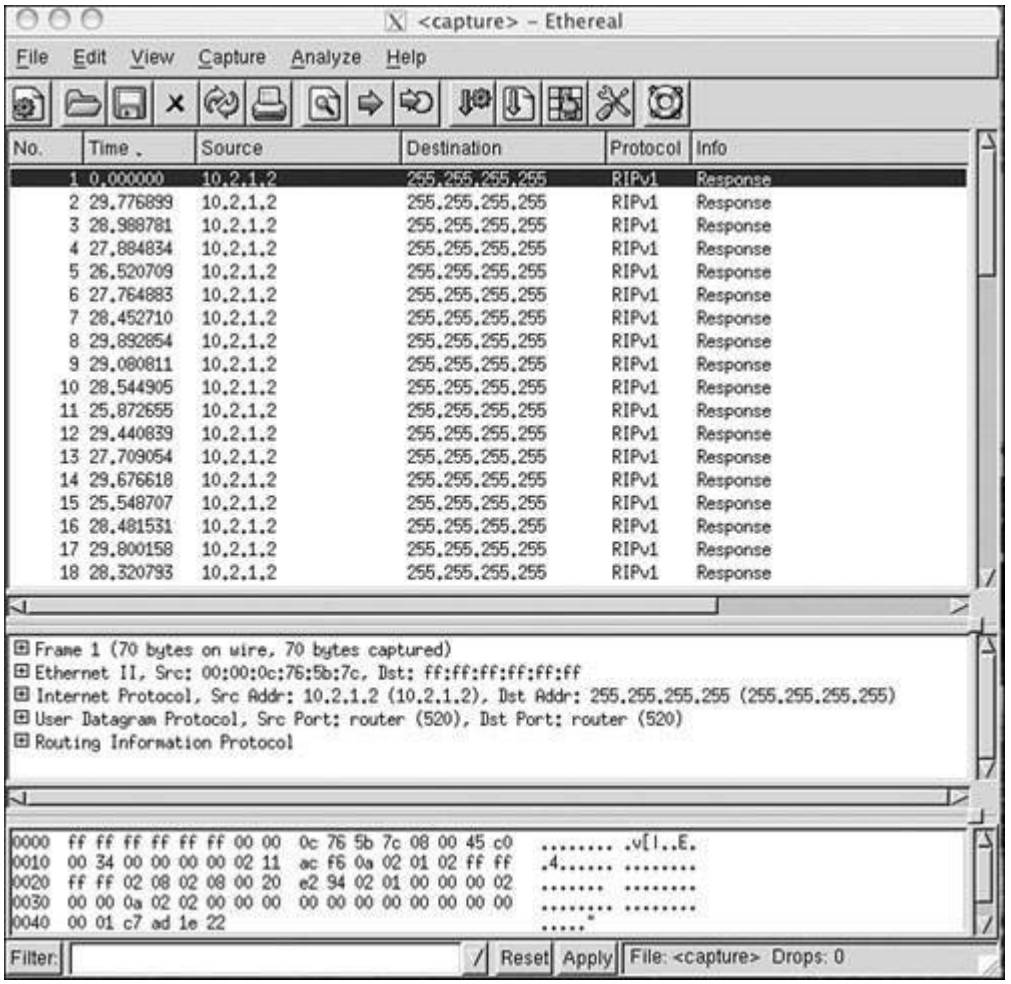
After startup, the router gratuitously sends a Response message out every RIP-enabled interface every 30 seconds, on average. The Response message, or update, contains the router's full route table with the exception of entries suppressed by the split horizon rule. The update timer initiating this periodic update includes a random variable to prevent table synchronization.[6] As a result, the time between individual updates from a typical RIP process might be from 25 to 35 seconds. The specific random variable used by Cisco IOS, RIP_JITTER, subtracts up to 15 percent (4.5 seconds) from the update time. Therefore, updates from Cisco routers vary between 25.5 and 30 seconds (Figure 5-1). The destination address of the update is the all-hosts broadcast 255.255.255.255.[7]

[6] Synchronization of route tables is discussed in Chapter 4, "Dynamic Routing Protocols."

[7] Some implementations of RIP might broadcast only on broadcast media and send updates to the directly connected neighbor on point-to-point links. The Cisco RIP will broadcast on any link type unless configured to do otherwise.

Figure 5-1. RIP adds a small random variable to the update timer at each reset to help avoid route table synchronization. The RIP updates from Cisco routers vary from 25.5 to 30 seconds, as shown in the delta times of these updates.

[View full size image]




Several other timers are employed by RIP. Recall from Chapter 4 the invalidation timer, which distance vector protocols use to limit the amount of time a route can stay in a route table without being updated. RIP calls this timer the *expiration* timer, or *timeout*. The Cisco IOS calls it the *invalid timer*. The expiration timer is initialized to 180 seconds whenever a new route is established and is reset to the initial value whenever an update is heard for that route. If an update for a route is not heard within that 180 seconds (six update periods), the hop count for the route is changed to 16, marking the route as unreachable.

Another timer, the *garbage collection* or *flush* timer, is set to 240 seconds60 seconds longer than the expiration time.[8] The route will be advertised with the unreachable metric until the garbage collection timer expires, at which time the route will be removed from the route table. Example 5-1 shows a route table in which a route has been marked as unreachable but has not yet been flushed.

[8] Cisco routers use a 60-second garbage collection timer, although RFC 1058 prescribes 120 seconds.

## Example 5-1. This router has not heard an update for subnet 10.3.0.0 for more than six update periods. The route has been marked unreachable but has not yet been flushed from the route table.

```
Mayberry#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
     10.0.0.0 255.255.0.0 is subnetted, 4 subnets
```

```
C     10.2.0.0 is directly connected, Serial0
R     10.3.0.0 255.255.0.0 is possibly down,
        routing via 10.1.1.1, Ethernet0
C     10.1.0.0 is directly connected, Ethernet0
R     10.4.0.0 [120/1] via 10.2.2.2, 00:00:00, Serial0
Mayberry#
```

The third timer is the holddown timer, although RFC 1058 does not call for the use of holddowns. The Cisco implementation of RIP does use them. An update with a hop count higher than the metric recorded in the route table will cause the route to go into holddown for 180 seconds (again, six update periods).

These four timers can be manipulated with the command:

**timers basic** *update invalid holddown flush*

This command applies to the entire RIP process. If the timing of one router is changed, the timing of all the routers in the RIP domain must be changed. Therefore, these timers should not be changed from their default values without a specific, carefully considered reason.

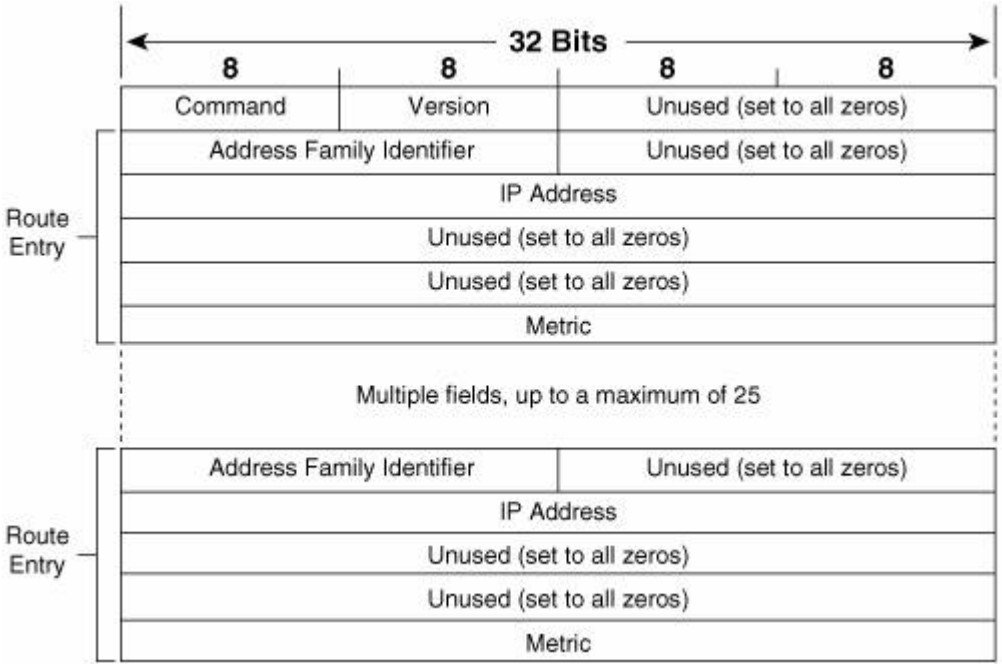
RIP employs split horizon with poison reverse and triggered updates. A triggered update occurs whenever the metric for a route is changed and, unlike regularly scheduled updates, might include only the entry or entries that changed. Also unlike regular updates, a triggered update does not cause the receiving router to reset its update timer; if it did, a topology change could cause many routers to reset at the same time and thus cause the periodic updates to become synchronized. To avoid a "storm" of triggered updates after a topology change, another timer is employed. When a triggered update is transmitted, this timer is randomly set between one and five seconds; subsequent triggered updates cannot be sent until the timer expires.

Some hosts might employ RIP in a "silent" mode. These so-called *silent hosts* do not generate RIP updates, but listen for them and update their internal route tables accordingly. As an example, using *routed* with the *-q* option enables RIP in silent mode on a UNIX host.

## RIP Message Format

The RIP message format is shown in Figure 5-2. Each message contains a command and a version number and can contain entries for up to 25 routes. Each route entry includes an address family identifier, the IP address reachable by the route, and the hop count for the route. If a router must send an update with more than 25 entries, multiple RIP messages must be produced. Note that the initial portion of the message is 4 octets, and each route entry is 20 octets. Therefore, the maximum message size is 4 + (25 x 20) = 504 octets. Including an eight-byte UDP header will make the maximum RIP datagram size (not including the IP header) 512 octets.

Figure 5-2. The RIP message format.




*Command* will always be set to either one, signifying a Request message, or two, signifying a Response message. There are other commands, but they are all either obsolete or reserved for private use.

*Version* will be set to one for RIPv1.

*Address Family Identifier* is set to two for IP. The only exception to this is a request for a router's (or host's) full route table, as discussed in the following section.

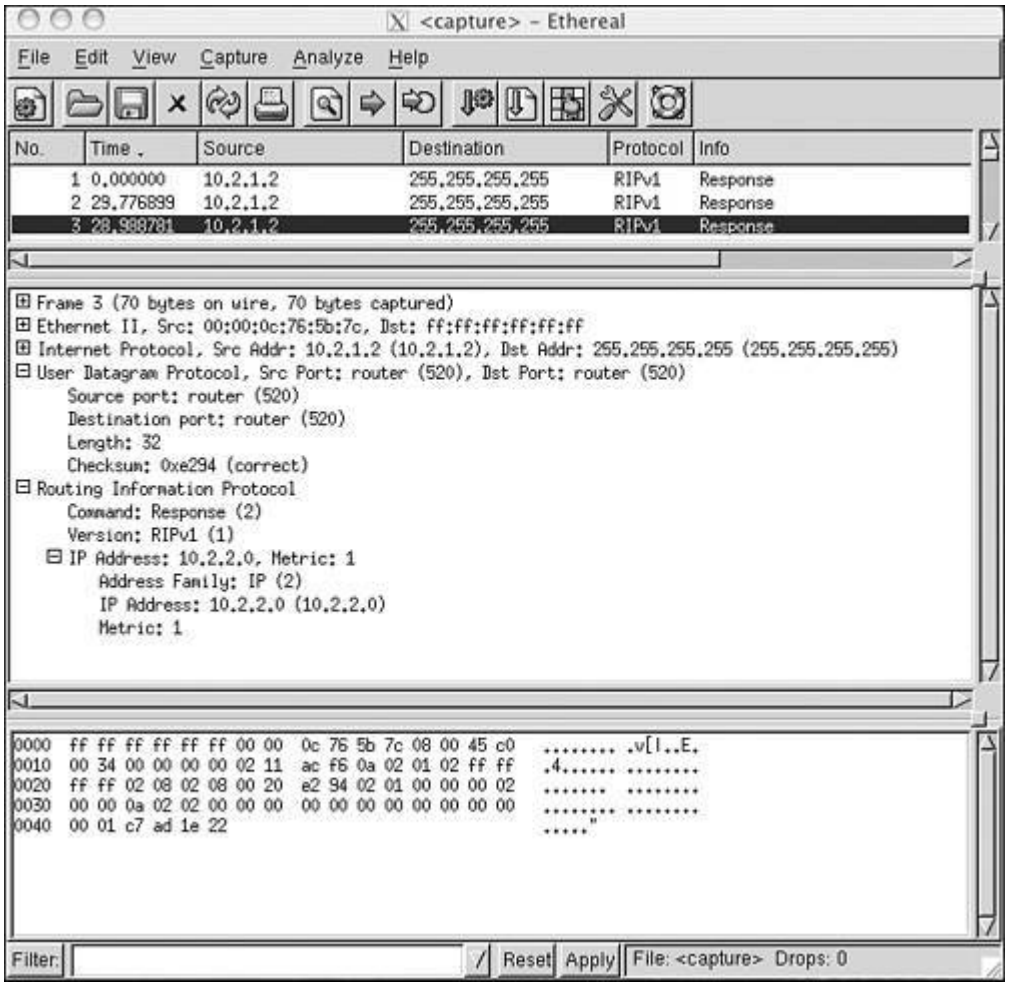
*IP Address* is the address of the destination of the route. This entry might be a major network address, a subnet, or a host route. The section titled "Classful Routing" examines how RIP distinguishes among these three types of entries.

*Metric* is, as previously mentioned, a hop count between 1 and 16.

An analyzer decode of a RIP message is shown in Figure 5-3.

Figure 5-3. An Ethereal decode of a RIPv1 message shows the values in the message fields of this specific Response message.

[View full size image]




Several historical influences contributed to the inelegant format of the RIP message in which far more bit spaces are unused than are used. These influences range from RIP's original development as an XNS protocol, and the developer's intentions for it to adapt to a large set of address families, to the influence of BSD, and its use of socket addresses, to the need for fields to fall on 32-bit word boundaries. Whatever the original motivations, you will see in Chapter 6 that these unused fields have since been put to good use.

## Request Message Types

A RIP Request message might request either a full route table or information on specific routes only. In the former case, the Request message will have a single route entry in which the address family identifier is set to zero, the address is all zeros (0.0.0.0), and the metric is 16. A device receiving such a request responds by unicasting its full route table to the requesting address, honoring such rules as split horizon and boundary summarization (discussed in "Classful Routing: Summarization at Boundary Routers," later in this chapter).

Some diagnostic processes might need to know information about a specific route or routes. In this case, a Request message might be sent with entries specifying the addresses in question. A device receiving this request will process the entries one by one, building a Response message from the Request message. If the device has an entry in its route table corresponding to an address in the request, it will enter the metric of its own route entry into the metric field. If not, the metric field will be set to 16. The response will tell exactly what the router "knows," with no consideration given to split horizon or boundary summarization.

As noted previously, hosts might run RIP in silent mode. This approach allows them to keep their route tables up to date by listening to RIP updates from routers without having to send RIP Response messages uselessly on the network. However, diagnostic processes might need to

examine the route table of these silent hosts. Therefore, RFC 1058 specifies that if a silent host receives a request from a UDP port other than the standard RIP port of 520, the host must send a response.

## Classful Routing

The route table in Example 5-2 contains RIP-derived routes, which are recognized from the key to the left of each entry. Of significance in these entries are the bracketed tuples; as discussed in Chapter 3, "Static Routing," the first number is the administrative distance, and the second number is the metric. It is readily seen that RIP has an administrative distance of 120, and as already stated, the metric for RIP is hop count. Therefore, network 10.8.0.0 is two hops away, via either E0 or S1. If more than one route exists to the same destination with equal hop counts, equal-cost load balancing will be performed. The route table of Example 5-2 contains several multiple, equal-cost routes.

### Example 5-2. This route table contains subnets of networks 10.0.0.0 and 172.25.0.0. All networks not directly connected were derived by RIP.

```
MtPilate#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
    10.0.0.0 255.255.0.0 is subnetted, 9 subnets
R    10.10.0.0 [120/3] via 10.5.5.1, 00:00:20, Serial1
               [120/3] via 10.1.1.1, 00:00:21, Ethernet0
R    10.11.0.0 [120/3] via 10.5.5.1, 00:00:21, Serial1
               [120/3] via 10.1.1.1, 00:00:21, Ethernet0
R    10.8.0.0 [120/2] via 10.1.1.1, 00:00:21, Ethernet0
              [120/2] via 10.5.5.1, 00:00:21, Serial1
R    10.9.0.0 [120/2] via 10.5.5.1, 00:00:21, Serial1
              [120/2] via 10.1.1.1, 00:00:21, Ethernet0
R    10.3.0.0 [120/1] via 10.1.1.1, 00:00:21, Ethernet0
              [120/1] via 10.5.5.1, 00:00:21, Serial1
C    10.1.0.0 is directly connected, Ethernet0
R    10.6.0.0 [120/1] via 10.1.1.1, 00:00:21, Ethernet0
              [120/1] via 10.5.5.1, 00:00:22, Serial1
R    10.7.0.0 [120/2] via 10.1.1.1, 00:00:22, Ethernet0
              [120/2] via 10.5.5.1, 00:00:22, Serial1
C    10.5.0.0 is directly connected, Serial1
   172.25.0.0 255.255.255.0 is subnetted, 3 subnets
R    172.25.153.0 [120/1] via 172.25.15.2, 00:00:03, Serial0
R    172.25.131.0 [120/1] via 172.25.15.2, 00:00:03, Serial0
C    172.25.15.0 is directly connected, Serial0
```

When a packet enters a RIP-speaking router and a route table lookup is performed, the various choices in the table are pruned until a single path remains. First, the network portion of the destination address is read and the route table is consulted for a match. It is this first step of reading the major class A, B, or C network number that defines a classful route table lookup. If there is no match for the major network, the packet is dropped and an ICMP Destination Unreachable message is sent to the packet's source. If there is a match for the network portion, the subnets listed for that network are examined. If a match can be found, the packet is routed. If a match cannot be made, the packet is dropped and a Destination Unreachable message is sent.

## Classful Routing: Directly Connected Subnets

Classful route lookups can be illustrated with three examples (referring to Example 5-2):

1. If a packet with a destination address of 192.168.35.3 enters this router, no match for network 192.168.35.0 is found in the route table, and the packet is dropped.
2. If a packet with a destination address of 172.25.33.89 enters the router, a match is made to class B network 172.25.0.0/24. The subnets listed for this network are then examined; no match can be made for subnet 172.25.33.0, so that packet, too, is dropped.
3. Finally, a packet destined for 172.25.153.220 enters the router. This time 172.25.0.0/24 is matched, then subnet 172.25.153.0 is matched, and the packet is forwarded to next-hop address 172.25.15.2.

Another look at Figure 5-2 reveals that there is no provision for RIP to advertise a subnet mask along with each route entry. And accordingly, no masks are associated with the individual subnets in the route table. Therefore, if the router whose forwarding database is depicted in Example 5-2 receives a packet with a destination address of 172.25.131.23, there is no positive way to determine where the subnet bits end and the host bits begin, or even if the address is subnetted at all.

The router's only recourse is to assume that the mask configured on one of its interfaces attached to 172.25.0.0 is used consistently throughout the network. It will use its own mask for 172.25.0.0 to derive the subnet of the destination address. As the route tables throughout this chapter illustrate, a router that is directly connected to a network will list the network in a heading along with the subnet mask of the connecting interface and will then list all the known subnets of the network. If the network is not directly connected, there is a listing only for the major-class network and no associated mask.

Because the destination addresses of packets being routed by a classful routing protocol are interpreted according to the subnet masks locally configured on the router's interfaces, all subnet masks within a major, class-level network must be consistent.

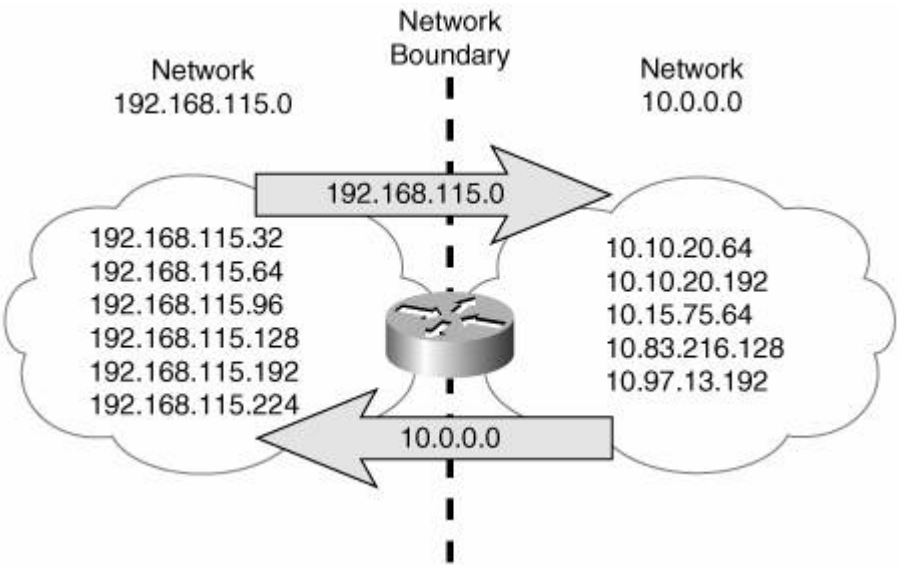
## Classful Routing: Summarization at Boundary Routers

A question arises from the preceding discussion: How does a RIP process interpret the subnet of a major network if it has no interfaces attached to that network? Without an interface on the class A, B, or C network of the destination, the router has no way of recognizing the correct subnet mask to use and therefore no way of correctly identifying the subnet.

The solution is simple: If a router has no direct attachments to the network, it needs only a single route entry pointing toward a router that is directly attached.

Figure 5-4 shows a router that is attached at the boundary of two major networks, the class A network 10.0.0.0 and the class C network 192.168.115.0. This *boundary router* does not send details of the subnets of one major network into the other major network. As the illustration shows, it automatically performs summarization, or *subnet hiding*. It advertises only the address 10.0.0.0 into network 192.168.115.0 and advertises only the address.

Figure 5-4. This router, at the boundary of two major networks, does not advertise the subnets of one network to routers within the other network.




In this way, the route tables for routers within network 192.168.115.0 have only a single entry that directs packets for 10.0.0.0 toward the boundary router. The boundary router has an interface directly on network 10.0.0.0 and, therefore, has a subnet mask with which to derive the subnet for routing a packet within that network's "cloud." Example 5-3 shows what the route table of a router within 192.168.115.0 would look like with a single, subnetless entry for 10.0.0.0.

### Example 5-3. This router has a single entry pointing toward network 10.0.0.0. The next-hop address is the boundary router, because the network is recorded as being one hop away.

```
Raleigh#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
R    10.0.0.0 [120/1] via 192.168.115.40, 00:00:10, Ethernet1
     192.168.115.0 255.255.255.240 is subnetted, 6 subnets
C       192.168.115.32 is directly connected, Ethernet1
R       192.168.115.64 [120/1] via 192.168.115.99, 00:00:13, Ethernet0
C       192.168.115.96 is directly connected, Ethernet0
C       192.168.115.128 is directly connected, Serial0
R       192.168.115.192 [120/1] via 192.168.115.99, 00:00:13, Ethernet0
R       192.168.115.224 [120/1] via 192.168.115.130, 00:00:25, Serial0
Raleigh#
```

Chapter 3's brief discussion of discontiguous subnetssubnets of a major network address separated by a different major networknotes that they present a problem for classful routing protocols such as RIP and IGRP. The problem occurs when discontiguous subnets are automatically summarized at network boundaries. A case study in the configuration section of this chapter demonstrates the problem and a solution.

## Classful Routing: Summary

The defining characteristic of a classful routing protocol is that it does not advertise an address

mask along with the advertised destination address. Therefore, a classful routing protocol must first match the major class A, B, or C network portion of a destination address. For every packet passing through the router

1. If the destination address is a member of a directly connected major network, the subnet mask configured on the interface attached to that network will be used to determine the subnet of the destination address. Therefore, the same subnet mask must be used consistently throughout that major network.
2. If the destination address is not a member of a directly connected major network, the router will try to match only the major class A, B, or C portion of the destination address.

PREV NEXT

# Configuring RIP

In keeping with the simple nature of RIP, configuration is an easy task. There is one command to the RIP process and one command for each network on which RIP is to run. Beyond that, RIP has configuration options.

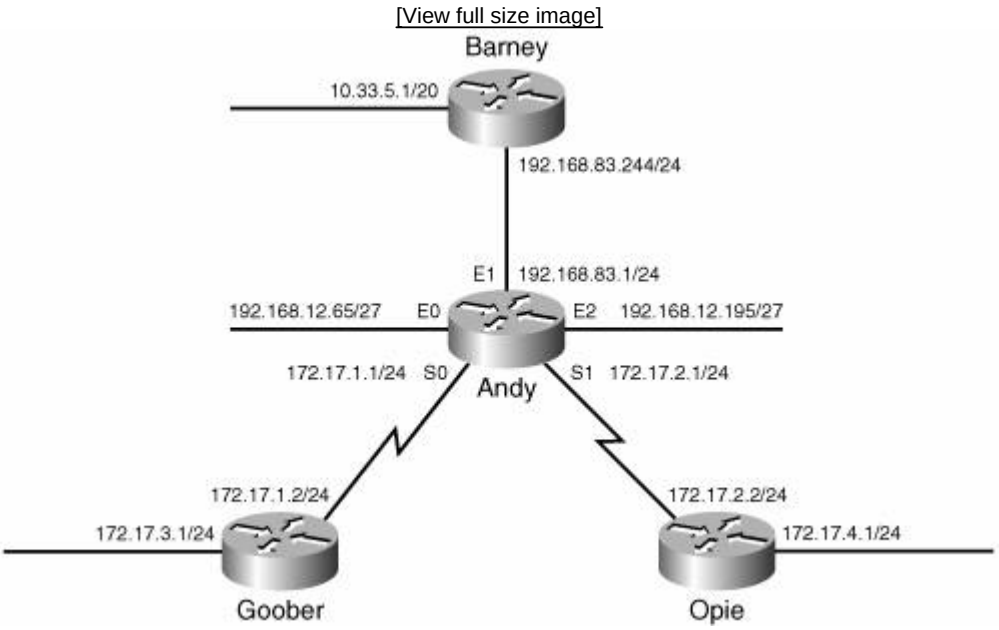
## Case Study: A Basic RIP Configuration

Only two steps are necessary to configure RIP:

1. Enable RIP with the command `router rip`.

2. Specify each major network on which to run RIP with the `network` command. Figure 5-5 show four-router network, with four major network numbers. Router Goober is attached to two subn of network 172.17.0.0.

Figure 5-5. Both Andy and Barney are border routers between class-l networks.




The commands necessary to enable RIP are displayed in Example 5-4.

Example 5-4. The RIP configuration of Goober.

```
router rip
 network 172.17.0.0
```

Similarly, Opie has two subnets of the same network and will configure with the commands in Exa 5-5.

## Example 5-5. Opie's RIP configuration.

```
router rip
 network 172.17.0.0
```

Executing any router command puts the router into config-router mode, indicated in the prompt classful nature of RIP and the subnet hiding at network boundaries mean that no subnets can be specified with the network commandonly major class A, B, or C network addresses. RIP can run interface configured with any address belonging to the network specified with the network comm

Barney is attached to two networks10.0.0.0 and 192.168.83.0. Therefore, both networks must be specified as in Example 5-6.

## Example 5-6. Barney's RIP configuration.

```
router rip
 network 10.0.0.0
 network 192.168.83.0
```

Andy has one attachment to network 192.168.83.0, attachments to two subnets of 192.168.12.0, attachments to two subnets of 172.17.0.0. Example 5-7 shows its configuration.

## Example 5-7. Andy's RIP configuration.

```
router rip
 network 172.17.0.0
 network 192.168.12.0
 network 192.168.83.0
```

In Example 5-8, the command debug ip rip has been turned on in Andy. Of particular interest he the subnet hiding that the router is performing. The subnets 192.168.12.64 and 192.168.12.192 a advertised between interfaces E0 and E2, which are both attached to network 192.168.12.0, but t network is summarized out E1, S0, and S1, which are all attached to different networks. Likewise, networks 192.168.83.0 and 172.17.0.0 are being summarized across classful boundaries. Notice a that Andy is receiving a summary route for network 10.0.0.0 from Barney. Finally, split horizon ca observed here. For example, the advertisement to Barney out E1 contains no entries for 10.0.0.0 192.168.83.0.

## Example 5-8. These debug messages show the RIP updates received an sent by Router Andy. The results of both network summarization and sp horizon can be observed in the update entries.

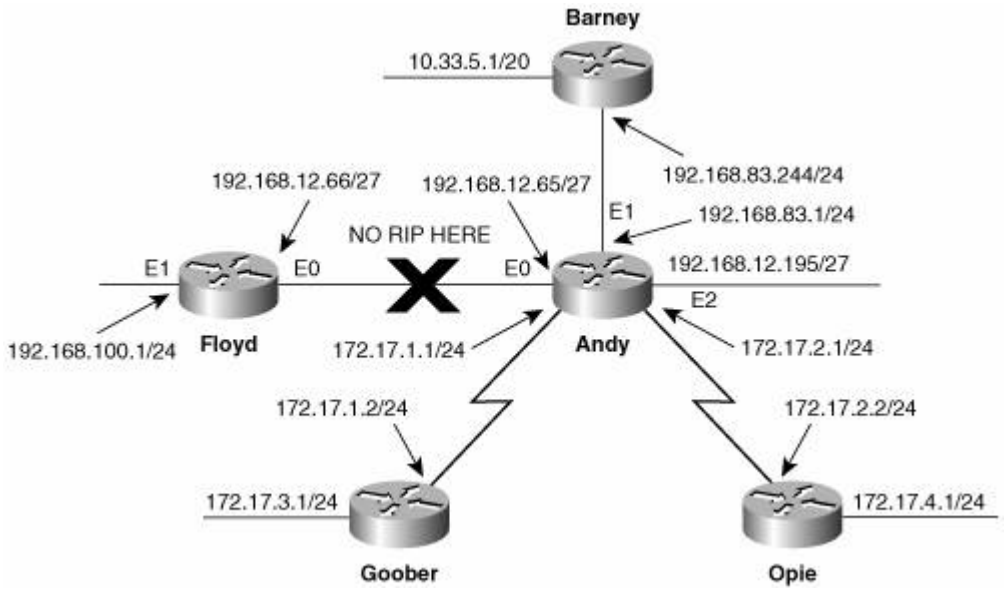
```
Andy#debug ip rip
RIP protocol debugging is on
Andy#
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (192.168.12.65)
RIP: build update entries
     subnet 192.168.12.192, metric 1
     network 10.0.0.0, metric 2
```

```
     network 192.168.83.0, metric 1
     network 172.17.0.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Ethernet1 (192.168.83.1)
RIP: build update entries
     network 192.168.12.0, metric 1
     network 172.17.0.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Ethernet2 (192.168.12.195)
RIP: build update entries
     subnet 192.168.12.64, metric 1
     network 10.0.0.0, metric 2
     network 192.168.83.0, metric 1
     network 172.17.0.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Serial0 (172.17.1.1)
RIP: build update entries
     subnet 172.17.4.0, metric 2
     subnet 172.17.2.0, metric 1
     network 10.0.0.0, metric 2
     network 192.168.83.0, metric 1
     network 192.168.12.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Serial1 (172.17.2.1)
RIP: build update entries
     subnet 172.17.1.0, metric 1
     subnet 172.17.3.0, metric 2
     network 10.0.0.0, metric 2
     network 192.168.83.0, metric 1
     network 192.168.12.0, metric 1
RIP: received v1 update from 172.17.1.2 on Serial0
     172.17.3.0 in 1 hops
RIP: received v1 update from 192.168.83.244 on Ethernet1
     10.0.0.0 in 1 hops
RIP: received v1 update from 172.17.2.2 on Serial1
     172.17.4.0 in 1 hops
```

## Case Study: Passive Interfaces

The Router Floyd has been added to the network (Figure 5-6).

Figure 5-6. Network policy calls for no RIP exchanges between Andy a
Floyd.




It is desired that no RIP advertisements be exchanged between Floyd and Andy. This is easy enou
Floyd, as displayed in Example 5-9.

### Example 5-9. Floyd's RIP configuration

```
router rip
 network 192.168.100.0
```

By not including a network statement for 192.168.12.0, Floyd will not advertise on interface 192.168.12.66. Andy, however, has two interfaces attached to 192.168.12.0; the network must b included under RIP. To block RIP broadcasts on an interface connected to a subnet of a RIP-enable network, add the passive-interface command to the RIP process. Andy's RIP configuration is dis in Example 5-10.

### Example 5-10. Andy's RIP configuration with a passive interface.

```
router rip
 passive-interface Ethernet0
 network 172.17.0.0
 network 192.168.12.0
 network 192.168.83.0
```

passive-interface is not a RIP-specific command; it might be configured under any IP routing pr Using the passive-interface command essentially makes a router a silent host on the data link specified. Like other silent hosts, it still listens to RIP broadcasts on the link and updates its route accordingly. If the desired result is to prevent the router from learning routes on the link, it must l achieved by more intricate control of routing updates, namely by filtering out updates. (Route filte discussed in Chapter 13, "Route Filtering.") Unlike a silent host, the router does not respond to a F Request received on a passive interface.

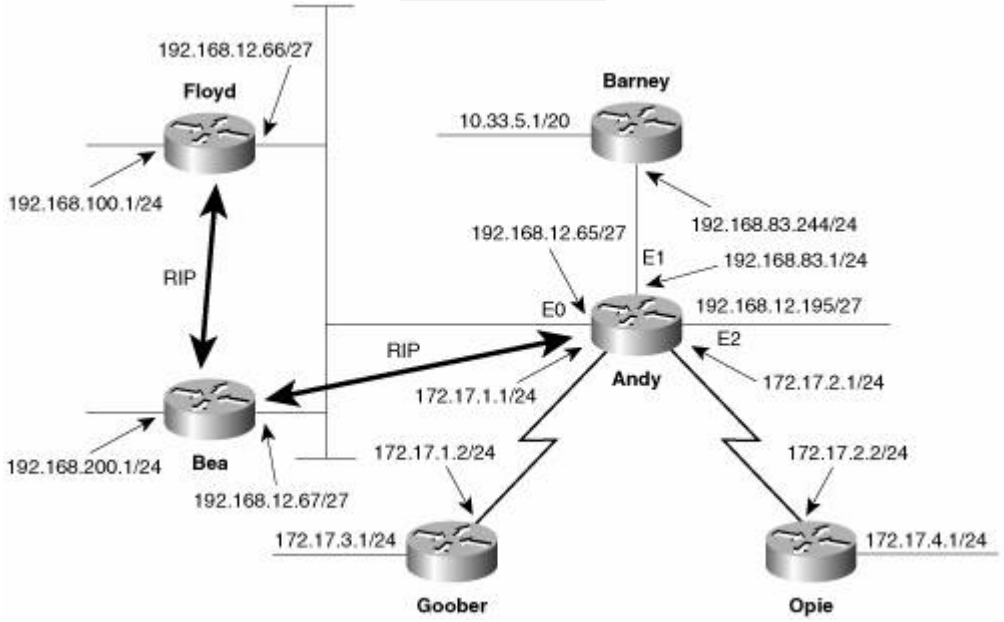
## Case Study: Configuring Unicast Updates

Next, Router Bea is added to the Ethernet link that Andy and Floyd share (Figure 5-7). The no-RIP

policy between Andy and Floyd remains in place, but now Bea and Andy, like Bea and Floyd, must exchange RIP advertisements.

Figure 5-7. No RIP updates should be exchanged between Andy and Fl but both should exchange updates with Bea.

[View full size image]




The configuration of Bea is straightforward (see Example 5-11).

## Example 5-11. Bea's RIP configuration.

```
router rip
 network 192.168.12.0
 network 192.168.200.0
```

The addition of a neighbor command under the RIP processes of Andy enables RIP to send a unic advertisement to Bea's interface while the passive-interface command continues to prevent bro updates on the link.[9]

[9] Another use of **neighbor** is to enable unicast updates on nonbroadcast media such as Frame Relay.

Andy's configuration is displayed in Example 5-12.

## Example 5-12. Andy's RIP configuration includes a passive interface wit neighbor statement to enable unicast updates.

```
router rip
 passive-interface Ethernet0
 network 172.17.0.0
 network 192.168.12.0
 network 192.168.83.0
 neighbor 192.168.12.67
```

Because Floyd must now send advertisements to Bea, a network command for 192.168.12.0 must added. passive-interface is also added to prevent broadcast updates, and a neighbor command added to enable unicast updates to Bea (see Example 5-13).

## Example 5-13. Floyd's RIP configuration with unicast updates to neighb 192.168.12.67.

```
router rip
passive-interface Ethernet0
network 192.168.12.0
network 192.168.100.0
neighbor 192.168.12.67
```

By enabling debug ip rip events at Andy, the results of the new configuration can be verified (Ex 5-14). Andy is receiving updates from Bea, but not from Floyd, and is sending updates directly to interface but is not broadcasting on its E0.

## Example 5-14. The only updates Andy is sending out interface E0 are unicasts to Bea. Updates are being received from Bea but not from Floyd

```
Andy#debug ip rip events
RIP event debugging is on
Andy#
RIP: received v1 update from 192.168.12.67 on Ethernet0
RIP: Update contains 1 routes
RIP: sending v1 update to 255.255.255.255 via Ethernet1 (192.168.83.1)
RIP: Update contains 4 routes
RIP: Update queued
RIP: Update sent via Ethernet1
RIP: sending v1 update to 255.255.255.255 via Ethernet2 (192.168.12.195)
RIP: Update contains 6 routes
RIP: Update queued
RIP: Update sent via Ethernet2
RIP: sending v1 update to 255.255.255.255 via Serial0 (172.17.1.1)
RIP: Update contains 7 routes
RIP: Update queued
RIP: Update sent via Serial0
RIP: sending v1 update to 255.255.255.255 via Serial1 (172.17.2.1)
RIP: Update contains 7 routes
RIP: Update queued
RIP: Update sent via Serial1
RIP: sending v1 update to 192.168.12.67 via Ethernet0 (192.168.12.65)
RIP: Update contains 4 routes
RIP: Update queued
RIP: Update sent via Ethernet0
RIP: received v1 update from 172.17.1.2 on Serial0
RIP: Update contains 1 routes
RIP: received v1 update from 172.17.2.2 on Serial1
RIP: Update contains 1 routes
RIP: received v1 update from 192.168.12.67 on Ethernet0
RIP: Update contains 1 routes
```

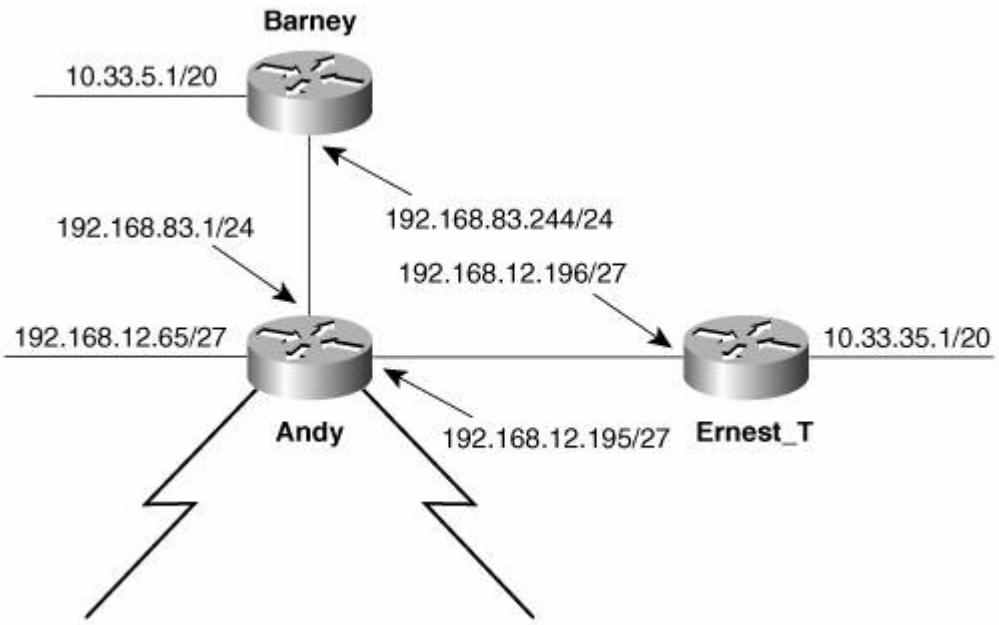
Although Bea has learned routes from both Andy and from Floyd, and is broadcasting updates on shared Ethernet, the policy still works because split horizon prevents Bea from advertising the rou

learned from those two routers back onto the Ethernet.

## Case Study: Discontiguous Subnets

In Figure 5-8, another router has been added to the network with a subnet 10.33.32.0/20 on its E interface. The problem is that the other subnet of network 10.0.0.0, 10.33.0.0/20, is connected to Barney, and the only route between the subnets is via 192.168.83.0 and 192.168.12.0two entirely different networks. As a result, network 10.0.0.0 is discontiguous.

Figure 5-8. Classful protocols such as RIP and IGRP cannot route a topo in which the subnets of network 10.0.0.0 are separated by different networks.




Barney will consider itself a border router between network 10.0.0.0 and network 192.168.83.0; likewise, Ernest_T will consider itself a border router between 10.0.0.0 and 192.168.12.0. Both wi advertise a summary route of 10.0.0.0, and as a result Andy will be "fooled into thinking" that it h equal-cost paths to the same network. Andy will load share on the links to Barney and Ernest_T, a there is now only a 50-50 chance that packets to network 10.0.0.0 will reach the correct subnet.

The solution is to configure subnets of network 10.0.0.0 on the same links on which 192.168.83.0 and 192.168.12.192/27 reside. This is accomplished with secondary IP addresses, as displayed in Example 5-15, Example 5-16, and Example 5-17.

Example 5-15. Barney is configured with secondary IP addresses.

```
interface e0
 ip address 10.33.55.1 255.255.240.0 secondary
```

Example 5-16. Andy is configured with secondary IP addresses, and a n network is added to RIP.

```
interface e1
 ip address 10.33.55.2 255.255.240.0 secondary
interface e2
 ip address 10.33.75.1 255.255.240.0 secondary
router rip
 network 10.0.0.0
```

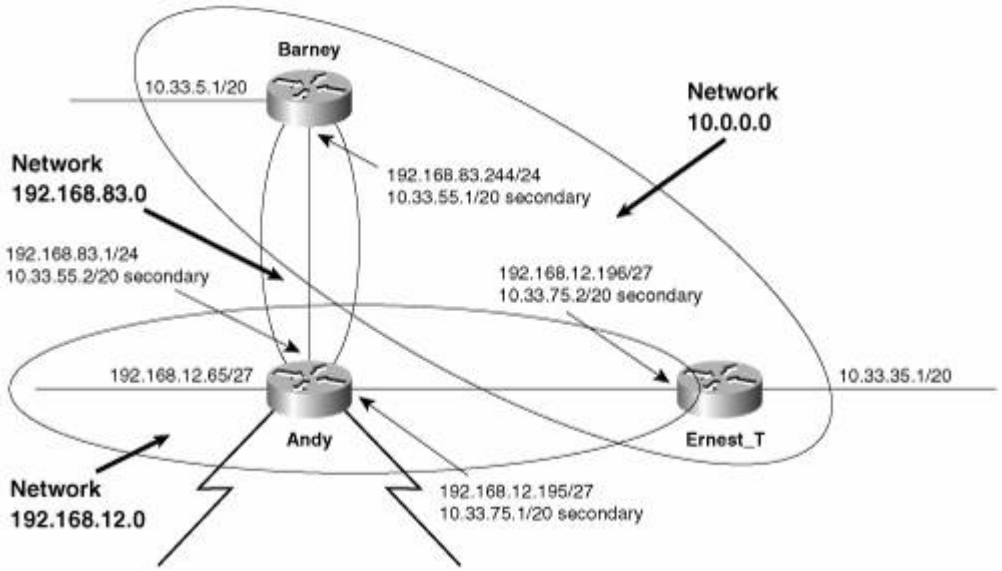
Example 5-17. Ernest_T is configured with secondary IP addresses.

```
interface e0
 ip address 10.33.75.2 255.255.240.0 secondary
```

Because Andy did not previously have an interface on network 10.0.0.0, a network statement is a
to the RIP process. The result of the configuration is shown in Figure 5-9. The existing logical netw
structure remains in place, and a contiguous network 10.0.0.0 is "overlaid" onto it.

Figure 5-9. Secondary addresses are used to connect the subnets o
network 10.0.0.0 across the same links on which other network addres
exist.

[View full size image]




Example 5-18 shows Ernest_T's route table. Of interest here are the dual, equal-cost routes assoc
with next-hop addresses 10.33.75.1 and 192.168.12.195.

Example 5-18. The routing process in this router sees the subnets 192.168.12.192/27 and 10.33.64.0/20 as separate links, although they reside on the same physical interface.

```
Ernest_T#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
```

```
Gateway of last resort is not set
    10.0.0.0 255.255.240.0 is subnetted, 4 subnets
C     10.33.32.0 is directly connected, Ethernet1
R     10.33.48.0 [120/1] via 10.33.75.1, 00:00:05, Ethernet0
R     10.33.0.0 [120/2] via 10.33.75.1, 00:00:05, Ethernet0
C     10.33.64.0 is directly connected, Ethernet0
R     192.168.83.0 [120/1] via 192.168.12.195, 00:00:05, Ethernet0
                   [120/1] via 10.33.75.1, 00:00:05, Ethernet0
      192.168.12.0 255.255.255.224 is subnetted, 2 subnets
R     192.168.12.64 [120/1] via 192.168.12.195, 00:00:05, Ethernet0
C     192.168.12.192 is directly connected, Ethernet0
R     192.168.200.0 [120/2] via 192.168.12.195, 00:00:05, Ethernet0
                    [120/2] via 10.33.75.1, 00:00:05, Ethernet0
R     172.17.0.0 [120/1] via 192.168.12.195, 00:00:06, Ethernet0
                 [120/1] via 10.33.75.1, 00:00:06, Ethernet0
Ernest_T#
```

Because the routing process sees secondary addresses as separate data links, caution should be u when planning them on RIP or IGRP networks. A separate RIP update will be broadcast on each su if the updates are large and the bandwidth of the physical link is limited (such as on serial links), t multiple updates can cause congestion. Multiple updates on a link configured with secondary addre can be observed in Example 5-21, later in this chapter.

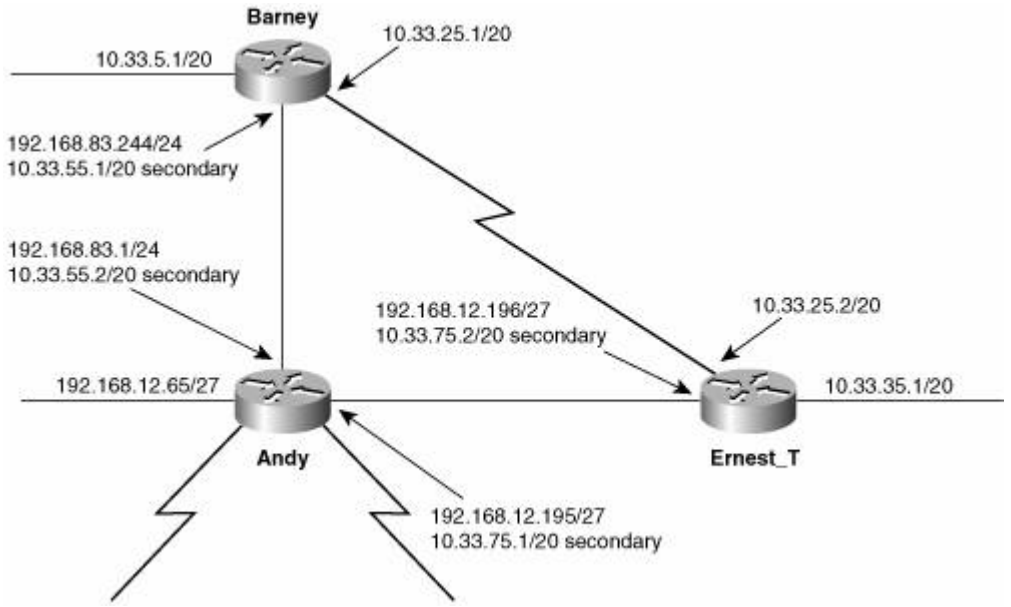
Type cautiously when entering secondary addresses. If you omit the keyword `secondary`, the rou assume that the primary address is to be replaced with the new address. Making this mistake on a service interface will have serious consequences.

## Case Study: Manipulating RIP Metrics

A serial link, to be used as a backup, has been added between Ernest_T and Barney (Figure 5-10) link should be used only if the route via Andy fails. The problem is that the path between Barney's 10.33.0.0 subnet and Ernest_T's 10.33.32.0 subnet is one hop via the serial link and two hops via preferred Ethernet links. Under normal circumstances, RIP will choose the serial link.

Figure 5-10. RIP metrics must be manipulated so that the two-hop Ethe route between Barney and Ernest_T will be preferred over the one-ho serial route.

[View full size image]




The route metrics can be manipulated with the offset-list command. The command specifies a nu to add to the metric of a route entry and references an access list[10] to determine which route en to modify. The syntax of the command is as follows:

[10] See Appendix B for a tutorial on access lists.

**offset-list** {*access-list-number* | *name*} { **in** | **out**} *offset* [*type number*]

The configuration of Ernest_T is in Example 5-19.

## Example 5-19. Ernest_T's RIP configuration with an inbound offset list.

```
access-list 1 permit 10.33.0.0 0.0.0.0
router rip
 network 192.168.12.0
 network 10.0.0.0
 offset-list 1 in 2 Serial0
```

An access list is written that identifies the route to subnet 10.33.0.0. The syntax of the offset list s "Examine RIP advertisements incoming from interface S0. For route entries matching the addresse specified in access list 1, add 2 hops to the metric."

After Barney is configured, it will have the entries in Example 5-20 in its configuration file.

## Example 5-20. Barney's RIP configuration with an inbound offset list.

```
router rip
 offset-list 5 in 2 Serial0
 network 10.0.0.0
 network 192.168.83.0
!
access-list 5 permit 10.33.32.0 0.0.0.0
```

Example 5-21 shows the results of the configuration at Ernest_T.

Example 5-21. The addition of the hops specified in the offset list chang the hop count to subnet 10.33.0.0/20 via S0 from one to three. Now the two-hop route via E0 is used.

```
Ernest_T#debug ip rip
RIP protocol debugging is on
Ernest_T#
RIP: received v1 update from 192.168.12.195 on Ethernet0
     192.168.12.64 in 1 hops
     10.0.0.0 in 1 hops
     192.168.83.0 in 1 hops
     192.168.200.0 in 2 hops
     172.17.0.0 in 1 hops
RIP: received v1 update from 10.33.75.1 on Ethernet0
     10.33.48.0 in 1 hops
     10.33.0.0 in 2 hops
     192.168.83.0 in 1 hops
     192.168.12.0 in 1 hops
     192.168.200.0 in 2 hops
     172.17.0.0 in 1 hops
RIP: received v1 update from 10.33.25.1 on Serial0
     10.33.32.0 in 3 hops
     10.33.48.0 in 1 hops
     10.33.0.0 in 3 hops
     192.168.83.0 in 1 hops
     192.168.200.0 in 3 hops
     172.17.0.0 in 2 hops
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (192.168.12.196)
RIP: build update entries
     network 10.0.0.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (10.33.75.2)
RIP: build update entries
     subnet 10.33.32.0, metric 1
     subnet 10.33.16.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Ethernet1 (10.33.35.1)
RIP: build update entries
     subnet 10.33.48.0, metric 2
     subnet 10.33.0.0, metric 3
     subnet 10.33.16.0, metric 1
     subnet 10.33.64.0, metric 1
     network 192.168.83.0, metric 2
     network 192.168.12.0, metric 1
     network 192.168.200.0, metric 3
     network 172.17.0.0, metric 2
RIP: sending v1 update to 255.255.255.255 via Serial0 (10.33.25.2)
RIP: build update entries
     subnet 10.33.32.0, metric 3
     subnet 10.33.0.0, metric 3
     subnet 10.33.64.0, metric 1
     network 192.168.12.0, metric 1
     network 192.168.200.0, metric 3
     network 172.17.0.0, metric 2
```

Alternatively, instead of having the two routers modify incoming routes, the routers can be configu modify their outgoing routes. The configurations in Example 5-22 and Example 5-23 will have the effects as the previous configurations.

Example 5-22. Ernest_T's RIP configuration uses outbound offset lists.

```
router rip
 offset-list 3 out 2 Serial0
 network 192.168.12.0
 network 10.0.0.0
!
access-list 3 permit 10.33.32.0 0.0.0.0
```

Example 5-23. Barney's RIP configuration uses outbound offset lists.

```
router rip
 offset-list 7 out 2 Serial0
 network 10.0.0.0
 network 192.168.83.0
!
access-list 7 permit 10.33.0.0 0.0.0.0
```

Several other options are available for configuring offset lists. If no interface is identified, the list v modify all incoming or outgoing updates specified by the access list on any interface. If no access called (by using a zero as the access list number), the offset list will modify all incoming or outgoi updates.

Caution should be applied when choosing whether to use offset lists on incoming or outgoing advertisements. If more than two routers are attached to a broadcast network, consideration mus given to whether a single router should broadcast a modified advertisement to all its neighbors or whether a single router should modify a received advertisement.

Care must also be exercised when implementing offset lists on routes that are in use. When an off causes a next-hop router to advertise a higher metric than it had been advertising, the route will b marked unreachable until the holddown timer expires.

## Case Study: Minimizing the Impact of Updates

There are times when you will want to minimize network traffic caused by routing updates. Regula updates, by default, occur every 30 seconds and include the full route table, in addition to the flas updates that are transmitted when there is a change in the metric of route entries. In a network w many subnets, routing updates can affect traffic, especially on low bandwidth links. You also migh to minimize routing updates if you are paying for traffic traversing a link.

Suppose utilization on the new serial link from Barney to Ernest_T, in Figure 5-10, is high. The RIP routing traffic can be reduced in two ways to minimize the traffic caused by the routing protocol: T timers can be adjusted so the updates don't occur as frequently, but this will cause longer converg time when the primary link fails. Or triggered extensions can be configured to eliminate periodic R updates.

The interface command `ip rip triggered` enables the triggered extensions of RIP.[11] After two ro on a serial link determine both are configured for triggered RIP, route table updates are minimized include only the initial exchange of route tables and updates when changes to the route tables occ This command is only available on serial links and must be configured on both ends of the link bef taking affect.

[11] Triggered Extensions are defined in RFC 2091 and were first introduced into IOS in release 12.0(1)T.

The triggered extensions are configured on Barney's serial link to Ernest_T with the configuration i Example 5-24.

## Example 5-24. Barney is configured with triggered, rather then periodic updates.

```
interface serial 0
 ip rip triggered
```

Debugging on Barney shows that Barney attempts to establish a triggered relationship with the ro on the other end of the link. Barney sends polls and waits for acknowledgments. When it doesn't r any acknowledgments, Barney begins sending regular RIPv1 updates. Output from debug ip rip a debug ip rip trigger is shown in Example 5-25.

## Example 5-25. When a router is initially configured with triggered RIP, i polls to determine if its neighbor is also configured for triggered RIP.

```
Barney#debug ip rip
RIP protocol debugging is
Barney#debug ip rip trigger
RIP trigger debugging is on

Barney(config)#interface serial 0
Barney(config-if)#ip rip triggered
* 13:22:10.223: RIP: sending triggered request on Serial0 to 255.255.255.255
* 13:22:10.223: RIP: Start poll timer from 10.33.25.1 on Serial0
* 13:22:15.223: RIP-TIMER: polling timer on Serial0(10.33.25.1) expired
* 13:22:15.223: RIP: sending triggered request on Serial0 to 255.255.255.255
* 13:22:15.223: RIP: Start poll timer from source 10.33.25.1 on Serial0
* 13:22:16.733: RIP: received v1 update from 10.33.25.2 on Serial0
* 13:22:16.733:      172.17.8.0 in 1 hops
* 13:22:16.733:      172.17.9.0 in 1 hops
* 13:22:16.737:      172.17.10.0 in 1 hops
* 13:22:16.737:      172.17.11.0 in 1 hops
* 13:22:19.466: RIP-TIMER: sending timer on Serial0 expired
* 13:22:20.223: RIP-TIMER: polling timer on Serial0(10.33.25.1) expired
* 13:22:20.223: RIP: sending triggered request on Serial0 to 255.255.255.255
* 13:22:20.223: RIP: Start poll timer from source 10.33.25.1 on Serial0
* 13:22:25.223: RIP-TIMER: polling timer on Serial0(10.33.25.1) expired
* 13:22:25.223: RIP: sending triggered request on Serial0 to 255.255.255.255
* 13:22:25.223: RIP: Start poll timer from source 10.33.25.1 on Serial0
* 13:22:30.224: RIP-TIMER: polling timer on Serial0(10.33.25.1) expired
* 13:22:30.224: RIP: sending triggered request on Serial0 to 255.255.255.255
* 13:22:30.224: RIP: Start poll timer from source 10.33.25.1 on Serial0
* 13:22:35.224: RIP-TIMER: polling timer on Serial0(10.33.25.1) expired
* 13:22:35.224: RIP: sending triggered request on Serial0 to 255.255.255.255
* 13:22:35.224: RIP: Start poll timer from source 10.33.25.1 on Serial0
* 13:22:40.224: RIP-TIMER: polling timer on Serial0(10.33.25.1) expired
* 13:22:40.224: RIP: Poll 6 times on Serial0 thru 10.33.25.1 without any ack
* 13:22:46.126: RIP: received v1 update from 10.33.25.2 on Serial0
* 13:22:46.126:      172.17.8.0 in 1 hops
* 13:22:46.126:      172.17.9.0 in 1 hops
* 13:22:46.130:      172.17.10.0 in 1 hops
* 13:22:46.130:      172.17.11.0 in 1 hops
* 13:22:49.054: RIP-TIMER: sending timer on Serial0 expired
* 13:22:49.054: RIP: sending v1 update to 255.255.255.255 via Serial0(10.33.25.1)
* 13:22:49.054: RIP: build update entries
* 13:22:49.054:   network 10.0.0.0 metric 2
* 13:22:49.054:   subnet 172.17.0.0 metric 2
* 13:22:49.054:   subnet 172.17.2.0 metric 1
```

```
* 13:22:49.058:   subnet 172.17.4.0 metric 2
* 13:22:49.058:   network 192.168.12.0 metric 1
* 13:22:49.058:   network 192.168.83.0 metric 1
* 13:23:13.070: RIP: received v1 update from 10.33.25.2 on Serial0
* 13:23:13.070:      172.17.8.0 in 1 hops
* 13:23:13.074:      172.17.9.0 in 1 hops
* 13:23:13.074:      172.17.10.0 in 1 hops
* 13:23:13.074:      172.17.11.0 in 1 hops
* 13:23:17.385: RIP-TIMER: sending timer on Serial0 expired
* 13:23:17.385: RIP: sending v1 update to 255.255.255.255 via Serial0(10.33.25.1)
* 13:23:17.385: RIP: build update entries
* 13:23:17.385:   network 10.0.0.0 metric 2
* 13:23:17.385:   subnet 172.17.0.0 metric 2
* 13:23:17.385:   subnet 172.17.2.0 metric 1
* 13:23:17.389:   subnet 172.17.4.0 metric 2
* 13:23:17.389:   network 192.168.12.0 metric 1
* 13:23:17.389:   network 192.168.83.0 metric 1
```

The debug output shows that the configured router sends six triggered requests. For each one, the router sets a poll timer, which expires in five seconds. If no acknowledgment is received within the seconds, another triggered request is sent. If no acknowledgment is received after six triggered requests have been sent and the polls have expired, the router waits for the next regular update t and broadcasts a RIP update. While Barney is sending triggered requests, Ernest_T continues to broadcast its own RIP updates, which Barney processes.

Now, Ernest_T's serial link to Barney is configured with triggered RIP. Through debugging on Erne Example 5-26 shows the initialization process between Barney and Ernest_T.

## Example 5-26. Debug output shows two routers establishing a triggered relationship.

```
* 13:35:04.612: RIP: received v1 triggered request from 172.17.1.2 on Serial0
* 13:35:04.616: RIP: Trigger rip running on network 172.17.0.0 thru Serial0
* 13:35:04.616: RIP: 172.17.1.2 change state from DOWN to INIT
* 13:35:04.616: RIP: 172.17.1.2 change state from INIT to LOADING
* 13:35:04.616: RIP: send v1 triggered flush update to 172.17.1.2 on Serial0
* 13:35:04.616: RIP: assigned sequence number 25 on Serial0
* 13:35:04.616: RIP: build update entries
* 13:35:04.620:   route 202: network 192.168.12.0 metric 1
* 13:35:04.620:   route 206: subnet 172.17.2.0 metric 1
* 13:35:04.620:   route 208: network 192.168.83.0 metric 1
* 13:35:04.620:   route 210: network 10.0.0.0 metric 2
* 13:35:04.620:   route 215: subnet 172.17.4.0 metric 2
* 13:35:04.620:   route 217: subnet 172.17.0.0 metric 2
* 13:35:04.624: RIP: Update contains 6 routes, start 202, end 226
* 13:35:04.624: RIP: start retransmit timer of 172.17.1.2
* 13:35:04.680: RIP: received v1 triggered ack from 172.17.1.2 on Serial0 flush sec
* 13:35:04.684: RIP: 172.17.1.2 change state from LOADING to FULL
* 13:35:04.688: RIP: received v1 triggered update from 172.17.1.2 on Serial0
* 13:35:04.688: RIP: sending v1 ack to 172.17.1.2 via Serial0 (172.17.1.1), flush,
seq# 14
* 13:35:04.736: RIP: received v1 triggered update from 172.17.1.2 on Serial0
* 13:35:04.740: RIP: sending v1 ack to 172.17.1.2 via Serial0 (172.17.1.1), seq# 15
* 13:35:04.740:      172.17.9.0 in 1 hops
* 13:35:04.740:      172.17.8.0 in 1 hops
* 13:35:04.740:      172.17.11.0 in 1 hops
* 13:35:04.744:      172.17.10.0 in 1 hops
* 13:35:52.879: RIP-TIMER: sending timer on Serial0 expired
```

```
* 13:36:21.907: RIP-TIMER: sending timer on Serial0 expired
* 13:36:47.421: RIP-TIMER: sending timer on Serial0 expired
```

The triggered state goes from DOWN, through INIT and LOADING, to FULL. Route information is exchanged and updates are acknowledged. At the end of the output, you can see that the RIP upd timers are expiring, but no new updates are sent, nor are updates received.

PREV NEXT

# Troubleshooting RIP

Troubleshooting RIP is relatively simple. Most difficulties with classful protocols such as RIP involve either misconfigured subnet masks or discontiguous subnets. If a route table contains inaccurate or missing routes, check all subnets for contiguity and all subnet masks for consistency.

A final command can be useful when a high-speed router is sending multiple RIP messages to a low-speed router. In such a case, the low-speed router might not be able to process updates as quickly as they are received, and routing information might be lost. `output-delay` *delay* can be used under the RIP command to set an interpacket gap of between 8 and 50 milliseconds. (The default is 0 milliseconds.)

# Looking Ahead

The simplicity, maturity, and widespread acceptance of RIP ensure that it will be in service for many years to come. However, you saw in this chapter the limitations of its classful nature. The next chapter continues to examine RIP, and you see both how the protocol has been extended to support classless routing and also how it has been extended to support IPv6.

# Summary Table: Chapter 5 Command Review

| Command | Description |
|---|---|
| debug ip rip [events] | Summarizes RIP traffic to and from the router |
| ip address *ip-address mask* secondary | Configures an interface with the indicated IP address as a secondary address |
| ip rip triggered | Configures triggered extensions to RIP on an interface |
| neighbor *ip-address* | Establishes the link indicated by the IP address as a neighbor of the interface |
| network *network-number* | Specifies the indicated network as one that will run RIP |
| offset-list {*access-list-number* \| *name*} {in \| out} *offset* [*type number*] | Stipulates that a route entry belonging to the indicated access list will have the indicated offset number added to its metric |
| output-delay *delay* | Sets an interpacket gap of the indicated delay length to accommodate processing delays between high-speed and low-speed routers |
| passive-interface *type number* | Blocks RIP broadcasts on the interface indicated by type and number |
| router rip | Enables RIP |
| timers basic *update invalid holddown flush* | Manipulates the value of the indicated timer |

# Recommended Reading

Hedrick, C. "Routing Information Protocol." RFC 1058, June 1988.

Meyer, G. and Sherry, S. "Triggered Extensions to RIP to Support Demand Circuits." RFC 2091, January 1997.

# Review Questions

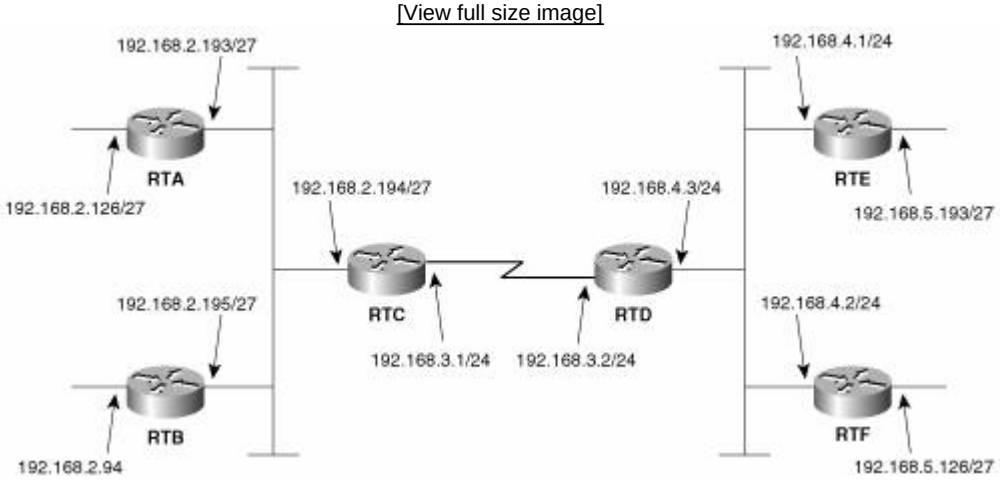
1. What port does RIP use?
2. What metric does RIP use? How is the metric used to indicate an unreachable network?
3. What is the update period for RIP?
4. How many updates must be missed before a route entry will be marked as unreachable?
5. What is the purpose of the garbage collection timer?
6. Why is a random timer associated with triggered updates? What is the range of this timer?
7. What is the difference between a RIP Request message and a RIP Response message?
8. Which two types of Request messages does RIP use?
9. Under what circumstances will a RIP response be sent?
10. Why does RIP hide subnets at major network boundaries?

# Configuration Exercises

1 Write configurations for the six routers in Figure 5-11 to route to all subnets via RIP.

Figure 5-11. Network for Configuration Exercises 1 through 4.

[View full size image]




2 Change the configurations of Configuration Exercise 1 so that RIP updates are unicast between RTC and RTD, instead of broadcast.

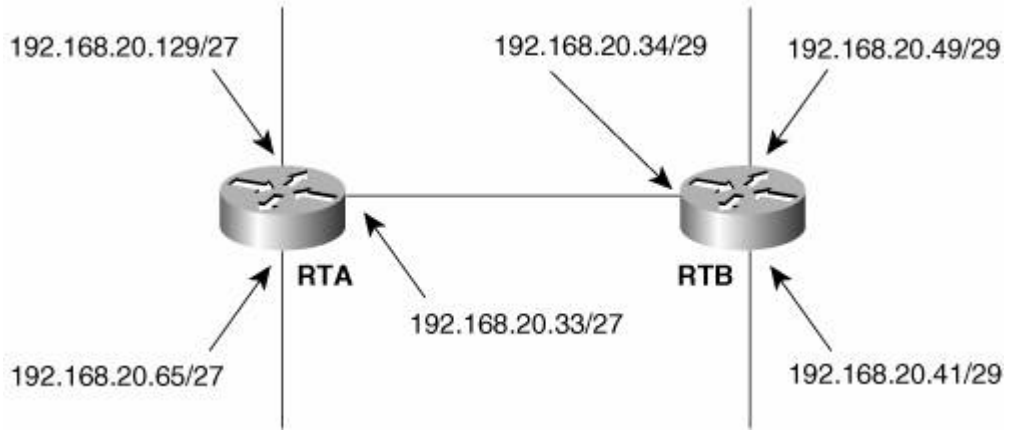
3 The bandwidth of the serial link between RTC and RTD in Figure 5-11 is very limited. Configure RIP to send updates across this link every two minutes. Carefully consider what timers must be changed and on what routers the timers must be changed.

4 A policy has been established that dictates that network 192.168.4.0 should be unreachable from RTA and that network 192.168.5.0 should be unreachable from RTB. Use one or more offset lists to implement this policy.

5 According to the section, "Classful Routing: Directly Connected Subnets," subnet masks within a major, class-level network must be consistent. The section does not, however, say that subnet masks within a major, class-level network must be identical. The RIP configuration for both routers in Figure 5-12 follows:

```
router rip
network 192.168.20.0
```

Figure 5-12. Network for Configuration Exercise 5.




Will packets be routed correctly in this small network? Explain why or why not.

.

PREV NEXT

# Troubleshooting Exercises

1 In the first offset list example, the access list in Barney is changed from

```
access-list 5 permit 10.33.32.0 0.0.0.0
```

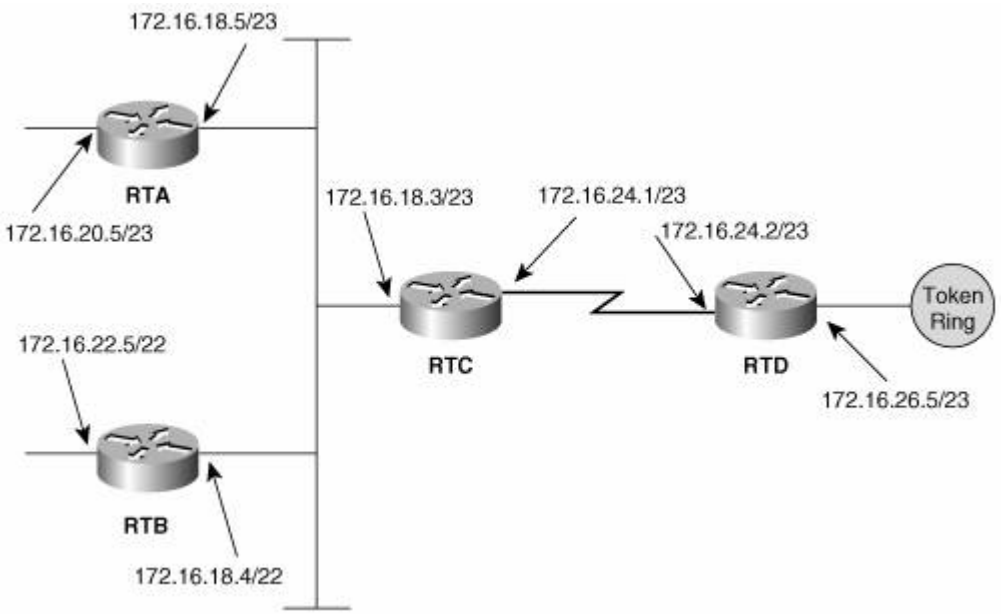
to

```
access-list 5 deny 10.33.32.0 0.0.0.0
access-list 5 permit any
```

What is the result?

2 Figure 5-13 shows a network in which the IP address masks on one router have been misconfigured. Example 5-27 through Example 5-29 show the route tables of RTA, RTB, and RTC, respectively. Based on what you know about the way RIP advertises and receives upda explain each entry in RTB's route table. Explain why RTB's entry for subnet 172.16.26.0 indicates a 32-bit mask. If any entries are missing in any of the route tables, explain why.

Figure 5-13. Network for Troubleshooting Exercises 2 and 3.




Example 5-27. Route table of RTA in Figure 5-13.

```
RTA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate defau
```

```
       U - per-user static route
Gateway of last resort is not set
     172.16.0.0/16 is subnetted, 4 subnets
R       172.16.24.0 [120/1] via 172.16.18.3, 00:00:01, Ethernet0
R       172.16.26.0 [120/2] via 172.16.18.3, 00:00:01, Ethernet0
C       172.16.20.0 is directly connected, Ethernet1
C       172.16.18.0 is directly connected, Ethernet0
RTA#
```

Example 5-28. Route table of RTB in Figure 5-13.

```
RTB#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate defau
       U - per-user static route, o - ODR
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
R       172.16.24.0/22 [120/1] via 172.16.18.3, 00:00:20, Ethernet0
R       172.16.26.0/32 [120/2] via 172.16.18.3, 00:00:20, Ethernet0
C       172.16.20.0/22 is directly connected, Ethernet1
C       172.16.16.0/22 is directly connected, Ethernet0
RTB#
```

Example 5-29. Route table of RTC in Figure 5-13.

```
RTC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate defau
       U - per-user static route, o - ODR
Gateway of last resort is not set
     172.16.0.0/23 is subnetted, 4 subnets
C       172.16.24.0 is directly connected, Serial0
R       172.16.26.0 [120/1] via 172.16.24.2, 00:00:09, Serial0
R       172.16.20.0 [120/1] via 172.16.18.5, 00:00:25, Ethernet0
C       172.16.18.0 is directly connected, Ethernet0
RTC#
```

3 Users on subnet 172.16.18.0/23 in Figure 5-13 are complaining that connectivity to subnet 172.16.26.0/23 is intermittentsometimes it can be reached, sometimes it can't. (The bad subnet masks of RTB have been corrected.) A first examination of the route tables of RTC a RTD (Example 5-30) seems to show no problems. All subnets are in both tables. Yet a minu so later, RTC shows subnet 172.16.26.0/23 to be unreachable (Example 5-31), whereas RT still shows all subnets. A few minutes after that, the subnet is back in RTC's route table (Example 5-32). In each of the three figures, the route table of RTD shows no change. A ca examination of the route tables in Example 5-30 through Example 5-32 will reveal the probl What is it?

Example 5-30. Route tables of RTC and RTD in Figure 5-13.

```
RTC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route, o - ODR
Gateway of last resort is not set
     172.16.0.0/23 is subnetted, 5 subnets
C       172.16.24.0 is directly connected, Serial0
R       172.16.26.0 [120/1] via 172.16.24.2, 00:02:42, Serial0
R       172.16.20.0 [120/1] via 172.16.18.5, 00:00:22, Ethernet0
R       172.16.22.0 [120/1] via 172.16.18.4, 00:00:05, Ethernet0
C       172.16.18.0 is directly connected, Ethernet0
```

---

```
RTD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route
Gateway of last resort is not set
     172.16.0.0/16 is subnetted, 5 subnets
C       172.16.24.0 is directly connected, Serial0
C       172.16.26.0 is directly connected, TokenRing0
R       172.16.20.0 [120/2] via 172.16.24.1, 00:00:00, Serial0
R       172.16.22.0 [120/2] via 172.16.24.1, 00:00:00, Serial0
R       172.16.18.0 [120/1] via 172.16.24.1, 00:00:00, Serial0
```

Example 5-31. Route tables of RTC and RTD, examined approximately 6( seconds after Example 5-30.

```
RTC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route, o - ODR
Gateway of last resort is not set
     172.16.0.0/23 is subnetted, 5 subnets
C       172.16.24.0 is directly connected, Serial0
R       172.16.26.0/23 is possibly down,
           routing via 172.16.24.2, Serial0
R       172.16.20.0 [120/1] via 172.16.18.5, 00:00:19, Ethernet0
R       172.16.22.0 [120/1] via 172.16.18.4, 00:00:24, Ethernet0
C       172.16.18.0 is directly connected, Ethernet0
```

---

```
RTD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route
Gateway of last resort is not set
     172.16.0.0/16 is subnetted, 5 subnets
C       172.16.24.0 is directly connected, Serial0
C       172.16.26.0 is directly connected, TokenRing0
R       172.16.20.0 [120/2] via 172.16.24.1, 00:00:15, Serial0
```

```
R       172.16.22.0 [120/2] via 172.16.24.1, 00:00:15, Serial0
R       172.16.18.0 [120/1] via 172.16.24.1, 00:00:15, Serial0
```

Example 5-32. Route tables of RTC and RTD, examined approximately 1: seconds after Example 5-31.

```
RTC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route, o - ODR
Gateway of last resort is not set
     172.16.0.0/23 is subnetted, 5 subnets
C       172.16.24.0 is directly connected, Serial0
R       172.16.26.0 [120/1] via 172.16.24.2, 00:00:09, Serial0
R       172.16.20.0 [120/1] via 172.16.18.5, 00:00:11, Ethernet0
R       172.16.22.0 [120/1] via 172.16.18.4, 00:00:18, Ethernet0
C       172.16.18.0 is directly connected, Ethernet0
```

---

```
RTD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route
Gateway of last resort is not set
     172.16.0.0/16 is subnetted, 5 subnets
C       172.16.24.0 is directly connected, Serial0
C       172.16.26.0 is directly connected, TokenRing0
R       172.16.20.0 [120/2] via 172.16.24.1, 00:00:19, Serial0
R       172.16.22.0 [120/2] via 172.16.24.1, 00:00:19, Serial0
R       172.16.18.0 [120/1] via 172.16.24.1, 00:00:19, Serial0
```

PREV NEXT

# Chapter 6. RIPv2, RIPng, and Classless Routing

This chapter covers the following subjects:

- Operation of RIPv2
- Operation of RIPng
- Configuring RIPv2
- Configuring RIPng
- Troubleshooting RIPv2 and RIPng

RIP Version 2 (RIPv2) is defined in RFC 1723[1] and is supported in IOS Versions 11.1 and later. RIPv2 is not a new protocol; rather, it is RIPv1 with some extensions to bring it more up to date with modern routing environments. These extensions are

[1] Supplemental to this RFC are RFC 1721, "RIP Version 2 Protocol Analysis," and RFC 1722, "RIP Version 2 Protocol Applicability Statement."

- Subnet masks carried with each route entry
- Authentication of routing updates
- Next-hop addresses carried with each route entry
- External route tags
- Multicast route updates

The most important of these extensions is the addition of a Subnet Mask field to the routing update entries, enabling the use of variable-length subnet masks and qualifying RIPv2 as a classless routing protocol.

RIPv2 is the first of the classless routing protocols discussed in this book. As such, this chapter serves as an introduction to classless routing, and to RIPv2.

RIP next generation (RIPng), a modification of RIPv2 for routing IPv6, is also covered in this chapter. Unlike IPv4, IPv6 is inherently classless; there are no class A, B, and C, or similar address groupings. Therefore RIPng is also a classless routing protocol.

# Operation of RIPv2

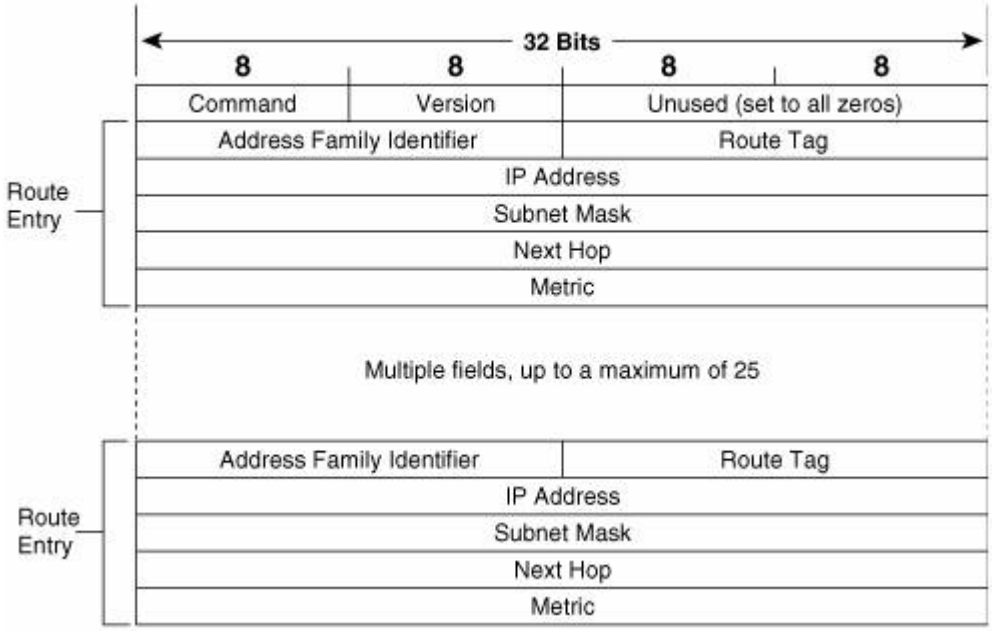
All of the operational procedures, timers, and stability functions of RIPv1 remain the same in Version 2, with the exception of the broadcast updates. RIPv2 multicasts updates to other RIPv2-speaking routers, using the reserved class D address 224.0.0.9. The advantage of multicasting is that devices on the local network that are not concerned with RIP routing do not have to spend time "unwrapping" broadcast packets from the router. The multicast updates are examined further in the section, "Compatibility with RIPv1."

After a look at how the RIP message format accommodates the Version 2 extensions, this section focuses on the operation and benefits of these additional features.

## RIPv2 Message Format

The RIPv2 message format is shown in Figure 6-1; the basic structure is the same as for RIPv1. All the extensions to the original protocol are carried within what were unused fields. Like Version 1, RIPv2 updates can contain entries for up to 25 routes. Also like Version 1, RIPv2 operates from UDP port 520 and has a maximum datagram size (with an eight-byte UDP header) of 512 octets.

Figure 6-1. RIPv2 takes advantage of the unused fields of the Version 1 message so that the extensions do not change the basic format.




- Command will always be set to either one, signifying a request message, or two, signifying a response message.
- Version will be set to two for RIPv2. If it is set to zero, or if it is set to one but the message is not a valid RIPv1 format, the message will be discarded. RIPv2 will process

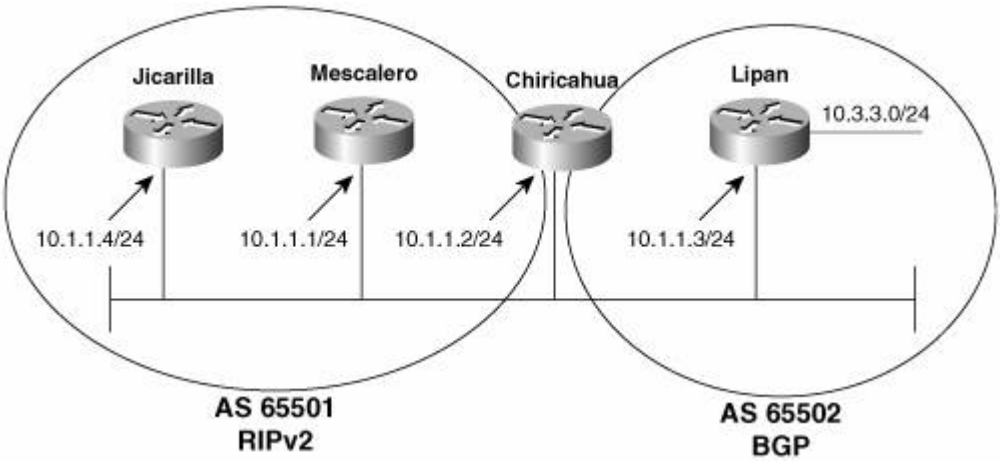
valid RIPv1 messages.

- Address Family Identifier is set to two for IPv4. The only exception is a request for a router's (or host's) full routing table, in which case it will be set to zero.
- Route Tag provides a field for tagging external routes or routes that have been redistributed into the RIPv2 process. One suggested use of this 16-bit field is to carry the autonomous system number of routes that have been imported from an external routing protocol. Although RIP itself does not use this field, external routing protocols connected to a RIP domain in multiple locations may use the route tag field to exchange information across the RIP domain. The field may also be used to group certain external routes for easier control within the RIP domain. The use of route tags is discussed further in Chapter 14, "Route Maps."
- IP Address is the IPv4 address of the destination of the route. It may be a major network address, a subnet, or a host route.
- Subnet Mask is 32-bit mask that identifies the network and subnet portion of the IPv4 address. The significance of this field is discussed in the section "Variable-Length Subnet Masking."
- Next Hop identifies a better next-hop address, if one exists, than the address of the advertising router. That is, it indicates a next-hop address, on the same subnet, that is metrically closer to the destination than the advertising router is. If the field is set to all zeros (0.0.0.0), the address of the advertising router is the best next-hop address. An example of where this field would be useful is given at the end of this section.
- Metric is a hop count between 1 and 16.

Figure 6-2 shows four routers connected to an Ethernet link.[2] Jicarilla, Mescalero, and Chiricahua are all in autonomous system number 65501 and are speaking RIPv2. Chiricahua is a border router between autonomous system 65501 and autonomous system 65502; in the second autonomous system, it speaks BGP to Lipan.

[2] This figure is an adaptation of an example presented by Gary Malkin in RFC 1722.

Figure 6-2. Although they share a common data link, Jicarilla and Mescalero speak only RIPv2; Lipan speaks only BGP. Chiricahua is responsible for informing the first two routers of any routes learned from the latter.



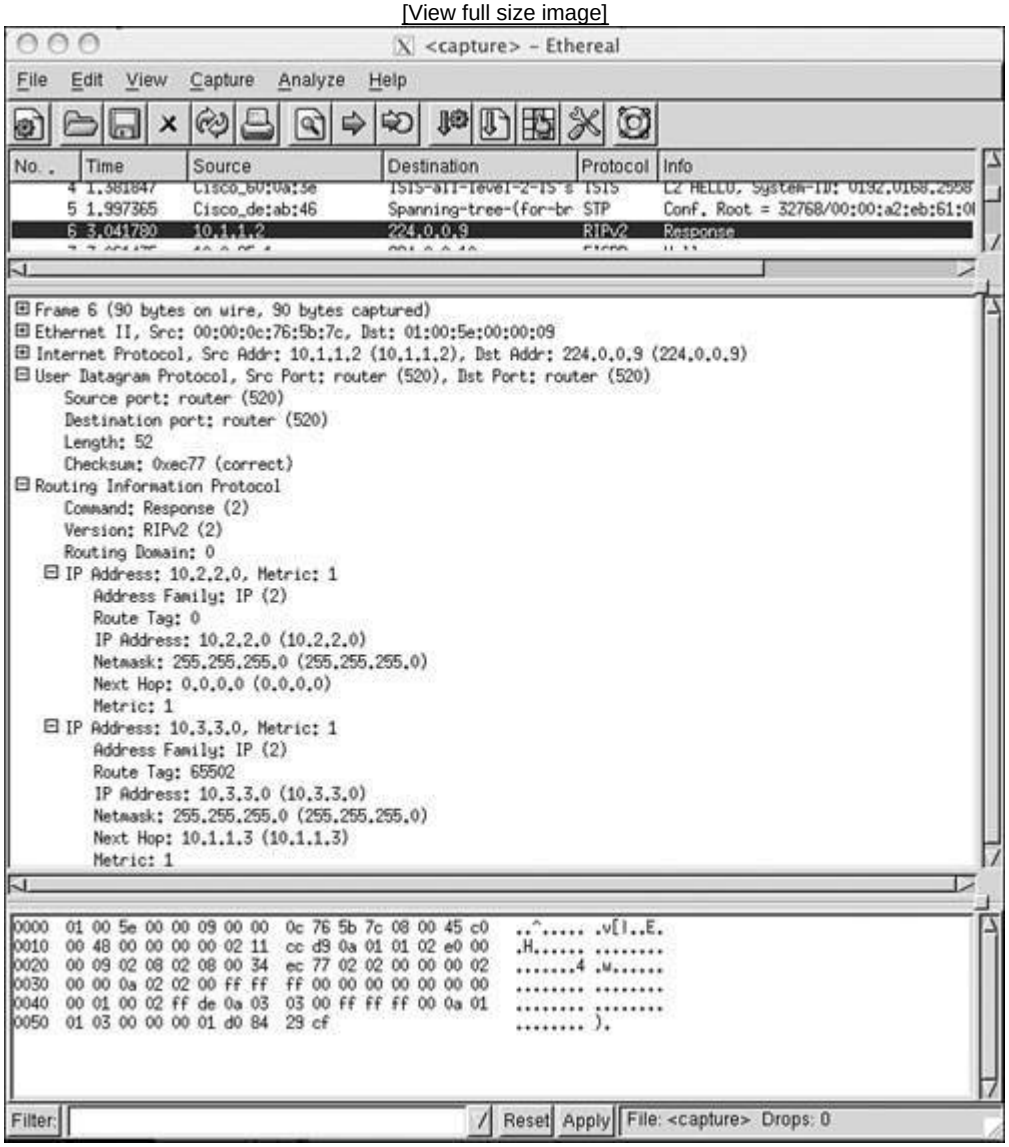

Here, Chiricahua is advertising routes it learns from BGP to the RIP-speaking routers (Figure 6-

3).[3] In its RIPv2 advertisements, Chiricahua will use the Route Tag field to indicate that subnet 10.3.3.0, with a mask of 255.255.255.0, is in autonomous system 65502 (0xFFDE). Chiricahua will also use the Next Hop field to inform Jicarilla and Mescalero that the best next-hop address to 10.3.3.0 is Lipan's interface, 10.1.1.3, rather than its own interface. Note that because Lipan does not run RIP, and Jicarilla and Mescalero do not run BGP, Jicarilla and Mescalero have no way of knowing directly that Lipan is the best next-hop router, even though it is reachable on the same subnet.

[3] Redistribution refers to the practice of advertising routes learned from one protocol to another protocol; it is discussed in detail in Chapter 11, "Route Redistribution."

Figure 6-3. This protocol capture of a RIPv2 update from Chiricahua shows the Route Tag, Subnet Mask, and Next Hop fields being used to advertise subnet 10.3.3.0.

[View full size image]




## Compatibility with RIPv1

RIPv1 handles updates in a flexible manner. If the Version field indicates Version 1 but any bits of any unused fields are set to one, the update is discarded. If the version is greater than one,

the fields defined as unused in Version 1 are ignored and the message is processed. As a result, newer editions of the protocol, like RIPv2, can be backward-compatible with RIPv1.

RFC 1723 defines a "compatibility switch" with four settings, which allows Versions 1 and 2 to interoperate:

1. *RIP-1*, in which only RIPv1 messages are transmitted
2. *RIP-1 Compatibility*, which causes RIPv2 to broadcast its messages instead of multicast them so that RIPv1 may receive them
3. *RIP-2*, in which RIPv2 messages are multicast to destination address 224.0.0.9
4. *None,* in which no updates are sent

The RFC recommends that these switches be configurable on a per interface basis. The Cisco commands for settings 1 through 3 are presented in the section "Configuring RIPv2"; setting 4 is accomplished by using the passive-interface command.

Additionally, RFC 1723 defines a "receive control switch" to regulate the reception of updates. The four recommended settings of this switch are

1. RIP-1 Only
2. RIP-2 Only
3. Both
4. None

This switch should also be configurable on a per interface basis. The Cisco commands for settings 1 through 3 are also presented in the configuration section of this chapter. Setting 4 can be accomplished by using an access list to filter UDP source port 520, by not including a network statement for the interface,[4] or by configuring a route filter as discussed in Chapter 13, "Route Filtering."

[4] This method would work only if no other interface on the router on which RIP should run is attached to the same major network.

## Classless Route Lookups

Chapter 5, "Routing Information Protocol (RIP)," explains classful route lookups, in which a destination address is first matched to its major network address in the routing table and is then matched to a subnet of the major network. If no match is found at either of these steps, the packet is dropped.

Classful route lookup was the default IOS behavior until 11.3, when the default route lookup behavior was changed to classless. For earlier IOS versions you can enable classless route lookup, even for classful routing protocols such as RIPv1 and IGRP, by entering the global command ip classless. When a router performs classless route lookups, it does not pay attention to the class of the destination address. Instead, it performs a bit-by-bit best match between the destination address and all its known routes. This capability can be very useful when working with default routes, as demonstrated in Chapter 12, "Default Routes and On-Demand Routing." When coupled with the other features of classless routing protocols, classless route lookups can be very powerful.

## Classless Routing Protocols

The true defining characteristic of classless routing protocols is the capability to carry subnet masks in their route advertisements. One benefit of having a mask associated with each route is that the all-zeros and all-ones subnets are now available for use. Chapter 1, "TCP/IP Review," explained that classful routing protocols cannot distinguish between an all-zeros subnet (172.16.0.0, for example) and the major network number (172.16.0.0). Likewise, they cannot distinguish between a broadcast on the all-ones subnet (172.16.255. 255) and an all-subnets broadcast (172.16.255.255).

If the subnet masks are included, this difficulty disappears. You can readily see that 172.16.0.0/16 is the major network number and that 172.16.0.0/24 is an all-zeros subnet. 172.168.255.255/16 and 172.16.255.255/24 are just as distinguishable.

By default, the Cisco IOS rejects an attempt to configure an all-zeros subnet as an invalid address/mask combination even if a classless routing protocol is running. To override this default behavior, enter the global command ip subnet-zero.

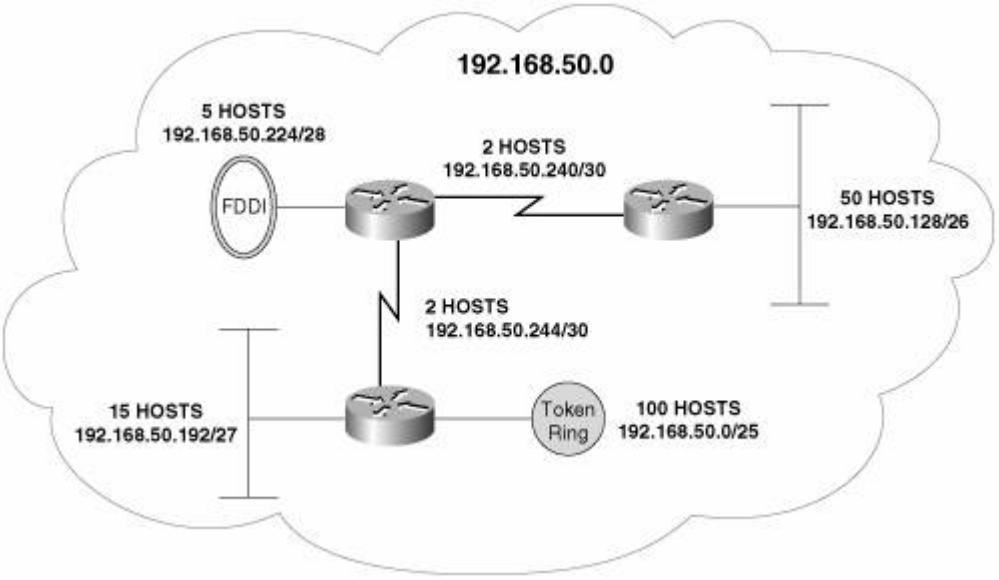
A much greater benefit of having a subnet mask associated with each route is being able to use variable-length subnet masking (VLSM) and to summarize a group of major network addresses with a single aggregate address. Variable-length subnet masks are examined in the following section, and address aggregation (or supernetting) is introduced in Chapter 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)."

## Variable-Length Subnet Masking

If a subnet mask can be individually associated with each destination address advertised throughout a network, there is no reason why all the masks must be of equal length. That fact is the basis for VLSM.

A simple application of VLSM is shown in Figure 6-4. Each data link of the network shown must have a uniquely identifiable subnet address, and each subnet address must contain enough host addresses to accommodate the devices attached to the data link.

Figure 6-4. Using VLSM, the class C address shown can be subnetted to accommodate this network and the hosts on each of its data links.



Given the class C network address assigned to this network, subnetting cannot be accomplished at all without VLSM. The token ring, with its need for 100 host addresses, requires a 25-bit mask (1 bit of subnetting); a mask any longer would not leave enough host bits. But if all masks must be of equal length, only one more subnet can be created from the class C address.[5] There would not be enough subnets to go around.

[5] This statement assumes that the all-zeros and all-ones subnetsthe only subnets available with a single bit of subnettingcan be routed.

With VLSM the widely varying host address requirements of the network of Figure 6-4 can be met using a class C network address. Table 6-1 shows the subnets and the address ranges available within each.

Table 6-1. The subnets of Figure 6-4.

| Subnet/Mask | Address Range | Broadcast Address |
|---|---|---|
| 192.168.50.0/25 | 192.168.50.1192.168.50.126 | 192.168.50.127 |
| 192.168.50.128/26 | 192.168.50.129192.168.50.190 | 192.168.50.191 |
| 192.168.50.192/27 | 192.168.50.193192.168.50.222 | 192.168.50.223 |
| 192.168.50.224/28 | 192.168.50.225192.168.50.238 | 192.168.50.239 |
| 192.168.50.240/30 | 192.168.50.241192.168.50.242 | 192.168.50.243 |
| 192.168.50.244/30 | 192.168.50.245192.168.50.246 | 192.168.50.247 |

Many people, including many who work with VLSM, make the technique more complicated than it is. The complete key to VLSM is this: After a network address is subnetted in the standard fashion, those subnets can themselves be subnetted. In fact, one will occasionally hear VLSM referred to as "sub-subnetting."

A close examination of the addresses in Table 6-1 (in binary, as always) will reveal how VLSM works.[6] First, a 25-bit mask is used to divide the network address into two subnets: 192.168.50.0/25 and 192.168.50.128/25. The first subnet provides 126 host addresses to meet the needs of the token ring in Figure 6-4.

[6] The reader is strongly encouraged to work through this entire example in binary.

From Chapter 1, you know that subnetting involves expanding the default network mask so that some host bits are interpreted as network bits. This same procedure is applied to the remaining subnet 192.168.50.128/25. One of the Ethernets requires 50 host addresses, so the mask of the remaining subnet is expanded to 26 bits. This step provides two sub-subnets, 192.168.50.128/26 and 192.168.192/26, each with 62 available host addresses. The first sub-subnet is taken for the larger Ethernet, leaving the second to again be subnetted for the other data links.

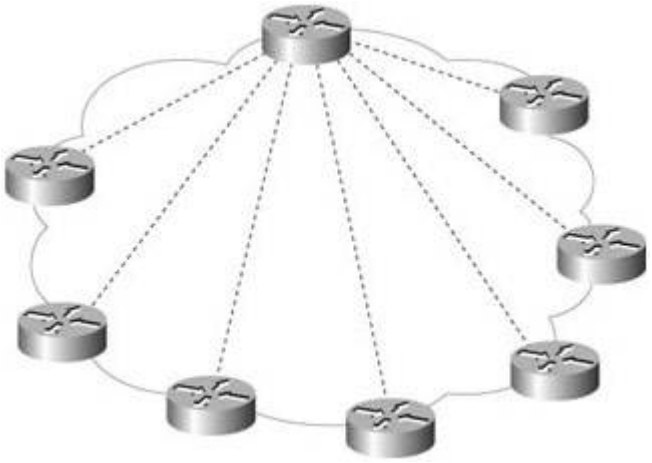
This procedure is repeated twice more to provide the necessary subnets of the necessary size for the smaller Ethernet and the FDDI ring. A subnet of 192.168.50.240/28 remains, as do two serial links requiring subnets. Any point-to-point link will, by its very nature, require only two host addressesone at each end. Thirty-bit masks are used to create the two serial link subnets, each with just two available host addresses.

Point-to-point links, requiring a subnet address but only two host addresses per subnet, are one justification for using VLSM. For example, Figure 6-5 shows a typical WAN topology with remote

routers connected via Frame Relay PVCs to a hub router. Modern practice usually calls for each of these PVCs to be configured on a point-to-point subinterface.[7] Without VLSM, equal-size subnets would be necessary; the size would be dictated by the subnet with the largest number of host devices.

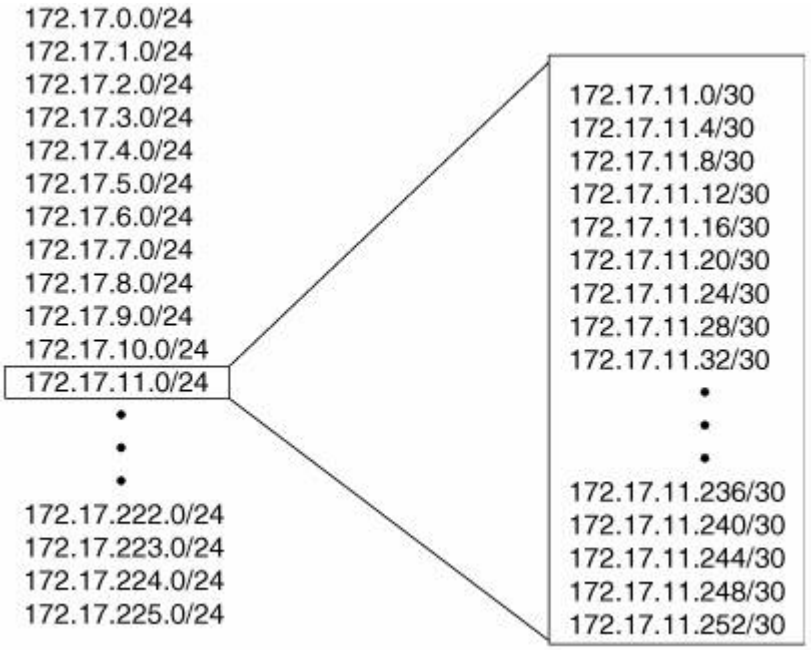
[7] Subinterfaces are logical interfaces configured on a physical interface. Many subinterfaces can be configured on a single physical interface, and each subinterface can have its own IP address; routing protocols treat them the same as physical interfaces. Subinterfaces are particularly useful with Frame Relay, ATM, and VLANs. Readers who are not already familiar with these useful tools are referred to the Cisco Configuration Guide.

Figure 6-5. VLSM allows each of these PVCs to be configured as a separate subnet without wasting host addresses.



Suppose a class B address is used for the network in Figure 6-5 and each router is attached to several LANs, each of which may have up to 175 attached devices. A 24-bit mask would be necessary for each subnet, including each PVC. Consequently, for every PVC in the network, 252 addresses are wasted. With VLSM, a single subnet can be selected and sub-subnetted with a 30-bit mask; enough subnets will be created for up to 64 point-to-point links (Figure 6-6).

Figure 6-6. This class B address has been subnetted with a 24-bit mask. 172.17.11.0 has been sub-subnetted with a 30-bit mask; the resulting 64 subnets can be assigned to point-to-point links.




Examples of VLSM address designs appear in this and subsequent chapters. Chapter 7 introduces another major justification for using VLSM, hierarchical addressing, and address aggregation.

## Authentication

A security concern with any routing protocol is the possibility of a router accepting invalid routing updates. The source of invalid updates may be an attacker trying to maliciously disrupt the network or trying to capture packets by tricking the router into sending them to the wrong destination. A more mundane source of invalid updates may be a malfunctioning router. RIPv2 includes the capability to authenticate the source of a routing update by including a password.

Authentication is supported by modifying what would normally be the first route entry of the RIP message, as shown in Figure 6-7. With authentication, the maximum number of entries a single update can carry is reduced to 24. The presence of authentication is indicated by setting the Address Family Identifier field to all ones (0xFFFF). The Authentication Type for simple password authentication is two (0x0002), and the remaining 16 octets carry an alphanumeric password of up to 16 characters. The password is left-justified in the field, and if the password is less than 16 octets, the unused bits of the field are set to zero.

Figure 6-7. The RIPv2 authentication information, when configured, is carried in the first route entry space.

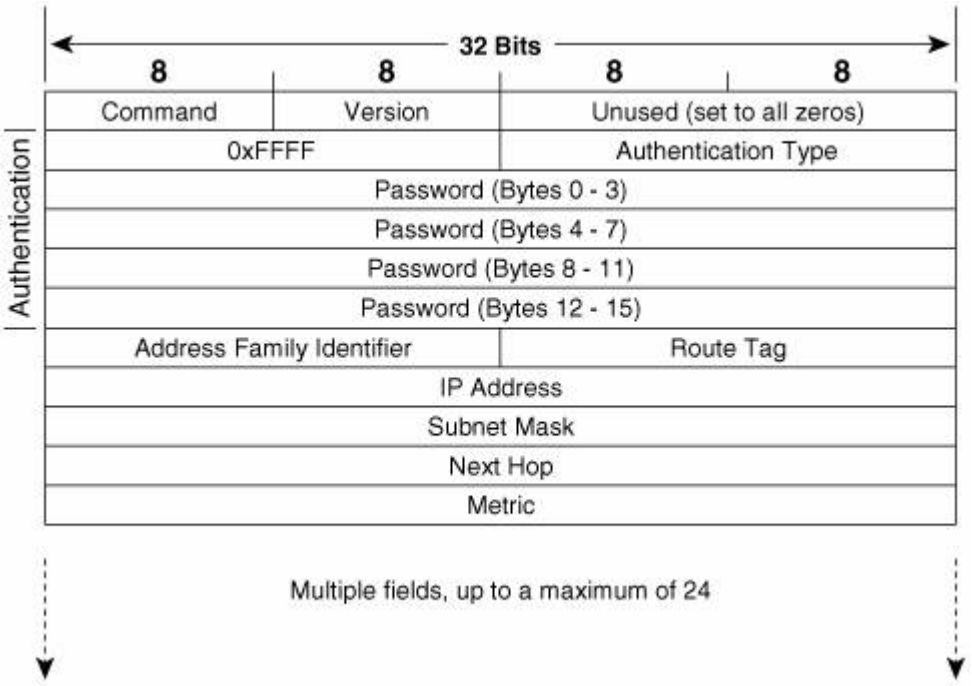
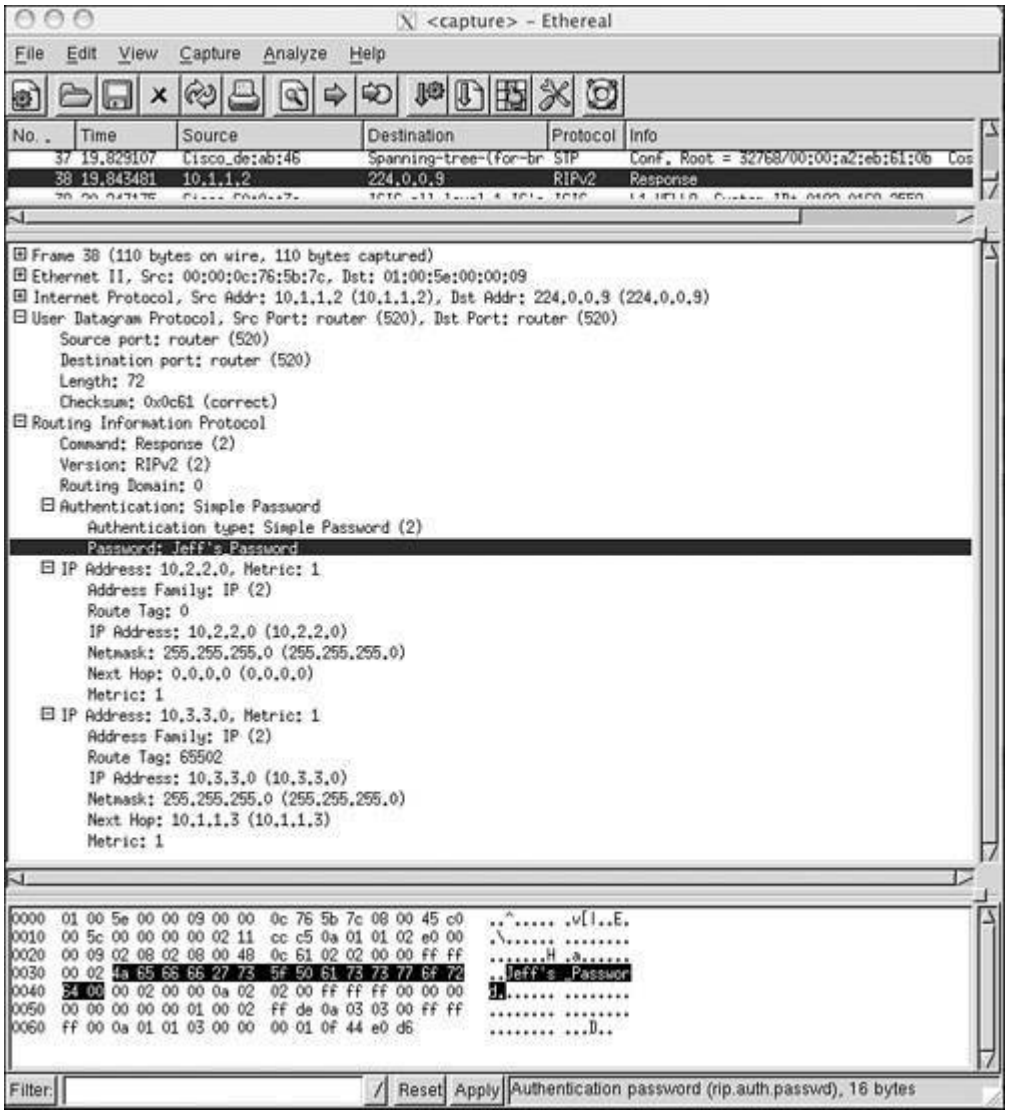



Figure 6-8 shows an analyzer capture of a RIPv2 message with authentication. The figure also shows a difficulty with the default RIP authentication: The password is transmitted in plain text. Anyone who can capture a packet containing a RIPv2 update message can read the authentication password.

Figure 6-8. When simple password authentication is used, the password is carried in plain text and can be read by anyone who can "sniff" the packet carrying the update.

[View full size image]




Although RFC 1723 describes only simple password authentication, foresight is shown by including the Authentication Type field. Cisco IOS takes advantage of this feature and provides the option of using MD5 authentication instead of simple password authentication.[8]

[8] MD5 is described in RFC 1321. A good discussion of MD5 can also be found in the following book: Charlie Kaufman, Radia Perlman, and Mike Spencer, *Network Security: Private Communication in a Public World*. Prentice Hall, 1995, pp. 120122.

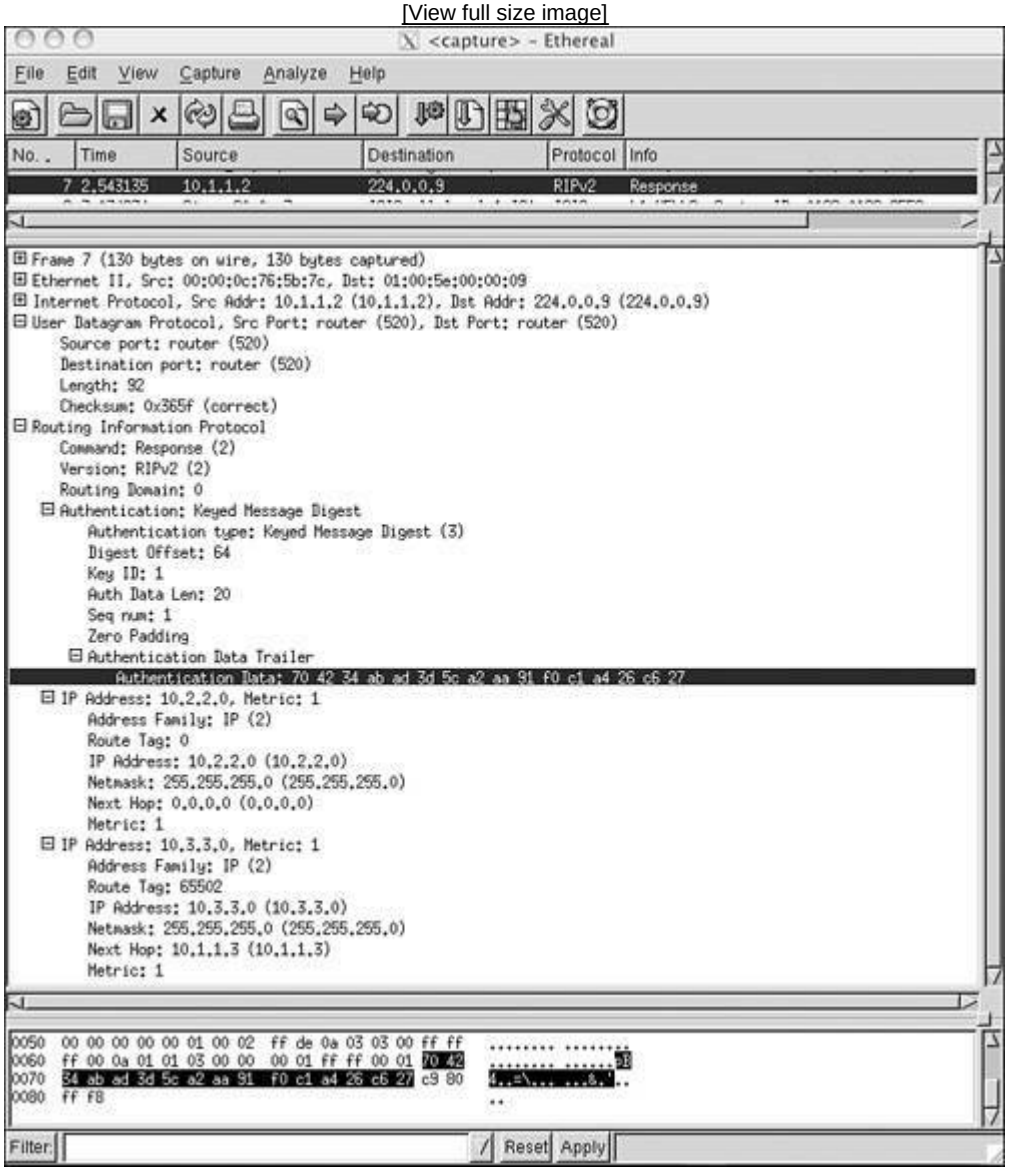
Cisco uses the first and last route entry spaces for MD5 authentication purposes.

MD5 is a one-way *message digest* or *secure hash* function, produced by RSA Data Security, Inc. It is also occasionally referred to as a *cryptographic checksum* because it works in somewhat the same way as an arithmetic checksum. MD5 computes a 128-bit hash value from a plain text message of arbitrary length (a RIPv2 update, for instance) and a password. This "fingerprint" is transmitted along with the message. The receiver, knowing the same password, calculates its own hash value. If nothing in the message has changed, the receiver's hash value should match the sender's value transmitted with the message.

Figure 6-9 shows an update from the same router of Figure 6-8, but with MD5 authentication. The authentication type is 3, and no password can be seen.

Figure 6-9. This update was originated from the same router as the update in Figure 6-8, but MD5 authentication is being used.

[View full size image]




PREV NEXT

# Operation of RIPng

RIPng for IPv6 is based on RIPv2, but it is not an extension of RIPv2; it is an entirely separate protocol. RIPng does not support IPv4, so to use RIP to route both IPv4 and IPv6, you must run RIPv1 or v2 for IPv4 and RIPng for IPv6.

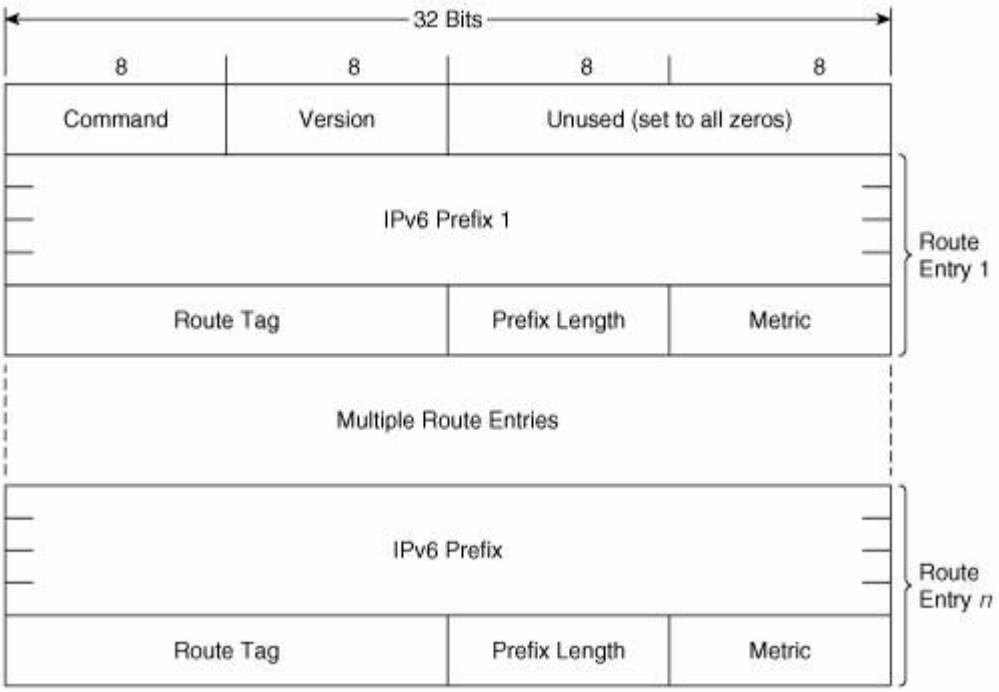
RIPng uses the same timers, procedures, and message types as RIPv2. For example, like RIPv2 it uses a 30-second update timer jittered to prevent message synchronization, a 180-second timeout period, a 120-second garbage-collection timer, and a 180-second holddown timer. It uses the same hop-count metric, with 16 indicating an unreachable value. And, it uses Request and Response messages in the same way that RIPv2 does. And, like RIPv2, Request and Response messages are multicast with the same few unicast exceptions that RIPv1 and v2 use. The IPv6 multicast address used by RIPng is FF02::9.

An exception to these parallel functions is authentication. RIPng does not have an authentication mechanism of its own, but instead relies on the authentication features built into IPv6.

There is, of course, no need for the RIPv1 compatibility switches used by RIPv2, because there is no backward support for IPv4.

Figure 6-10 shows the RIPng message format. Unlike RIPv1 and v2, which run at UDP port 520, RIPng sends and receives its messages at UDP port 521. Also unlike RIPv1 and v2, there is no set message size. The message size is dependent only on the MTU of the link on which it is being sent.

Figure 6-10. The RIPng message format.




- Command is always set to 1, indicating a Request message, or to 2, indicating a Response

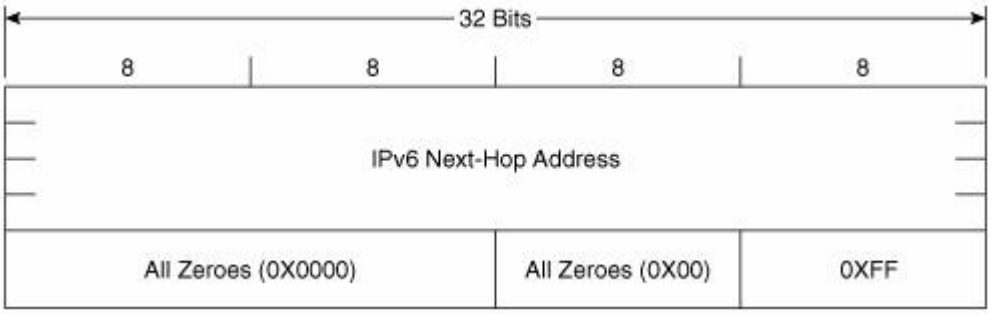
message.

- Version is always set to 1, that is, the current version of RIPng in RIPngv1.
- IPv6 Prefix is the 128-bit IPv6 destination prefix of the route entry.
- Route Tag is used in the same way the 16-bit RIPv2 Route Tag field is used: for transporting external route attributes across the RIP domain.
- Prefix Length is an 8-bit field specifying, in bits, the significant part of the address in the IPv6 Prefix field. For example, if the advertised prefix is 3ffe:2100:1201::/48, the value in the Prefix Length field is 48 (0x30). If the advertised route entry is a default route, the IPv6 prefix is 0:0:0:0:0:0:0:0 and the Prefix Length is 0.
- Metric is the same hop count metric used by RIPv1 and v2. But as the maximum possible value is 16, the field is reduced to 8 bits from the unnecessarily large 16 bits of RIPv1 and v2.

RIPng specifies a next-hop address the same way that RIPv2 does. That is, a valid non-zero next-hop address specifies a next-hop router other than the originator of the Response message and a next-hop address of 0:0:0:0:0:0:0:0 specifies the originator of the Response message. But rather than associate a next-hop address with each route entry as RIPv2 does, RIPng specifies the next-hop address in a special route entry and then groups all route entries that use the next-hop address after it. That is, the next-hop address specified in a next-hop route entry applies to all of the route entries following it, either to the end of the Response message or until another special next-hop route entry is found.

Figure 6-11 shows the format of the next-hop route entry. The 128-bit address is either the IPv6 address of another router or, if it is ::, specifies the address of the originator. The Route Tag and Prefix Length fields are set to all zeroes. Receiving routers recognize the next-hop route entry because its Metric field is set to all ones (0xFF).

Figure 6-11. A metric value of all ones specifies that the route entry is a next-hop route entry. All route entries following this one, until the end of the message or until another next-hop route entry is encountered, use this next-hop address.




PREV NEXT

# Configuring RIPv2

Because RIPv2 is merely an enhancement of RIPv1 and not a separate protocol, the commands introduced in Chapter 5, "Routing Information Protocol (RIP)," for manipulating timers and metrics, and for configuring unicast updates or no updates at all, are used in exactly the same way. After a brief look at configuring a RIPv2 process, the rest of this section concentrates on configuring the new extensions.

## Case Study: A Basic RIPv2 Configuration

By default, a RIP process configured on a Cisco router sends only RIPv1 messages but listens to both RIPv1 and RIPv2. This default is changed with the version command, as in Example 6-1.

### Example 6-1. The *version 2* command causes RIP to send and listen to RIPv2 messages only.

```
router rip
 version 2
 network 172.25.0.0
 network 192.168.50.0
```

In this mode, the router sends and receives only RIPv2 messages. Likewise, the router can be configured to send and receive only RIPv1 messages, as in Example 6-2.

### Example 6-2. The *version 1* command causes RIP to send and listen to RIPv1 messages only.

```
router rip
 version 1
 network 172.25.0.0
 network 192.168.50.0
```

The default behavior can be restored by entering the command no version in config-router mode.

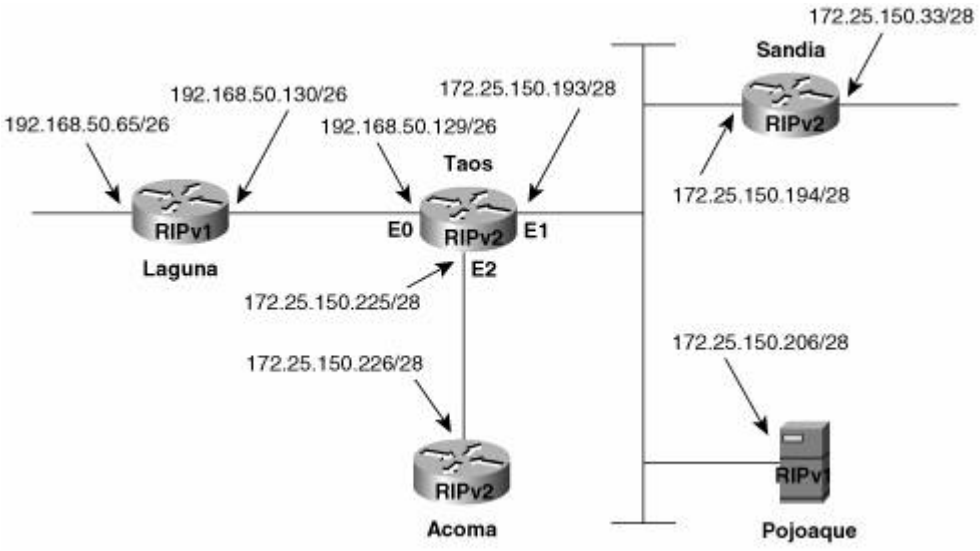
## Case Study: Compatibility with RIPv1

The interface-level "compatibility switches" recommended by RFC 2453 are implemented in Cisco IOS with the commands ip rip send version and ip rip receive version.

The network in Figure 6-12 contains routers speaking both RIPv1 and RIPv2. Additionally, Pojoaque is a Linux host running *routed*, which understands only RIPv1. The configuration of Taos is displayed in Example 6-3.

### Figure 6-12. Taos is running RIPv2 but must also speak Version 1 to

some devices.




## Example 6-3. Taos's configuration.

```
interface Ethernet0
 ip address 192.168.50.129 255.255.255.192
 ip rip send version 1
 ip rip receive version 1
!
interface Ethernet1
 ip address 172.25.150.193 255.255.255.240
 ip rip send version 1 2
!
interface Ethernet2
 ip address 172.25.150.225 255.255.255.240
!
router rip
 version 2
 network 172.25.0.0
 network 192.168.50.0
```

Because router Laguna is a RIPv1 speaker, E0 of Taos is configured to send and receive RIPv1 updates. E1 is configured to send both Version 1 and 2 updates, to accommodate the RIPv1 at Pojoaque and the RIPv2 at Sandia. E2 has no special configuration; it sends and receives Version 2 by default.

In Example 6-4, `debug ip rip` is used to observe the messages sent and received by Taos. There are several points of interest here. First, notice the difference between the traps for RIPv1 and RIPv2 messages. The address mask and the Next Hop and Route Tag fields (both set to all zeros, in this case) of the RIPv2 updates can be observed. Second, it can be observed that interface E1 is broadcasting RIPv1 updates and multicasting RIPv2 updates. Third, because Taos has not been configured to receive RIPv1, the updates from Pojoaque (172.25.150.206) are being ignored. (Pojoaque has been misconfigured and is broadcasting its routing table.)[9]

[9] Intentionally misconfigured for this example actually, with the **routed -s** option.

## Example 6-4. Using debugging, the RIP versions sent and received by Taos can be observed.

```
Taos#debug ip rip
RIP protocol debugging is on
Taos#
RIP: received v2 update from 172.25.150.194 on Ethernet1
     172.25.150.32/28 - 0.0.0.0 in 1 hops
RIP: ignored v1 packet from 172.25.150.206 (illegal version)
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (192.168.50.129)
     network 172.25.0.0, metric 1
RIP: sending v1 update to 255.255.255.255 via Ethernet1 (172.25.150.193)
     subnet 172.25.150.224, metric 1
     network 192.168.50.0, metric 1
RIP: sending v2 update to 224.0.0.9 via Ethernet1 (172.25.150.193)
     172.25.150.224/28 - 0.0.0.0, metric 1, tag 0
     192.168.50.0/24 - 0.0.0.0, metric 1, tag 0
RIP: sending v2 update to 224.0.0.9 via Ethernet2 (172.25.150.225)
     172.25.150.32/28 - 0.0.0.0, metric 2, tag 0
     172.25.150.192/28 - 0.0.0.0, metric 1, tag 0
     192.168.50.0/24 - 0.0.0.0, metric 1, tag 0
RIP: received v1 update from 192.168.50.130 on Ethernet0
     192.168.50.64 in 1 hops
RIP: received v2 update from 172.25.150.194 on Ethernet1
     172.25.150.32/28 - 0.0.0.0 in 1 hops
```

Perhaps the most important observation to be made from Example 6-4 concerns the update being broadcast to Pojoaque: It does not include subnet 172.25.150.32. Taos knows this subnet, having learned it via multicast RIPv2 updates from Sandia. But Pojoaque cannot receive those multicasts because it speaks only RIPv1. Moreover, although Taos knows the subnet, the split horizon rule prevents Taos from advertising it out the same interface on which it was learned.

So, Pojoaque does not know about subnet 172.25.150.32. Two remedies are available: First, Sandia could be configured to send both RIP versions. Second, split horizon can be turned off at Taos's E1 interface with the configuration in Example 6-5.

## Example 6-5. Split horizon is turned off here, in Taos's configuration.

```
interface Ethernet1
 ip address 172.25.150.193 255.255.255.240
 ip rip send version 1 2
 no ip split-horizon
```

Example 6-6 shows the results. Taos is now including subnet 172.25.150.32 in its updates. Some forethought must be given to the possible consequences of disabling split horizon; Taos is now not only advertising 172.25.150.32 to Pojoaque but also advertising it back to Sandia.

## Example 6-6. With split horizon disabled on E1, Taos now includes subnet 172.25.150.32 in its updates to Pojoaque.

```
Taos#debug ip rip
RIP protocol debugging is on
Taos#
RIP: ignored v1 packet from 172.25.150.206 (illegal version)
RIP: received v2 update from 172.25.150.194 on Ethernet1
     172.25.150.32/28 -> 0.0.0.0 in 1 hops
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (192.168.50.129)
     network 172.25.0.0, metric 1
```

```
RIP: sending v1 update to 255.255.255.255 via Ethernet1 (172.25.150.193)
     subnet 172.25.150.32, metric 2
     subnet 172.25.150.224, metric 1
     subnet 172.25.150.192, metric 1
     network 192.168.50.0, metric 1
RIP: sending v2 update to 224.0.0.9 via Ethernet1 (172.25.150.193)
     172.25.150.32/28 -> 172.25.150.194, metric 2, tag 0
     172.25.150.224/28 -> 0.0.0.0, metric 1, tag 0
     172.25.150.192/28 -> 0.0.0.0, metric 1, tag 0
     192.168.50.0/24 -> 0.0.0.0, metric 1, tag 0
```

## Case Study: Using VLSM

Referring again to Figure 6-12, the subnet 172.25.150.0/24 has been assigned to the Internet shown. That subnet has been further subnetted to fit the various data links by expanding the mask to 28 bits; the available sub-subnets, in binary and dotted decimal, are shown in Table 6-2. Each of the subnets[10] will have, according to the $2^n$ 2 formula, 14 host addresses. Out of these, 172.25.150.32, 172.25.150.192, and 172.25.150.224 have been used.

[10] Now that the concept should be firmly in place, from here on the single term subnet will be used for a subnet, a sub-subnet, a sub-sub-subnet, or whatever.

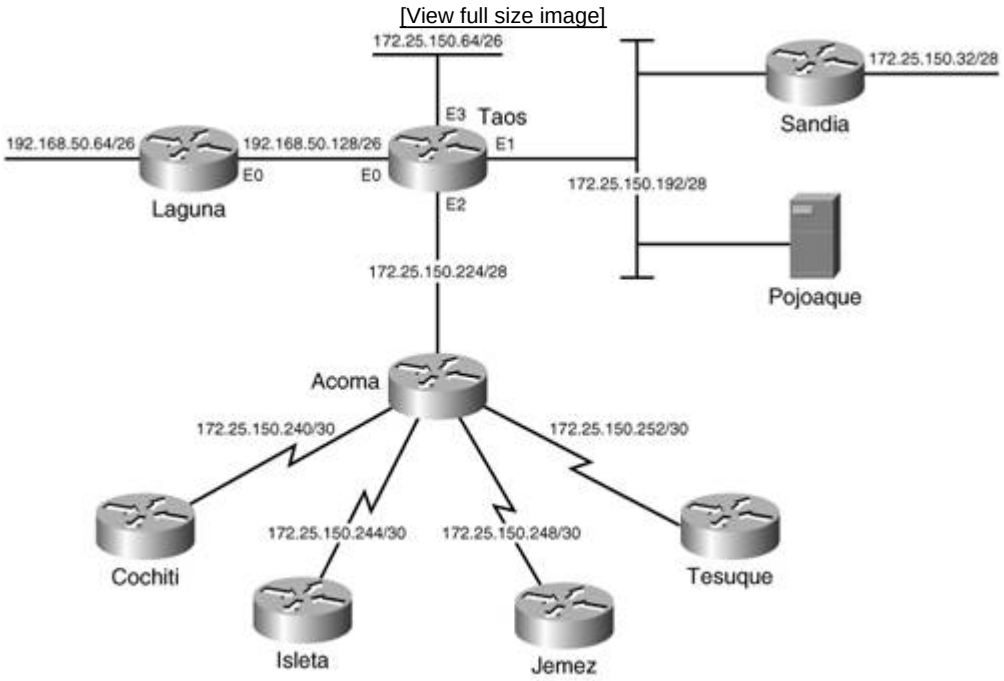
Table 6-2. VLSM is applied to subnet 172.25.150.0/24.

| Binary Representation | Dotted Decimal |
|---|---|
| 11111111111111111111111111110000 | 255.255.255.240 |
| 10101100000110011001011000000000 | 172.25.150.0/28 |
| 10101100000110011001011000010000 | 172.25.150.16/28 |
| 10101100000110011001011000100000 | 172.25.150.32/28 |
| 10101100000110011001011000110000 | 172.25.150.48/28 |
| 10101100000110011001011001000000 | 172.25.150.64/28 |
| 10101100000110011001011001010000 | 172.25.150.80/28 |
| 10101100000110011001011001100000 | 172.25.150.96/28 |
| 10101100000110011001011001110000 | 172.25.150.112/28 |
| 10101100000110011001011010000000 | 172.25.150.128/28 |
| 10101100000110011001011010010000 | 172.25.150.144/28 |
| 10101100000110011001011010100000 | 172.25.150.160/28 |
| 10101100000110011001011010110000 | 172.25.150.176/28 |
| 10101100000110011001011011000000 | 172.25.150.192/28 |
| 10101100000110011001011011010000 | 172.25.150.208/28 |
| 10101100000110011001011011100000 | 172.25.150.224/28 |
| 10101100000110011001011011110000 | 172.25.150.240/28 |

In Figure 6-13, a new Ethernet segment, Ethernet 3, has been added to Taos, with 60 hosts. A subnet with at least six host bits is required to accommodate this data link. A classful routing protocol would require that five of the subnets from Table 6-2 be assigned to Ethernet 3 [5 x ($2^4$ 2) = 70], using secondary addresses. With classless protocols and VLSM, four of the subnets from Table 6-2 can be combined into a single subnet with a 26-bit mask. This step will provide six host bits (62 host addresses), and no secondary addressing is necessary. Four subnets, 172.25.150.64/28 to 172.25.150.112/28, are combined under one 26-bit mask: 172.25.150.64/26. Notice that the four subnets are not selected at random; the first 26 masked bits are identical and are unique within the group of 16 subnets.[11]

[11] The technique used here to combine several addresses into one address serves as an introduction to address aggregation, covered in Chapter 7.

Figure 6-13. VLSM can be used to adapt addresses to the requirements of individual data links.




Also, in Figure 6-13, four serial links and four routers have been added to the network. Without VLSM, four of the subnets from Table 6-2 would have to be used for the four serial links. With VLSM, a single subnet from Table 6-2 can be used for all four serial links. 172.25.150.240 is selected, and a 30-bit mask is used to create the subnets in Table 6-3. Each of the resulting four subnets contains two host addresses.

Table 6-3. A 30-bit mask is applied to subnet 172.25.150.240.

| Binary Representation | Dotted Decimal |
|---|---|
| 11111111111111111111111111111100 | 255.255.255.252 |
| 10101100000110011001011011110000 | 172.25.150.240/30 |
| 10101100000110011001011011110100 | 172.25.150.244/30 |

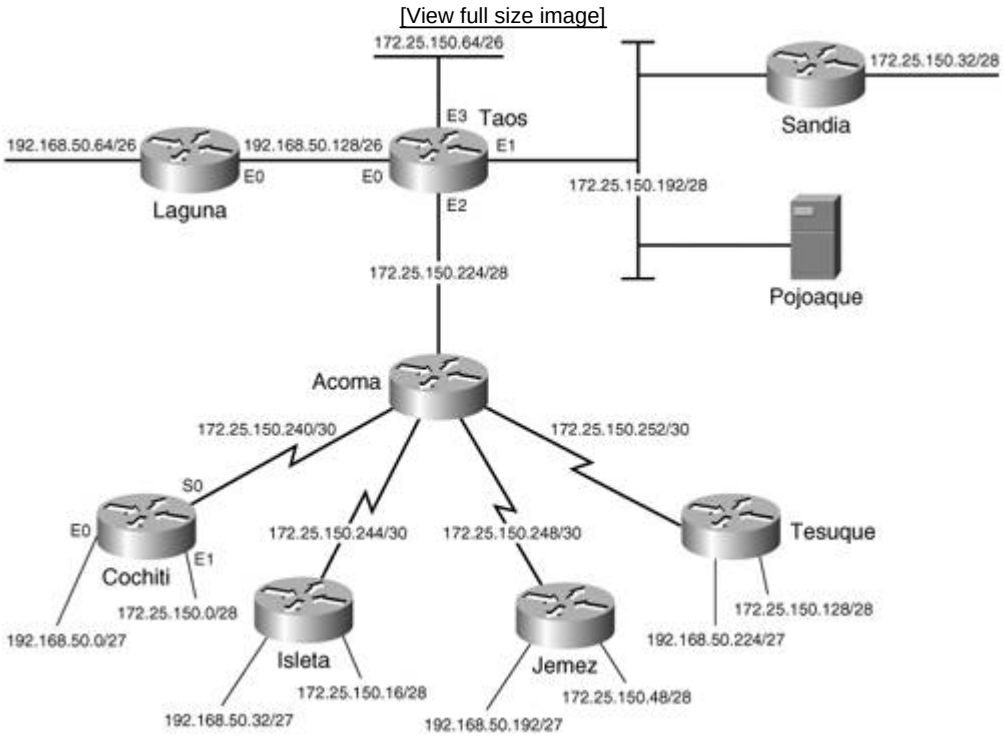
| Binary Representation | Dotted Decimal |
|---|---|
| 10101100000110011001011011111000 | 172.25.150.248/30 |
| 10101100000110011001011011111100 | 172.25.150.252/30 |

The fundamental objective of subnetting is always the same: A router must be able to identify every data link with a unique address, distinct from any other address in the network. That is the common goal in the preceding two examples. In the first example, multiple addresses were combined into a single address by reducing the size of the mask until only the bits common to all of the addresses remain. Note that this result also happens when subnets are summarized by the major network address. In the second example, multiple subnets were created from a single subnet by expanding the subnet mask.

## Case Study: Discontiguous Subnets and Classless Routing

Figure 6-14 shows that two Ethernets are connected to each of the four new routers. At each site, one Ethernet is a member of subnet 172.25.150.0/24 and will have no more than 12 hosts. This is easy enough. Four unused subnets are chosen from Table 6-2 and assigned.

Figure 6-14. Cochiti, Isleta, Jemez, and Tesuque are each attached to two Ethernets. One Ethernet at each router is a member of subnet 172.25.150.0/24, and the other is a member of network 192.168.50.0/24.




The other Ethernet at each site is a member of network 192.168.50.0 and will have no more than 25 hosts. Subnets 192.168.50.64/26 and 192.168.50.128/26 are being used, which leaves 192.168.50.0/26 and 192.168.50.192/26. By increasing the mask to 27 bits, these two subnets can be divided into four, each with five host bitsenough for 30 host addresses per subnet. Table

6-4 shows the four subnets in binary.

### Table 6-4. Subnet 192.169.50.0/26 is further subnetted with a 27-bit mask.

| Binary Representation | Dotted Decimal |
|---|---|
| 11111111111111111111111111100000 | 255.255.255.224 |
| 11000000101010001100100000000000 | 192.169.50.0/27 |
| 11000000101010001100100000100000 | 192.168.50.32/27 |
| 11000000101010001100100011000000 | 192.168.50.192/27 |
| 11000000101010001100100011100000 | 192.168.50.224/27 |

Having assigned all subnet addresses, the next concern is the fact that the subnets of 192.168.50.0 are discontiguous. Chapter 5 presents a case study on discontiguous subnets and demonstrates how to use secondary interfaces to connect them. Classless routing protocols have no such difficulties with discontiguous subnets. Because each route update includes a mask, subnets of one major network can be advertised into another major network.

The default behavior of RIPv2, however, is to summarize at network boundaries the same as RIPv1. To turn off summarization and allow subnets to be advertised across network boundaries, use the command `no auto-summary` with the RIP process. The configuration for Cochiti is displayed in Example 6-7.

### Example 6-7. Cochiti's configuration with no automatic summarization.

```
interface Ethernet0
 ip address 192.168.50.1 255.255.255.224
!
interface Ethernet1
 ip address 172.25.150.1 255.255.255.240
!
interface Serial0
 ip address 172.25.150.242 255.255.255.252
!
router rip
 version 2
 network 172.25.0.0
 network 192.168.50.0
 no auto-summary
```

Isleta, Jemez, and Tesuque will have similar configurations. Summarization must also be turned off at Taos and at Acoma. Recall from Figure 6-12 that Laguna was running RIPv1. For this configuration to work, it must be changed to Version 2.

Careful consideration should be given to what effect the variable masks will have on Pojoaque, where RIPv1 continues to run. The debug messages in Example 6-8 show the Version 1 and Version 2 updates sent from Taos onto subnet 172.25.150.192/28. The Version 1 updates will include only those subnets whose masks are 28 bits, the same as the subnet on which the updates are being broadcast. Although Pojoaque will not receive advertisements for 172.25.150.64/26 or any of the serial link subnets, an analysis of those subnet addresses shows that Pojoaque will, in this case, correctly interpret the addresses as being different from its own

subnet. Packets destined for these subnets will be sent to Taos for routing.

### Example 6-8. Although the RIPv2 update from Taos includes all subnets in the network, the RIPv1 update includes only a summary route to network 192.168.50.0 and only those subnets of 172.25.150.0 whose masks are the same as the interface on which the update is being sent.

```
Taos#debug ip rip
RIP protocol debugging is on
RIP: sending v1 update to 255.255.255.255 via Ethernet0 (172.25.150.193)
     subnet 172.25.150.0, metric 3
     subnet 172.25.150.16, metric 3
     subnet 172.25.150.32, metric 2
     subnet 172.25.150.48, metric 3
     subnet 172.25.150.128, metric 3
     subnet 172.25.150.192, metric 1
     subnet 172.25.150.224, metric 1
     network 192.168.50.0, metric 1
RIP: sending v2 update to 224.0.0.9 via Ethernet0 (172.25.150.193)
     172.25.150.0/28 -> 0.0.0.0, metric 3, tag 0
     172.25.150.16/28 -> 0.0.0.0, metric 3, tag 0
     172.25.150.32/28 -> 0.0.0.0, metric 2, tag 0
     172.25.150.48/28 -> 0.0.0.0, metric 3, tag 0
     172.25.150.64/26 -> 0.0.0.0, metric 1, tag 0
     172.25.150.128/28 -> 0.0.0.0, metric 3, tag 0
     172.25.150.192/28 -> 0.0.0.0, metric 1, tag 0
     172.25.150.224/28 -> 0.0.0.0, metric 1, tag 0
     172.25.150.240/30 -> 0.0.0.0, metric 2, tag 0
     172.25.150.244/30 -> 0.0.0.0, metric 2, tag 0
     172.25.150.248/30 -> 0.0.0.0, metric 2, tag 0
     172.25.150.252/30 -> 0.0.0.0, metric 2, tag 0
     192.168.50.0/27 -> 0.0.0.0, metric 3, tag 0
     192.168.50.32/27 -> 0.0.0.0, metric 3, tag 0
     192.168.50.64/26 -> 0.0.0.0, metric 2, tag 0
     192.168.50.128/26 -> 0.0.0.0, metric 1, tag 0
     192.168.50.192/27 -> 0.0.0.0, metric 3, tag 0
     192.168.50.224/27 -> 0.0.0.0, metric 3, tag 0
```

## Case Study: Authentication

The Cisco implementation of RIPv2 message authentication includes the choice of simple password or MD5 authentication, and the option of defining multiple keys, or passwords, on a "key chain." The router may then be configured to use different keys at different times.

The steps for setting up RIPv2 authentication follow:

Step 1. Define a key chain with a name.

Step 2. Define the key or keys on the key chain.

Step 3. Enable authentication on an interface and specify the key chain to be used.

Step 4. Specify whether the interface will use clear text or MD5 authentication.

Step 5. Optionally configure key management.

In Example 6-9, a key chain named Tewa is configured at Taos. Key 1, the only key on the chain, has a password of Kachina; interface E0 then uses the key, with MD5 authentication, to validate updates from Laguna.

## Example 6-9. Taos's authentication configuration.

```
key chain Tewa
 key 1
 key-string Kachina
interface Ethernet 0
 ip rip authentication key-chain Tewa
 ip rip authentication mode md5
```

A key chain must be configured, even if there is only one key on it. Although any routers that will exchange authenticated updates must have the same password, the name of the key chain has significance only on the local router. Laguna, for instance, might have a key chain named Keres, but the key string must be Kachina to speak to Taos.

If the command ip rip authentication mode md5 is not added, the interface will use the default clear text authentication. Although clear text authentication may be necessary to communicate with some RIPv2 implementations, it is almost always wise to use the far more secure MD5 authentication whenever possible.

Key management is used to migrate from one authentication key to another. In Example 6-10, Laguna is configured to begin using the first key at 4:30 p.m. on July 1, 2004, for 12 hours (43200 seconds). The second key becomes valid at 4:00 a.m. on July 2, 2004, and will be used until 1:00 p.m. on December 31, 2004. The third key becomes valid at 12:30 p.m. on December 31, 2004, and will remain valid permanently after that.

## Example 6-10. Laguna's authentication configuration.

```
key chain Keres
 key 1
  key-string Kachina
  accept-lifetime 16:30:00 Jul 1 2004 duration 43200
  send-lifetime 16:30:00 Jul 1 2004 duration 43200
 key 2
  key-string Kiva
  accept-lifetime 04:00:00 Jul 2 2004 13:00:00 Dec 31 2004
  send-lifetime 04:00:00 Jul 2 2004 13:00:00 Dec 31 2004
 key 3
  key-string Koshare
  accept-lifetime 12:30:00 Dec 31 2004 infinite
  send-lifetime 12:30:00 Dec 31 2004 infinite
```

```
!
interface Ethernet0
 ip address 198.168.50.130 255.255.255.192
 ip rip authentication key-chain Keres
 ip rip authentication mode md5
```

As the configuration shows, the password that is accepted from other routers and the password that is used with transmitted messages are managed separately. Both the accept-lifetime and the send-lifetime commands must have a specified start time and may have either a specified duration or end time or the keyword infinite. The key numbers are examined from the lowest to the highest, and the first valid key is used.

Although this configuration uses a 30-minute overlap to compensate for differences in system times, it is highly recommended that a time synchronization protocol such as Network Time Protocol (NTP) be used with key management.[12]

PREV NEXT

# Configuring RIPng

The steps for configuring RIPng are the same as RIPv1 and v2: Create the routing process, enable the routing protocol on interfaces, and perform any desired customization. The commands, however, are very different. In fact, the creation of the routing process and enabling the routing protocol on the first interface is accomplished in one step, using an interface command. The case studies for RIPng will be similar to those of RIPv1 and v2 to illustrate the similarities in the processes and the differences in configuration.

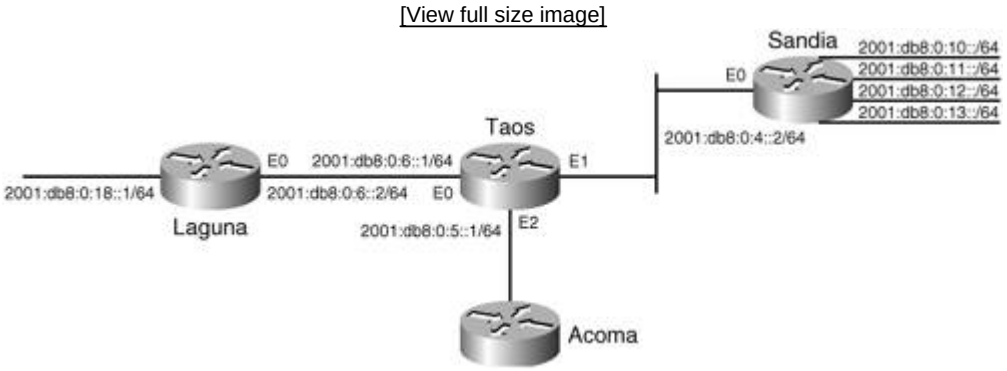
## Case Study: Basic RIPng Configuration

After IPv6 unicast routing is enabled on the router, only one command is required to enable RIPng for IPv6 routing. It is an interface command: ipv6 rip *process-name* enable. This creates the RIPng process called *process-name* and enables RIPng on the interface at the same time. The command is entered on each interface that will run RIPng.

Figure 6-15 shows that IPv6, RIPng, and some new Ethernet interfaces have been added to a portion of the network in Figure 6-14.

Figure 6-15. IPv6 addresses are configured on Laguna, Taos, and Sandia's interfaces that are to run RIPng.

[View full size image]




Example 6-11 shows Taos's configuration.

Example 6-11. Taos's IPv6 and RIPng configuration.

```
ipv6 unicast-routing
interface Ethernet0
 ipv6 address 2001:db8:0:6::1/64
 ipv6 rip bigMountain enable

interface Ethernet1
 ipv6 address 2001:db8:0:4::1/64
 ipv6 rip bigMountain enable

interface Ethernet2
```

```
ipv6 address 2001:db8:0:5::1/64
ipv6 rip bigMountain enable
```

The RIPng process is created and activated on an interface with a single interface command. This the difference between configuring RIPv1 and v2, and configuring RIPng. Recall that the RIPv1 and v2 configuration requires a global command (`router rip`) to create the process, and the `network <address>` statement to specify the interfaces on which to run the RIP process. With RIPng, when the interface command `ipv6 rip bigMountain enable` is configured on the first interface, the RIPng process called bigMountain is automatically created. In addition, IOS automatically creates an entry in the configuration (`ipv6 router rip bigMountain`) for the routing process.

The process name specified on Taos is bigMountain. Notice that the process name is specified on each interface. This name is relevant only to the local router, and is used to distinguish between multiple RIPng processes that might be running on the router. Different interfaces can each run a unique RIPng process, without exchanging information between these interfaces, or multiple processes might be running on a single interface. Only a single RIPv1 or RIPv2 process, on the other hand, can run on a router, and an interface either belongs to this process, or it doesn't run RIP.[13]

[13] MPLS VPNs allow multiple RIP processes to run on a single provider edge router. A single RIP process is configured for a given VPN. MPLS and VPNs are outside the scope of this book.

Example 6-12 shows Sandia's IPv6 route table. All the addresses in the network are listed because Taos is configured with a single RIPng process on each of its IPv6 interfaces.

## Example 6-12. Taos's IPv6 route table.

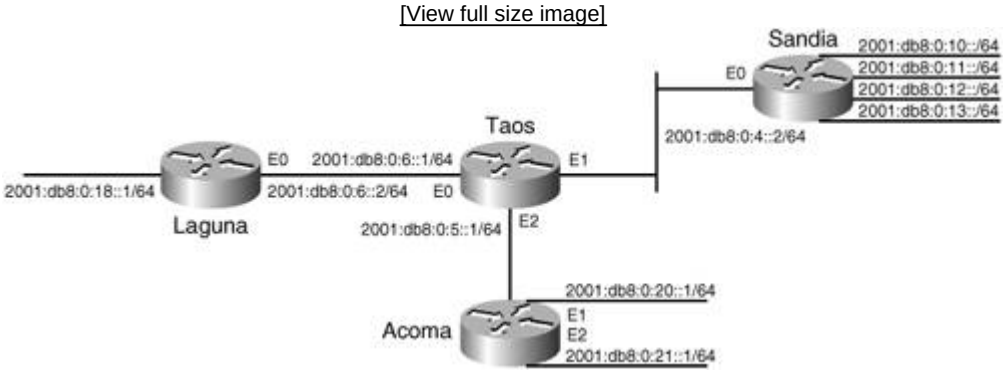
```
Sandia#show ipv6 route
IPv6 Routing Table - 16 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2001:DB8:0:4::/64 [0/0]
     via ::, Ethernet0
L   2001:DB8:0:4::2/128 [0/0]
     via ::, Ethernet0
R   2001:DB8:0:5::/64 [120/2]
     via FE80::205:5EFF:FE6B:50A1, Ethernet0
R   2001:DB8:0:6::/64 [120/2]
     via FE80::205:5EFF:FE6B:50A1, Ethernet0
C   2001:DB8:0:10::/64 [0/0]
     via ::, Ethernet1
L   2001:DB8:0:10:2B0:64FF:FE30:1DE0/128 [0/0]
     via ::, Ethernet1
C   2001:DB8:0:11::/64 [0/0]
     via ::, Ethernet2
L   2001:DB8:0:11:2B0:64FF:FE30:1DE0/128 [0/0]
     via ::, Ethernet2
C   2001:DB8:0:12::/64 [0/0]
     via ::, Ethernet3
L   2001:DB8:0:12:2B0:64FF:FE30:1DE0/128 [0/0]
     via ::, Ethernet3
C   2001:DB8:0:13::/64 [0/0]
     via ::, Ethernet4
L   2001:DB8:0:13:2B0:64FF:FE30:1DE0/128 [0/0]
     via ::, Ethernet4
R   2001:DB8:0:18::/64 [120/3]
```

```
     via FE80::205:5EFF:FE6B:50A1, Ethernet0
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
```

Two new Ethernet interfaces have been added to Acoma, as shown in Figure 6-16. New network policy states that IPv6 traffic in the network of Figure 6-16 should be segmented according to the following policy: Hosts on the LANs connected to Sandia access each other, and they also access devices on Ethernet1 connected to Acoma. Hosts connected to the Laguna router access the devic on Ethernet2 of Acoma. Sandia and Laguna hosts do not need to access each other. Acoma Ethernet1 and Ethernet 2 do not access each other.

## Figure 6-16. Acoma has been added to the network, with two IPv6-enabled Ethernet user LAN segments.

[View full size image]




The configuration in Example 6-13 is loaded into Taos.

## Example 6-13. Taos's configuration to support network policy to segment IPv6 network traffic by creating multiple RIPng routing processes.

```
Interface Ethernet0
 ipv6 rip bigMountain enable
Interface Ethernet1
 no ipv6 rip bigMountain enable
 ipv6 rip smallMountain enable
interface Ethernet2
 ipv6 rip bigMountain enable
 ipv6 rip smallMountain enable

ipv6 router rip smallMountain
 port 527 multicast ff02::9
```

Interface Ethernet 0, the link to Laguna, still belongs to the bigMountain RIPng process. A new process, smallMountain, is now enabled on Ethernet 1, the link to Sandia, and bigMountain has been disabled on the interface. Both processes are enabled on Ethernet2, connecting to Acoma. N two RIPng routing processes can use the same UDP port number if both processes are going to be active on a single interface. By changing the port number on the smallMountain process, Taos sends updates for smallMountain to UDP port 527, and updates for the bigMountain process to the

default RIPng port, UDP 521. Receiving routers use the port number to distinguish which routing process a received update belongs to. A RIPng router running a single routing process, receiving updates from more then one RIPng process running on a router using identical UDP port numbers, will process each update. If there is conflicting information about an address in both updates, such as a differing metric, the receiving route table may be changing with each update. If a receiving router is running multiple RIPng processes and the same UDP port number is used on both, the receiving router will not be able to distinguish which RIPng process should receive the update. Changing the UDP port number on the second and subsequent RIPng process eliminates these problems. All routers and hosts that need to participate in the RIPng process with the modified UDP port number must use the same UDP port number. Because routers and other TCP/IP devices use port numbers identify which UDP process the packets need to be forwarded to, and RIPng updates are multicast to all RIPng routers, the new UDP port number must not be the same as any other process running on any RIPng router. The smallMountain process is configured to use port 527 in this example. The port number must be changed on the routers exchanging RIPng updates with the smallMountain RIPng process. Sandia and Acoma's configuration changes follow in Example 6-14 and Example 6-15, respectively.

### Example 6-14. Sandia's RIPng configuration with a modified UDP port number.

```
ipv6 router rip smallMountain
  port 527 multicast-group FF02::9
```

### Example 6-15. Acoma's RIPng configuration with a modified UDP port number.

```
interface Ethernet0
 ipv6 address 2001:DB8:0:5::2/64
 ipv6 rip hill enable
 ipv6 rip summit enable
interface Ethernet1
 ipv6 address 2001:DB8:0:20::/64
 ipv6 rip hill enable
interface Ethernet2
 ipv6 address 2001:DB8:0:21::/64
 ipv6 rip summit enable
ipv6 router rip hill
  port 527 multicast-group FF02::9
```

Notice that the process names on Acoma are not the same as those on Taos. RIPng process names are relevant only to the local router. They are not exchanged in route updates. Acoma's Ethernet0 is connected to Taos's Ethernet2. It is running two routing processes: hill and summit. hill is also running on interface Ethernet1. Ethernet1's IPv6 address is advertised to Taos using the modified UDP port number 527. This is associated with the smallMountain RIPng process on Taos, and therefore, the IPv6 address gets advertised to Sandia, not Laguna.

Example 6-16 displays information about the RIPng routing processes running on Taos. The command show ipv6 rip displays each process individually. Notice that the default maximum number of parallel paths is 16, and the timer values have not been modified from the default.

### Example 6-16. *show ipv6 rip* displays information about each RIPng process running on a router.

```
Taos#show ipv6 rip
RIP process "bigMountain", port 521, multicast-group FF02::9, pid 104
```

```
     Administrative distance is 120. Maximum paths is 16
     Updates every 30 seconds, expire after 180
     Holddown lasts 0 seconds, garbage collect after 120
     Split horizon is on; poison reverse is off
     Default routes are not generated
     Periodic updates 1078, trigger updates 5
  Interfaces:
    Ethernet2
    Ethernet0
  Redistribution:
    None
RIP process "smallMountain", port 527, multicast-group FF02::9, pid 117
     Administrative distance is 120. Maximum paths is 16
     Updates every 30 seconds, expire after 180
     Holddown lasts 0 seconds, garbage collect after 120
     Split horizon is on; poison reverse is off
     Default routes are not generated
     Periodic updates 1080, trigger updates 5
  Interfaces:
    Ethernet1
    Ethernet2
  Redistribution:
    None
```

The process bigMountain is running on interfaces Ethernet0 and Ethernet2. Interfaces Ethernet1 and Ethernet2 belong to the smallMountain process, using UDP port number 527.

The IPv6 route table shows that Taos has installed routes from both processes into its route table (Example 6-17).

## Example 6-17. IPv6 route table displays all known routes from each active RIPng process.

```
Taos#show ipv6 route
IPv6 Routing Table - 15 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2001:DB8:0:4::/64 [0/0]
     via ::, Ethernet1
L   2001:DB8:0:4::1/128 [0/0]
     via ::, Ethernet1
C   2001:DB8:0:5::/64 [0/0]
     via ::, Ethernet2
L   2001:DB8:0:5::1/128 [0/0]
     via ::, Ethernet2
C   2001:DB8:0:6::/64 [0/0]
     via ::, Ethernet0
L   2001:DB8:0:6::1/128 [0/0]
     via ::, Ethernet0
R   2001:DB8:0:10::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:11::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:12::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
```

```
R   2001:DB8:0:13::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:18::/64 [120/2]
     via FE80::204:C1FF:FE50:F1C0, Ethernet0
R   2001:DB8:0:20::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Ethernet2
R   2001:DB8:0:21::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Ethernet2
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
```

Taking a look at Laguna and Sandia's route tables in Example 6-18 and Example 6-19, you can se that each router learns about the routes belonging to the appropriate RIPng process. Taos does nc forward smallMountain addresses into the bigMountain process, and vice versa.

## Example 6-18. IPv6 route table displays Laguna's routes learned from the bigMountain RIPng process on Taos.

```
Laguna#show ipv6 route rip
IPv6 Routing Table - 8 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R   2001:DB8:0:5::/64 [120/2]
     via FE80::205:5EFF:FE6B:50A0, Ethernet0
R   2001:DB8:0:21::/64 [120/3]
     via FE80::205:5EFF:FE6B:50A0, Ethernet0
```

## Example 6-19. IPv6 route table displays Sandia's routes learned from the smallMountain RIPng process on Taos.

```
Sandia#show ipv6 route rip
IPv6 Routing Table - 14 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R   2001:DB8:0:5::/64 [120/2]
     via FE80::205:5EFF:FE6B:50A1, Ethernet0
R   2001:DB8:0:20::/64 [120/3]
     via FE80::205:5EFF:FE6B:50A1, Ethernet0
```

Example 6-17 shows that Taos has learned four addresses from Sandia on Ethernet1, which belongs to the smallMountain RIPng process: 2001:db8:0:10::/64, 2001:db8:0:11::/64, 2001:db8:0:12::/64, and 2001:db8:0:13::/64. 2001:db8:0:5::/64, connected to Ethernet2, is advertised into both the smallMountain and the bigMountain processes. Laguna's route table in Example 6-18 shows that no smallMountain routes have been installed.

Debugging RIPng on Taos (Example 6-20) shows the RIPng updates that are sent out and receivec into each interface for each process.

Example 6-20. *debug ipv6 rip* displays the RIPng updates on each interface for each active process.

```
Taos#debug ipv6 rip
RIP Routing Protocol debugging is on
Taos#
RIPng: Sending multicast update on Ethernet2 for bigMountain
        src=FE80::205:5EFF:FE6B:50A0
        dst=FF02::9 (Ethernet2)
        sport=521, dport=521, length=72
        command=2, version=1, mbz=0, #rte=3
        tag=0, metric=1, prefix=2001:DB8:0:5::/64
        tag=0, metric=1, prefix=2001:DB8:0:6::/64
        tag=0, metric=2, prefix=2001:DB8:0:18::/64
RIPng: Sending multicast update on Ethernet0 for bigMountain
        src=FE80::205:5EFF:FE6B:50A0
        dst=FF02::9 (Ethernet0)
        sport=521, dport=521, length=72
        command=2, version=1, mbz=0, #rte=3
        tag=0, metric=1, prefix=2001:DB8:0:5::/64
        tag=0, metric=1, prefix=2001:DB8:0:6::/64
        tag=0, metric=2, prefix=2001:DB8:0:21::/64
RIPng: response received from FE80::2B0:64FF:FE30:1DE0 on Ethernet1 for
  smallMountain
        src=FE80::2B0:64FF:FE30:1DE0 (Ethernet1)
        dst=FF02::9
        sport=527, dport=527, length=112
        command=2, version=1, mbz=0, #rte=5
        tag=0, metric=1, prefix=2001:DB8:0:4::/64
        tag=0, metric=1, prefix=2001:DB8:0:10::/64
        tag=0, metric=1, prefix=2001:DB8:0:11::/64
        tag=0, metric=1, prefix=2001:DB8:0:12::/64
        tag=0, metric=1, prefix=2001:DB8:0:13::/64
RIPng: response received from FE80::204:C1FF:FE50:E700 on Ethernet2 for
  smallMountain
        src=FE80::204:C1FF:FE50:E700 (Ethernet2)
        dst=FF02::9
        sport=527, dport=527, length=52
        command=2, version=1, mbz=0, #rte=2
        tag=0, metric=1, prefix=2001:DB8:0:5::/64
        tag=0, metric=1, prefix=2001:DB8:0:20::/64
RIPng: response received from FE80::204:C1FF:FE50:F1C0 on Ethernet0 for bigMountai
        src=FE80::204:C1FF:FE50:F1C0 (Ethernet0)
        dst=FF02::9
        sport=521, dport=521, length=52
        command=2, version=1, mbz=0, #rte=2
        tag=0, metric=1, prefix=2001:DB8:0:6::/64
        tag=0, metric=1, prefix=2001:DB8:0:18::/64
RIPng: response received from FE80::204:C1FF:FE50:E700 on Ethernet2 for bigMountai
        src=FE80::204:C1FF:FE50:E700 (Ethernet2)
        dst=FF02::9
        sport=521, dport=521, length=52
        command=2, version=1, mbz=0, #rte=2
        tag=0, metric=1, prefix=2001:DB8:0:5::/64
        tag=0, metric=1, prefix=2001:DB8:0:21::/64
RIPng: Sending multicast update on Ethernet1 for smallMountain
        src=FE80::205:5EFF:FE6B:50A1
        dst=FF02::9 (Ethernet1)
        sport=527, dport=527, length=72
```

```
        command=2, version=1, mbz=0, #rte=3
        tag=0, metric=1, prefix=2001:DB8:0:4::/64
        tag=0, metric=1, prefix=2001:DB8:0:5::/64
        tag=0, metric=2, prefix=2001:DB8:0:20::/64
RIPng: Sending multicast update on Ethernet2 for smallMountain
        src=FE80::205:5EFF:FE6B:50A0
        dst=FF02::9 (Ethernet2)
        sport=527, dport=527, length=132
        command=2, version=1, mbz=0, #rte=6
        tag=0, metric=1, prefix=2001:DB8:0:4::/64
        tag=0, metric=1, prefix=2001:DB8:0:5::/64
        tag=0, metric=2, prefix=2001:DB8:0:10::/64
        tag=0, metric=2, prefix=2001:DB8:0:11::/64
        tag=0, metric=2, prefix=2001:DB8:0:12::/64
        tag=0, metric=2, prefix=2001:DB8:0:13::/64
```

smallMountain updates are sent out Ethernet1, to Sandia, and Ethernet2, to Acoma. bigMountain updates are sent out Ethernet0, to Laguna, and Ethernet2, to Acoma. Two updates are received o Ethernet2 from Acoma. One uses the default UDP port 521. The other uses UDP port 527, the UDP port associated with smallMountain. The update that uses port 527 includes the prefix 2001:DB8:0:20::/64. This is the prefix assigned to Ethernet1 on Acoma, the prefix which Sandia i permitted to access. The update received on Ethernet2 using UDP port 521 includes prefix 2001:DB8:0:21::/64. This prefix is assigned to Ethernet2 on Acoma and is distributed into the bigMountain process on Taos, and sent to Laguna.

## Case Study: RIPng Process Customization

Some customization to the routing process is achieved in a way similar to RIPv1 and v2. The globa configuration command ipv6 router rip *process_name* lets you configure global parameters to th RIPng process.

The administrative distance, the maximum number of paths the routing process will use for load sharing, and RIP timers were discussed in Chapter 5. These parameters are used the same way fo RIPng as they are for RIPv1 and v2. RIPng's default administrative distance is 120, the same as RIPv1 and v2. The default maximum number of paths over which RIPng will load share is 16. RIPn can load share on up to 64 paths. The timer parameters, although not all the values, are the same as for RIPv1 and v2: update every 30 seconds, expire after 180 seconds, holddown is 0, garbage collection is 120 seconds.

A configuration for Taos that modifies the distance, maximum-paths, and timers is shown in Example 6-21.

### Example 6-21. Taos's configuration modifies the administrative distance the maximum number of paths for load sharing, and the protocol times.

```
ipv6 router rip bigMountain
  timers 10 30 30 60
  maximum-paths 8
  distance 200
```

Example 6-22 shows the output of show ipv6 rip and displays the new parameter values.

### Example 6-22. The *show ipv6 rip* command on Taos shows the modified RIPng values.

```
Taos#show ipv6 rip
RIP process "bigMountain", port 521, multicast-group FF02::9, pid 104
     Administrative distance is 200. Maximum paths is 8
     Updates every 10 seconds, expire after 30
     Holddown lasts 30 seconds, garbage collect after 60
     Split horizon is on; poison reverse is off
     Default routes are not generated
     Periodic updates 2513, trigger updates 7
  Interfaces:
    Ethernet2
    Ethernet0
  Redistribution:
    None
RIP process "smallMountain", port 527, multicast-group FF02::9, pid 122
     Administrative distance is 120. Maximum paths is 16
     Updates every 30 seconds, expire after 180
     Holddown lasts 0 seconds, garbage collect after 120
     Split horizon is on; poison reverse is off
     Default routes are not generated
     Periodic updates 2511, trigger updates 0
  Interfaces:
    Ethernet2
    Ethernet1
  Redistribution:
    None
Taos#
```

Compare the values in the bigMountain process with those in the smallMountain process, which sti has the default values. Be careful modifying the timers and the administrative distance. All routers running the same RIPng process must have the same timer values. The administrative distance is only relevant to the local router, but consider carefully the consequence of changing the administrative distance if multiple routing protocols or routing processes are running on the router and addresses are being learned from multiple routing processes. Changing the administrative distance changes the degree of preference for the routing process. An address learned via a process with a lower distance will get inserted into the routing table. If the same address is learne via a higher administrative distance process, the address will get inserted into the routing table only if the first entry fails. The configuration in Example 6-23 changes the values back to the defaults.

### Example 6-23. Taos's RIPng parameters are changed back to the defaul values in this configuration.

```
ipv6 router rip bigMountain
 no timers 10 30 30 60
 no maximum-paths 8
 no distance 200
```

## Case Study: Metric Manipulation

As with RIPv1 and v2, RIPng metrics are adjusted by adding an offset. RIPng does not, however, increment the metrics of each entry in a list of prefixes. RIPng modifies the hop count that is associated with an interface. This increments the metric of every prefix that is advertised to the router via this interface by the amount of the hop count configured on the interface. By default, RIPng adds a single hop to the metric of prefixes advertised by a neighboring router. Example 6-2 shows the RIPng route table on Taos and a RIPng update received from Sandia. Notice that each o the metrics in the route table has a value of two. Each RIPng route in the table is directly connecte

to a router that is one hop away, either Sandia or Laguna. Sandia and Laguna advertise their prefixes with a metric of one. Taos adds a hop count of one to each of the received metrics. To change the value that Taos adds to the received metric, use the metric-offset command, as in Example 6-25.

### Example 6-24. RIPng route table and *debug ipv6 rip* showing a received update illustrates the metric offset added to a received metric.

```
Taos#show ipv6 route rip
IPv6 Routing Table - 15 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R   2001:db8:0:18::/64[120/2]
     via FE80::204:C1FF:FE50:F1C0, Ethernet0
R   2001:db8:0:10::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:db8:0:11::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:db8:0:12::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:db8:0:13::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:20::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Ethernet2
R   2001:DB8:0:21::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Ethernet2
Taos#debug ipv6 rip
RIP Routing Protocol debugging is on
   RIPng: response received from FE80::2B0:64FF:FE30:1DE0 on Ethernet1 for
     smallMountain
        src=FE80::2B0:64FF:FE30:1DE0 (Ethernet1)
        dst=FF02::9
        sport=521, dport=521, length=112
        command=2, version=1, mbz=0, #rte=4
        tag=0, metric=1, prefix=2001:db8:0:10::/64
        tag=0, metric=1, prefix=2001:db8:0:11::/64
        tag=0, metric=1, prefix=2001:db8:0:12::/64
        tag=0, metric=1, prefix=2001:db8:0:13::/64
```

### Example 6-25. The RIPng metric-offset is increased to three on Taos's Ethernet1 interface.

```
interface Ethernet1
 ipv6 rip smallMountain metric-offset 3
```

Example 6-26 shows the new route table on Taos. Taos has added a hop count of 3 to each of the prefixes the router received on Ethernet1.

### Example 6-26. RIPng route table after the metric offset has been modified on the receiving interface shows a higher metric value for each prefix learned via the modified interface.

```
Taos#show ipv6 route rip
IPv6 Routing Table - 15 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R   2001:DB8:0:10::/64 [120/4]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:11::/64 [120/4]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:12::/64 [120/4]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:13::/64 [120/4]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:18::/64 [120/2]
     via FE80::204:C1FF:FE50:F1C0, Ethernet0
R   2001:DB8:0:20::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Ethernet2
R   2001:DB8:0:21::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Ethernet2
```

This command allows you to adjust the hop count associated with a given link. All addresses received by the adjusted interface will have the adjusted metric. By contrast, RIPv1 and v2 allow you to add the offset to a subset (or all) of the addresses received by the interface before they are added to the route table, or the offset can be added to the addresses just before they are advertised out of the interface.

## Case Study: Route Summarization

As with RIPv2, RIPng routes can be summarized before they are advertised to neighbors. Sandia advertises routes for 2001:DB8:0:10::/64, 2001:DB8:0:11::/64, 2001:DB8:0:12::/64, and 2001:DB8:0:13::/64. The routes can be summarized into a single entry: 2001:DB8: 0:10::/62. Sandia's configuration is in Example 6-27.

### Example 6-27. Sandia is configured to summarize advertised RIPng addresses.

```
interface Ethernet0
 ipv6 address 2001:DB8:0:4::2/64
 ipv6 rip bigMountain enable
 ipv6 rip bigMountain summary-address 2001:DB8:0:10::/62
```

The more specific prefixes are automatically suppressed when an encompassing summary address is configured. Example 6-28 shows Taos's new route table.

### Example 6-28. Four consecutive prefixes with a length of 64 bits have been summarized into a single entry with a length of 62 bits.

```
Taos#show ipv6 route rip
IPv6 Routing Table - 12 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
```

```
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R   2001:DB8:0:10::/62 [120/4]
     via FE80::2B0:64FF:FE30:1DE0, Ethernet1
R   2001:DB8:0:18::/64 [120/2]
     via FE80::204:C1FF:FE50:F1C0, Ethernet0
R   2001:DB8:0:20::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Ethernet2
R   2001:DB8:0:21::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Ethernet2
```

PREV NEXT

# Troubleshooting RIPv2 and RIPng

Two configuration problems common to RIPv2 are mismatched versions and misconfigured authentication. Both difficulties are easy to discover with debugging, as Example 6-29 shows.

### Example 6-29. Debugging reveals mismatched versions and misconfigured authentication.

```
Jemez#debug ip rip events
RIP event debugging is on
Jemez#
RIP: ignored v1 packet from 172.25.150.249 (illegal version)
RIP: ignored v2 packet from 172.25.150.249 (invalid authentication)
Jemez#
```

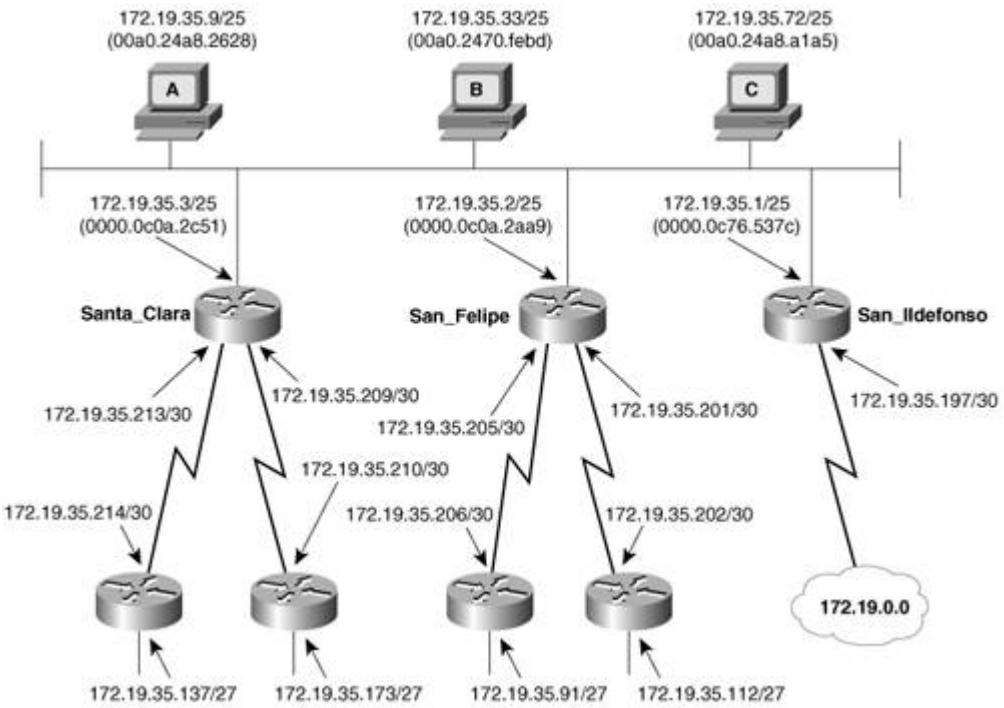
A more likely source of trouble with RIPv2 or RIPng, or any classless routing protocol, is a misconfigured variable-length subnet mask. VLSM is not difficult, but if a VLSM scheme is not designed and managed carefully it can cause some unusual routing difficulties.

## Case Study: Misconfigured VLSM

Host C in Figure 6-17 cannot communicate across the network, and it cannot even ping the other hosts or routers on the local data link. Hosts A and B have no communication problems with each other or with any other host across the network, but they cannot communicate with C. All hosts are configured to use 172.19.35.1 as the default gateway address.

### Figure 6-17. Hosts A and B can communicate across the network, but host C cannot.

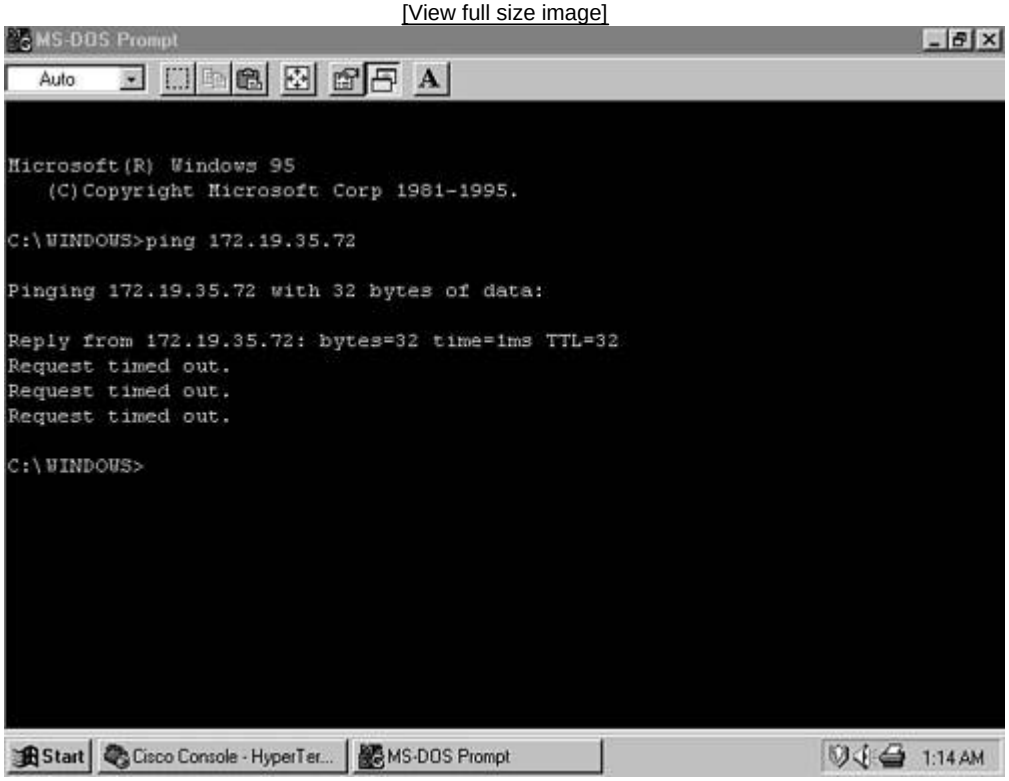
[View full size image]




When an attempt is made to ping host C from host A or B, the first ping is successful and subsequent pings fail (Figure 6-18). Because at least one ICMP Echo Request packet reached C, and at least one Echo Reply packet was returned, the problem probably is not related to the hardware or data link.

Figure 6-18. When host B pings host C, the first ping is successful and subsequent pings fail.

[View full size image]

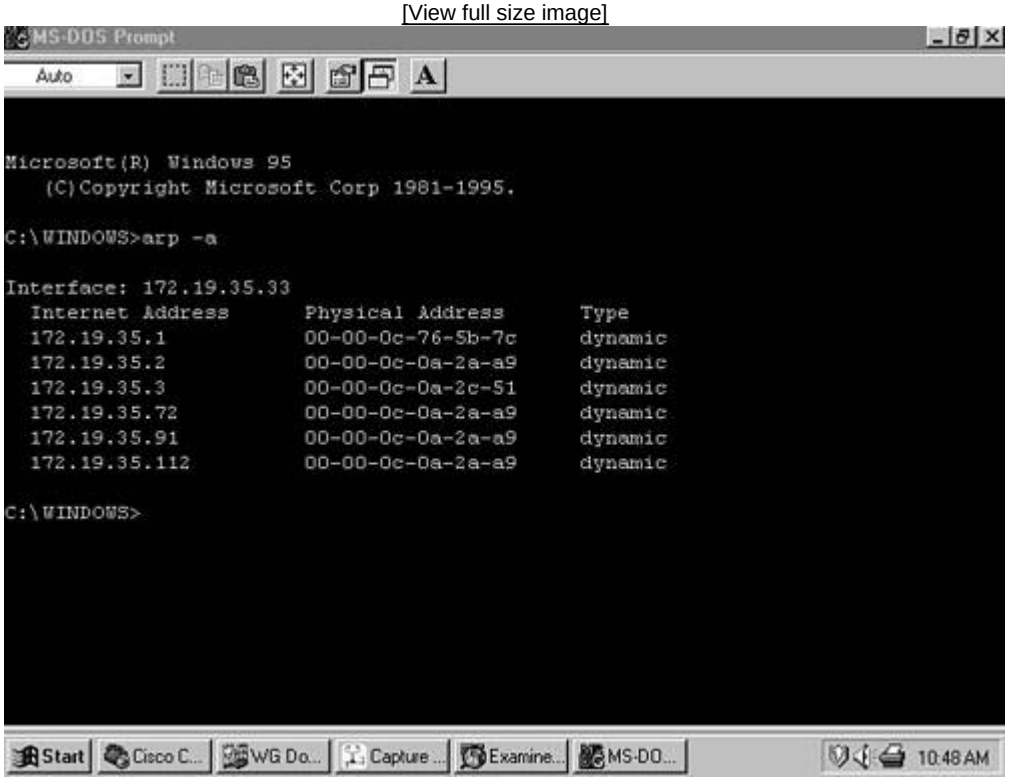


The strange ping behavior leads to the hypothesis that after the first successful packet, subsequent packetseither the Echo Requests from B or the Echo Replies from Care somehow being misdirected. Because this behavior is happening on the local data link, the Address Resolution Protocol (ARP) caches should be examined.

Example 6-30 and Figure 6-19 show the ARP caches for C and B, respectively. The suspicions about ARP are confirmed here. C's cache contains the correct MAC address for B (00a0.2470.febd), but B's cache has a MAC address of 0000.0c0a.2aa9 associated with C. A closer look at both caches shows that 0000.0c0a.2aa9 is the MAC address of San_Felipe's locally attached interface. This information can be deduced because the same MAC address is mapped to IPv4 address 172.19.35.2 and to the IPv4 addresses reachable via that router.

Figure 6-19. Host B's ARP cache shows that C's IPv4 address is mapped to the MAC address of San_Felipe's interface 172.19.35.2.

[View full size image]




Example 6-30. Host C's ARP cache shows the correct MAC address associated with all addresses.

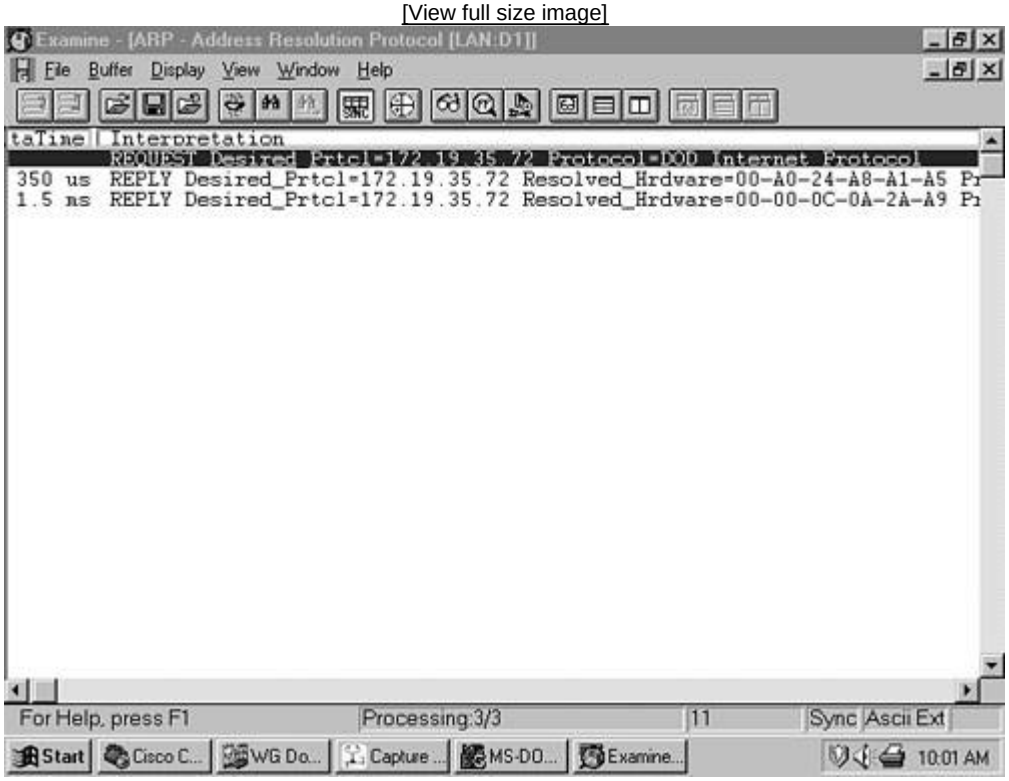
```
Linux 1.2.13 (Zuni.pueblo.com) (ttyp0)
Zuni login: root
Password:
Last login: Sat Nov 29 11:21:57 on tty1
Linux 1.2.13.
Zuni:~# arp -a
Address          HW type            HW address            Flags     Mask
172.19.35.112    10Mbps Ethernet    00:00:0C:0A:2A:A9     C         *
172.19.35.1      10Mbps Ethernet    00:00:0C:76:5B:7C     C         *
172.19.35.33     10Mbps Ethernet    00:A0:24:70:FE:BD     C         *
172.19.35.2      10Mbps Ethernet    00:00:0C:0A:2A:A9     C         *
172.19.35.3      10Mbps Ethernet    00:00:0C:0A:2C:51     C         *
```

```
172.19.35.9     10Mbps Ethernet    00:A0:24:A8:26:28     C          *
172.19.35.91    10Mbps Ethernet    00:00:0C:0A:2A:A9     C          *
Zuni:~#
```

The ping results begin to make sense. B broadcasts an ARP Request for 172.19.35.72. C sends an ARP Reply, and B sends its first ping correctly. In the meantime, San_Felipe has received the ARP Request and apparently believes that it has a route to 172.19.35.72. It responds with a proxy ARP (later than C because it has to perform a route lookup first), which causes B to overwrite C's MAC address. Subsequent Echo Request packets are sent to San_Felipe, where they are routed off the local data link and lost. A protocol analyzer attached to the Ethernet proves the point (Figure 6-20).

Figure 6-20. A protocol analyzer, filtering for ARP packets, shows B's ARP request to C and replies from both host C (00a0.24a8.a1a5) and router San_Felipe (0000.0c0a.2aa9).

[View full size image]




If you know that the trouble is a routing problem, it remains only to find the cause. First, the subnet addresses for each data link should be determined (Figure 6-21). Next, C's IPv4 address should be compared with all the subnets reachable via San_Felipe, in binary, to find any conflicts. Table 6-5 shows the addresses with the subnet bits of the last octet in bold.

Figure 6-21. When analyzing any addressing scheme, and especially a VLSM design, the subnets for every data link should be determined so that conflicts and overlaps may be discovered.

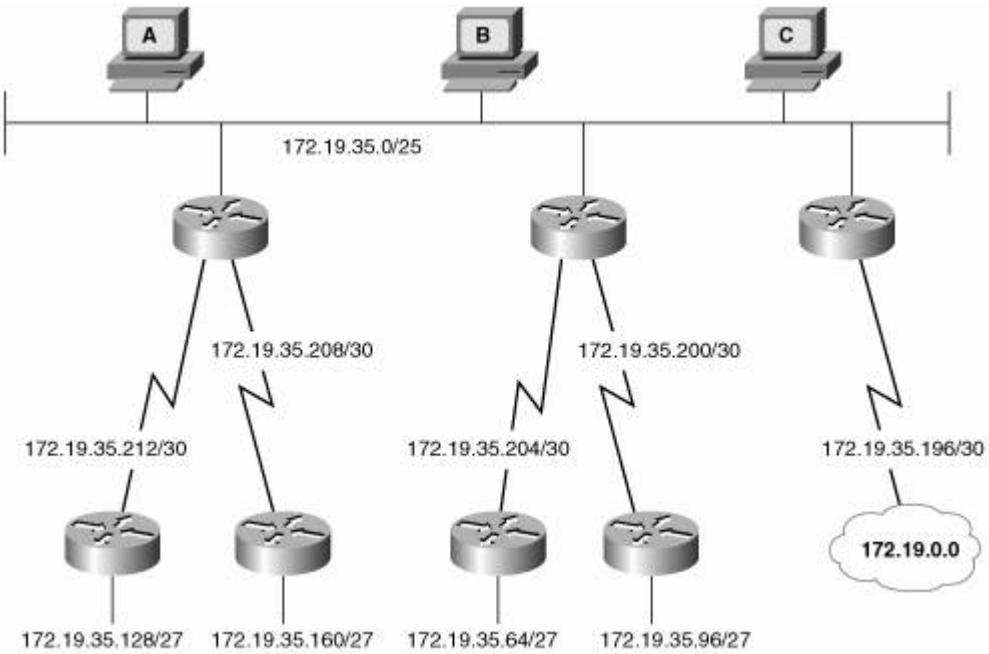



Table 6-5. C's IPv4 addresses with subnet bits of the last octet highlighted.

| Binary Representation | Dotted Decimal |
|---|---|
| 10101100000100110010001101001000 | 172.19.35.72/25 |
| 10101100000100110010001100000000 | 172.19.35.0/25 |
| 10101100000100110010001101000000 | 172.19.35.64/27 |
| 10101100000100110010001101100000 | 172.19.35.96/27 |
| 10101100000100110010001111001000 | 172.19.35.200/30 |
| 10101100000100110010001111001100 | 172.19.35.204/30 |

A comparison shows that the first three bits of 172.19.35.72/25 match subnet 172.19.35.64/27. San_Felipe has routes to both 172.19.35.0/25 and to 172.19.35.64/27 (Example 6-31). When it receives a packet destined for host C, it will be able to match one bit to subnet 172.19.35.0/25 but three bits to 172.19.35.64/27. As a result, the router will choose the more specific subnet and route the packet off the local data link and into oblivion.

## Example 6-31. San_Felipe has routes to both 172.19.35.0/25 and to 172.19.35.64/27; the second route is a better match of host C's address than is the first route.

```
San_Felipe#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default,
       U - per-user static route
Gateway of last resort is 172.19.35.1 to network 0.0.0.0
     172.19.0.0/16 is variably subnetted, 10 subnets, 3 masks
```

```
R      172.19.35.128/27 [120/1] via 172.19.35.3, 00:00:07, Ethernet0
R      172.19.35.160/27 [120/1] via 172.19.35.3, 00:00:08, Ethernet0
R      172.19.35.212/30 [120/1] via 172.19.35.3, 00:00:08, Ethernet0
R      172.19.35.208/30 [120/1] via 172.19.35.3, 00:00:08, Ethernet0
C      172.19.35.204/30 is directly connected, Serial0
C      172.19.35.200/30 is directly connected, Serial1
R      172.19.35.196/30 [120/1] via 172.19.35.1, 00:00:17, Ethernet0
C      172.19.35.0/25 is directly connected, Ethernet0
R      172.19.35.64/27 [120/1] via 172.19.35.206, 00:00:11, Serial0
R      172.19.35.96/27 [120/1] via 172.19.35.202, 00:00:23, Serial1
R* 0.0.0.0/0 [120/1] via 172.19.35.1, 00:00:18, Ethernet0
San_Felipe#
```

The solution to this trouble is to readdress either host C or subnet 172.19.35.64. This step sounds easy on paper. In real life, it may involve some difficult decisions, as it did for the client on whose network this case study is based.

Table 6-6 shows all the subnets of 172.19.35.0, based on a 27-bit mask. The intention was to combine the first four subnets into a single subnet with a 25-bit mask to accommodate up to 85 hosts on the "backbone" Ethernet. This decision is valid because the grouping will use all the subnets whose first bit is zero; no other address can cause a conflict. Next, 172.19.35.192/27 is subnetted with a 30-bit mask to be used on the serial links. Again, this design decision is valid. Subnets 172.19.35.128/27 and 172.19.35.160 are used as is. The error occurred when subnets 172.19.35.64/27 and 172.19.35.96/27 were selected to be used on two "remote" networks. These subnets had already been spoken for.

### Table 6-6. A 27-bit subnet mask is applied to subnet 172.19.35.0.

| Binary Representation | Dotted Decimal |
|---|---|
| 11111111111111111111111111111111 | 255.255.255.224 |
| 10101100000100110010001100000000 | 172.19.35.0/27 |
| 10101100000100110010001100100000 | 172.19.35.32/27 |
| 10101100000100110010001101000000 | 172.19.35.64/27 |
| 10101100000100110010001101100000 | 172.19.35.96/27 |
| 10101100000100110010001110000000 | 172.19.35.128/27 |
| 10101100000100110010001110100000 | 172.19.35.160/27 |
| 10101100000100110010001111000000 | 172.19.35.192/27 |
| 10101100000100110010001111100000 | 172.19.35.224/27 |

The difficult decision is in deciding whether to re-address the backbone, giving up some address space there, or to re-address the two remote subnets and give up address space on each of them. The second option was chosen, using a 28-bit mask to divide 172.19.35.224/27 into two subnets for the remote sites.

PREV NEXT

# Looking Ahead

Although RIPv2 presents some decided improvements over RIPv1, it is still limited to a maximum of 15 hops and, therefore, to small networks. Chapter 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)," Chapter 8, "OSPFv2," and Chapter 10, "Integrated IS-IS," examine three protocols that can be used in much larger networks and with which design strategies such as VLSM become very powerful tools for controlling them.

# Summary Table: Chapter 6 Command Review

| Command | Description |
|---|---|
| accept-lifetime *start-time* {infinite \| *end-time*\| duration *seconds*} | Specifies the time period during which the authentication key on a key chain is received as valid |
| auto-summary | Toggles the automatic summarization of routes on network boundaries |
| debug ip rip [events] | Enables the display of messages on RIP transactions |
| debug ipv6 rip | Enables the display of RIPng transactions |
| ip classless | Enables the forwarding of packets on the route for which the router can find the best match without consideration of the class of the destination address |
| ip rip authentication key-chain *name-of-chain* | Enables RIPv2 authentication on an interface and specifies the name of the key chain to be used |
| ip rip authentication mode {text \| md5} | Specifies whether an interface will use clear text or MD5 authentication |
| ip rip receive version [1] [2] | Specifies the version or versions of RIP messages that an interface will accept |
| ip rip send version [1] [2] | Specifies the version or versions of RIP messages that an interface will send |
| ip split-horizon | Toggles the split horizon function on an interface |
| ip subnet-zero | Allows the use of all-zeros subnets for interface addresses and routing updates |
| ipv6 rip *process-name* enable | Enables the IPv6 RIPng process with the given name on the interface on which this command is issued |
| ipv6 rip *process-name* metric-offset *value* | Modifies the metric associated with an inbound interface |
| ipv6 rip *process-name* summary-address *prefix* | Summarizes longer prefix length prefixes into shorter prefixes |
| ipv6 router rip *process-name* | Enables the IPv6, RIPng process on a router, and enters the configuration mode to change global RIPng parameters |
| port *port-num* multicast *multicast-value* | Changes the UDP port number and/or multicast address for a RIPng process |
| key *number* | Specifies a key on a key chain |
| key chain *name-of-chain* | Specifies a group of keys |
| key-string *text* | Specifies the authentication string, or password, used by a key |
| network *network-number* | Specifies the network address of one or more directly connected interfaces on which IGRP, EIGRP, or RIP processes should be enabled |

| Command | Description |
| --- | --- |
| passive-interface *type number* | Disables the transmission of routing updates on an interface |
| router rip | Enables the RIP routing process on a router |
| send-lifetime *start-time* {infinite \| *end-time*\| duration *seconds*} | Specifies the time period during which the authentication key on a key chain may be sent |
| show ip route [*address* [*mask*]] [*protocol* [*process-ID*]] | Displays the current routing table as a whole or by entry |
| show ipv6 rip | Displays the RIPng protocol information |
| show ipv6 route rip | Displays routes installed into the IPv6 route table from the RIPng process |
| version | Specifies the version of the RIP routing process |

PREV NEXT

# Recommended Reading

Malkin, G. "RIP Version 2." RFC 2453, November 1998.

Malkin, G., and R. Minnear. "RIPng for IPv6." RFC 2080, January 1997.

# Review Questions

1. Which three fields are new to the RIPv2 message format?
2. Besides the extensions defined by the three fields of question 1, what are the other two major changes from RIPv1?
3. What is the multicast address used by RIPv2? What is the advantage of multicasting messages over broadcasting them?
4. What is the purpose of the Route Tag field in the RIPv2 message?
5. What is the purpose of the Next Hop field?
6. What is the UDP port number used by RIPv2?
7. What is the UDP port number used by RIPng?
8. Which one feature must a routing protocol have to be a classless routing protocol?
9. Which one feature must a routing protocol have to use VLSM?
10. Which two types of authentication are available with the Cisco RIPv2? Are they both defined in RFC 2453?

# Configuration Exercises

1. In the example of Figure 6-12, router Taos was configured to send both Version 1 and Version 2 updates so that the *routed* process in the Linux host Pojoaque would understand the updates from Taos. Is there another way to configure Taos besides using the `ip rip send version` command?

2. A network has been assigned the address 192.168.100.0. Subnet this address to meet the following requirements:

   One subnet with 50 hosts

   Five subnets with 10 hosts

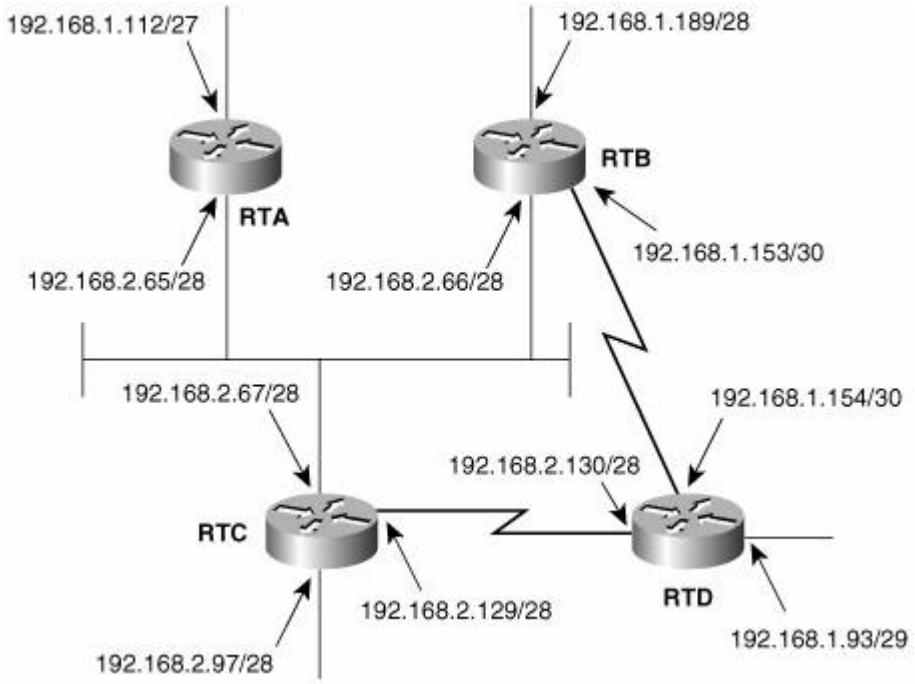
   One subnet with 25 hosts

   Four subnets with 5 hosts

   Ten serial links

3. Configure the four routers in Figure 6-22 to run RIP. RTC is running IOS 10.3 and for corporate policy reasons cannot be upgraded, therefore can only run RIPv1.

Figure 6-22. The network for Configuration Exercises 3 through 6.




4. Configure RTB and RTD in Figure 6-22 to authenticate the RIP updates being exchanged over the serial link.

5 Configure RTB and RTD in Figure 6-22 to change to a new authentication key three days from the time the key in Configuration Exercise 4 went into effect. Have the new key stay in effect for 10 hours and then have the routers change to yet another key.

6 Configure RTA and RTB in Figure 6-22 to run RIPng. The following IPv6 prefixes are to be assigned to interfaces:

RTA:

2001:DB8:0:1::/64
2001:DB8:0:2::/64

RTB:

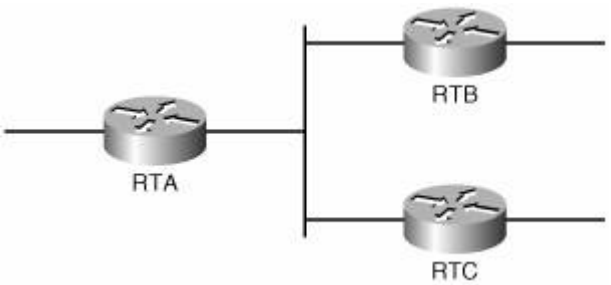
2001:DB8:0:3::/64
2001:DB8:0:2::/64

PREV NEXT

# Troubleshooting Exercises

1. Example 6-32 through Example 6-34 show the configurations for the three routers in Figure 6-23. Which subnets are in the routing tables of each router? From each router, which subnets are reachable and which subnets (if any) are unreachable?

Figure 6-23. The network for Troubleshooting Exercises 1 and 2.




Example 6-32. The configuration of RTA in Figure 6-23.

```
RTA#show running-config
Building configuration...
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RTA
!
!
!
interface Ethernet0
  ip address 192.168.13.86 255.255.255.248
!
interface Serial0
  no ip address
  shutdown
!
interface Serial1
  no ip address
  shutdown
!
interface TokenRing0
  ip address 192.168.13.34 255.255.255.224
  ring-speed 16
!
router rip
  version 2
```

```
  network 192.168.13.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
  login
!
end
RTA#
```

Example 6-33. The configuration of RTB in Figure 6-23.

```
RTB#show running-config
Building configuration...
Current configuration:
!
version 11.2
no service udp-small-servers
no service tcp-small-servers
!
hostname RTB
!
!
!
interface Ethernet0
  ip address 192.168.13.90 255.255.255.240
!
interface Serial0
  no ip address
  shutdown
!
interface Serial1
  no ip address
  shutdown
!
interface TokenRing0
  ip address 192.168.13.35 255.255.255.224
  ring-speed 16
!
router rip
  version 2
  network 192.168.13.0
!
no ip classless
!
!
line con 0
line aux 0
line vty 0 4
  login
!
end
RTB#
```

Example 6-34. The configuration of RTC in Figure 6-23.

```
RTC#show running-config
Building configuration...
Current configuration:
!
version 11.1
service udp-small-servers
service tcp-small-servers
!
hostname RTC
!
!
!
interface Ethernet0
  ip address 192.168.13.75 255.255.255.224
!
interface Serial0
  no ip address
  shutdown
!
interface Serial1
  no ip address
  shutdown
!
interface TokenRing0
  ip address 192.168.13.33 255.255.255.224
  ring-speed 16
!
router rip
  network 192.168.13.0
!
no ip classless
!
line con 0
line 1 8
line aux 0
line vty 0 4
  login
!
end
RTC#
```

2 The configurations of RTA and RTB in Figure 6-23 are changed as follows:[14]

[14] The configuration shown is for RTB. RTA is identical, except for the IP address, which is 192.168.13.34.

```
interface Ethernet0
ip address 192.168.13.35 255.255.255.224
ip rip receive version 1 2
```

Will any subnets be added to any routing tables as a result of this change? Explain why or why not.

# Chapter 7. Enhanced Interior Gateway Routing Protocol (EIGRP)

This chapter covers the following subjects:


- The Roots of EIGRP: An Overview of IGRP
- From IGRP to EIGRP
- Operation of EIGRP
- Configuring EIGRP
- Troubleshooting EIGRP


First released in IOS 9.21, Enhanced Interior Gateway Routing protocol (EIGRP) is, as the name says, an enhancement of the Cisco Interior Gateway Routing Protocol (IGRP). The name is apt because unlike RIPv2, EIGRP is far more than the same protocol with some added extensions. Like IGRP, EIGRP is a distance vector protocol and uses the same composite metrics as IGRP uses. Beyond that, there are few similarities.

IGRP was discontinued as of IOS releases 12.2(13)T and 12.2(Rls4)S. Although innovative in its time, modern network operators who desire more capability than RIP provides move to EIGRP or OSPF, not the moderately more capable IGRP. However, to understand EIGRP, it is useful to first examine the protocol from which it evolved.

# The Roots of EIGRP: An Overview of IGRP

Cisco developed IGRP in the mid-1980s as an answer to the limitations of RIP, the most significant of which are the hop count metric and the 15-hop network size. IGRP calculated a composite metric from a variety of route variables and provided "knobs" for weighting the variables to reflect the specific characteristics and needs of the network. Although hop count is not one of these variables, IGRP did track hop count, and could be implemented on networks of up to 255 hops in diameter.

The other advantages that IGRP presented over RIP are

- Unequal-cost load sharing
- An update period three times longer than RIP's
- A more efficient update packet format

The chief disadvantage of both IGRP and EIGRP is that they are proprietary to Cisco and therefore limited to Cisco products, whereas RIP is a part of any IP routing process on any platform.

The Cisco objective when developing IGRP was to create a versatile, robust protocol capable of being adapted to a variety of routed protocol suites. In addition to routing IP, IGRP could also route the ISO Connectionless Network Protocol (CLNP). This multiprotocol capability was also adapted in EIGRP, which can route not only IP but also IPX and AppleTalk.

From a high-altitude view, IGRP shares many operational characteristics with RIP. It is a classful distance vector protocol that periodically broadcasts its entire routing tablewith the exception of routes suppressed by split horizonto all its neighbors. Like RIP, IGRP broadcasts a request packet out all IGRP-enabled interfaces upon startup and performs a sanity check on received updates to verify that the source address of the packet belongs to the same subnet on which the update was received.[1] New update entries with reachable metrics are placed in the routing table, and an entry replaces an older entry to the same destination only if the metric is smaller. Split horizon with poisoned reverse, triggered updates, and holddown timers are used for stability; IGRP summarizes addresses at network boundaries.

[1] This sanity check can be disabled with the command **no validate-update-source**.

Unlike RIP, which is accessed via UDP, the IGRP process is accessed directly from the IP layer as protocol 9.

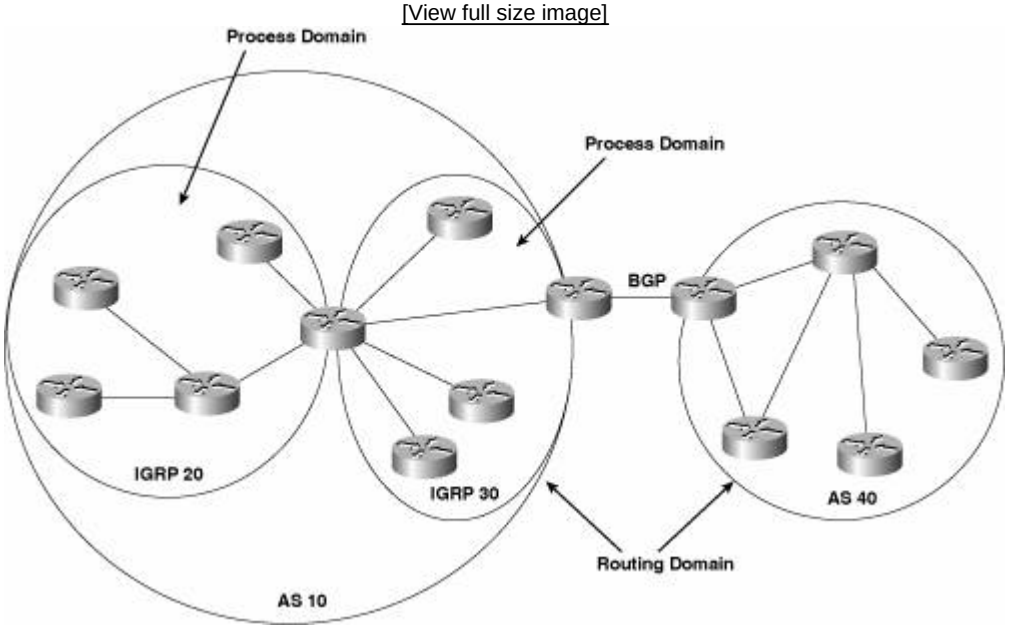
## Process Domains

IGRP also uses the concept of process domains. By defining and tracking multiple process domains, you can isolate the communications within one domain from the communications within another domain. Traffic between the domains can then be closely regulated by redistribution (Chapter 11, "Route Redistribution") and route filtering (Chapter 13, "Route Filtering").

Figure 7-1 illustrates the contrast between process domains and routing domains. Here two autonomous systems (AS) are defined: AS 10 and AS 40. These systems are routing domainsa set of routers running one or more IGPs under a common administration. They communicate via an Exterior Gateway Protocol (in this case, Border Gateway Protocol, or BGP).

Figure 7-1. An autonomous system number may specify a routing domain, which is a group of routers running one or more IGP processes under a single administrative domain. Under IGRP, an autonomous system number may also specify a process domain, which is a group of routers sharing routing information by means of a single routing process.

[View full size image]




Within AS 10 are two IGRP process domains: IGRP 20 and IGRP 30. Under IGRP, the 20 and 30 are defined as autonomous system numbers. In this context, the numbers serve to distinguish two routing processes within the same routing domain. IGRP 20 and IGRP 30 communicate via the single router connected to both domains. This router runs both IGRP processes and automatically redistributes between them.

Within its updates, IGRP classifies route entries into one of three categories: interior routes, system routes, and exterior routes.

An *interior* route is a path to a subnet of the network address of the data link on which the update is being broadcast. In other words, a subnet advertised as an interior route is "local" to the major network to which the advertising router and the receiving router are commonly connected.

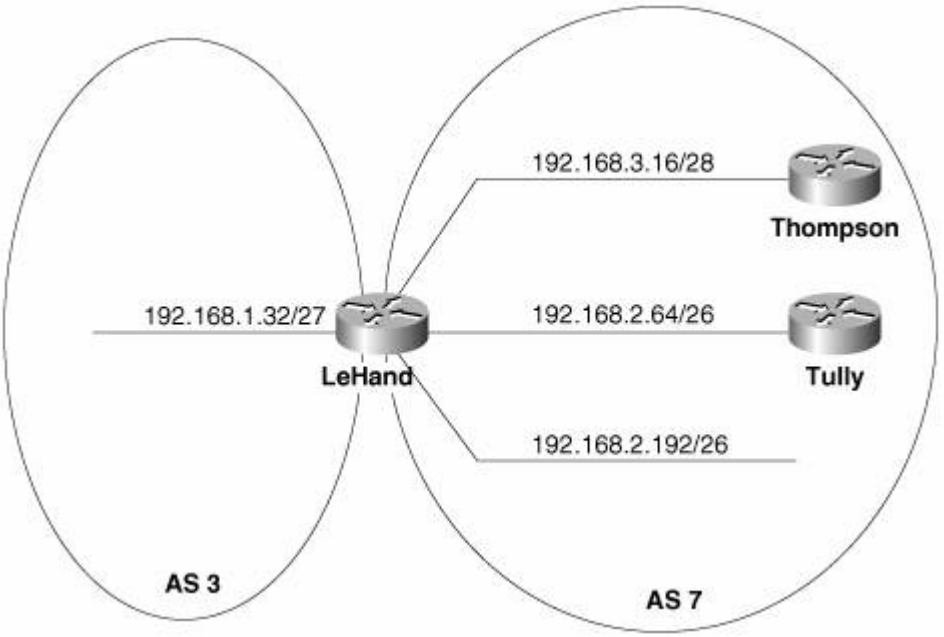
A *system* route is a path to a network address, which has been summarized by a network boundary router.

An *exterior* route is a path to a network that has been flagged as a *default network*. A default network is an address to which a router will send any packet that cannot be matched to a more specific destination.[2] Default networks and their configuration are covered in Chapter 12, "Default Routes and On-Demand Routing."

[2] Classifying a default network as an external route is unique to IGRP and EIGRP. Open protocols such as RIP and OSPF advertise default networks with the address 0.0.0.0.

Figure 7-2 shows how IGRP uses these three categories. The routers LeHand and Tully are connected to subnet 192.168.2.64/26, so major network 192.168.2.0 is considered the "local" network shared by those two routers. LeHand is also attached to 192.168.2.192/26, which is another subnet of the network connecting the two routers. Therefore, LeHand advertises that subnet to Tully as an internal route.

Figure 7-2. LeHand advertises subnet 192.168.2.192/26 to Tully as an internal route. Network 192.168.3.0 is advertised to Tully as a system route, and 192.168.1.0 is advertised as an external route.




However, the local network for LeHand and Thompson is 192.168.3.0. LeHand is the boundary router between major networks 192.168.2.0 and 192.168.3.0, so 192.168.2.0 will be advertised to Thompson as a system route. Likewise, 192.168.3.0 is advertised to Tully as a system route.

192.168.1.0 is a network in another autonomous system, and LeHand has been configured to advertise that network address as a default route. 192.168.1.0 will therefore be advertised to both Thompson and Tully as an external route.

## IGRP Timers and Stability Features

The IGRP update period is 90 seconds. A random jitter variable of up to 20 percent is subtracted from each update time to prevent update timer synchronization, so the time elapsed between individual updates will vary from 72 to 90 seconds.

When a route is first learned, the invalid timer for that route is set for 270 seconds, or three times the update period. The flush timer is set for 630 secondsseven times the update period. Each time an update is received for the route, these timers are reinitialized. If the invalid timer expires before an update is heard, the route is marked as unreachable. It will be held in the routing table and advertised as unreachable until the flush timer expires, at which time the route will be deleted from the table.

The 90-second timer used by IGRP, in comparison to the 30-second timer used by RIP, means that compared to RIP, IGRP uses less bandwidth for periodic updates. However, the trade-off is that in some cases IGRP might be slower to converge than RIP. For example, if a router goes offline, IGRP takes three times as long as RIP to detect the dead neighbor.

If a destination becomes unreachable or if the next-hop router increases the metric of a destination enough to cause a triggered update, the route will be placed in holddown for 280 seconds (three update periods plus 10 seconds). Until the holddown timer expires, no new information will be accepted about this destination. IGRP holddown may be disabled with the

command `no metric holddown`. In loop-free topologies, where holddown has no real benefit, disabling the function can reduce reconvergence time.

The default timers can be changed with the following command:

**timers basic** *update invalid holddown flush [sleeptime]*

This command is also used to manipulate RIP timers with the exception of the *sleeptime* option. Sleeptime is a timer used to specify a period, in milliseconds, to delay a regular routing update after receiving a triggered update.

## IGRP Metrics

One of the most significant changes from RIP that IGRP introduces, and which carries over into EIGRP, is the use of multiple metric parameters, based on link characteristics. It is useful to study how IGRP handles these metrics to understand how EIGRP handles its metrics.

The link characteristics from which IGRP calculates its composite metric are bandwidth, delay, load, and reliability. By default, IGRP chooses a route based on bandwidth and delay. If a data link is thought of as a pipe, then bandwidth is the width of the pipe and delay is the length of the pipe. In other words, bandwidth is a measure of the carrying capacity, and delay is a measure of the end-to-end travel time. Load and reliability are taken into consideration only if the router is configured to do so. IGRP also tracks the smallest Maximum Transmission Unit (MTU) along each route, although the MTU is not used in the composite metric calculation. The quantities associated with IGRP's composite metric on a specific interface can be observed with the `show interfaces` command (Example 7-1).

### Example 7-1. The output of every show interface command includes metric statistics for the interface. This Ethernet interface shows MTU = 1500 bytes, bandwidth = 10 megabits per second, delay = 1000 microseconds, reliability = 100 percent, and load = 39 percent (minimum load).

```
Newfoundland#show interfaces ethernet 0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0000.0c76.5b7c (bia 0000.0c76.5b7c)
  Internet address is 10.2.1.2/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:07, output 00:00:01, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     601753 packets input, 113607697 bytes, 0 no buffer
     Received 601753 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     693494 packets output, 557731861 bytes, 0 underruns
     0 output errors, 5 collisions, 13 interface resets
     0 babbles, 0 late collision, 48 deferred
     0 lost carrier, 0 no carrier
```

```
     0 output buffer failures, 0 output buffers swapped out
Newfoundland#
```

Bandwidth is expressed in units of kilobits per second. It is a static number used for metric calculation only and does not necessarily reflect the actual bandwidth of the linkthat is, bandwidth is not measured dynamically. For example, the default bandwidth of a serial interface is 1544, whether the interface is attached to a T1 or a 56K line. This bandwidth number may be changed from the default with the bandwidth command.

IGRP updates use a three-octet number, referred to in this book as $BW_{IGRP}$, which is the inverse of the bandwidth scaled by a factor of $10^7$. So if the bandwidth of an interface is 1544, then

$$BW_{IGRP} = 10^7/1544 = 6476, \text{ or } 0x00194C.$$

Delay, like bandwidth, is a static figure and is not measured dynamically. It is displayed by the show interface command as DLY, in units of microseconds. The default delay of an interface may be changed with the delay command, which specifies the delay in tens of microseconds. Example 7-2 shows the bandwidth and delay commands used to change the defaults of the interface of Example 7-1.

## Example 7-2. The bandwidth and delay commands are used to change the metric defaults of the e0 interface. The new quantities can be seen in the output of the *show interfaces* command.

```
Newfoundland(config)#interface e0
Newfoundland(config-if)#bandwidth 75000
Newfoundland(config-if)#delay 5
Newfoundland(config-if)#^Z
Newfoundland#
%SYS-5-CONFIG_I: Configured from console by console
Newfoundland#show interfaces ethernet0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0000.0c76.5b7c (bia 0000.0c76.5b7c)
  Internet address is 10.2.1.2/24
  MTU 1500 bytes, BW 75000 Kbit, DLY 50 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:02, output 00:00:02, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     601888 packets input, 113637882 bytes, 0 no buffer
     Received 601888 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     693646 packets output, 557884632 bytes, 0 underruns
     0 output errors, 5 collisions, 13 interface resets
     0 babbles, 0 late collision, 48 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
Newfoundland#
```

When carried in an IGRP update, delay is a three-octet number expressed in the same 10-

microsecond units as specified by the delay command. To avoid confusion, this number will be referred to as $DLY_{IGRP}$, to differentiate it from DLY, in microseconds, observed with show interface. For example, if DLY is 50, then

$DLY_{IGRP}$ = DLY/10 = 50/10 = 5, or 0x000005.

IGRP also uses delay to indicate an unreachable route by setting $DLY_{IGRP}$ = 0xFFFFFF. This number translates to approximately 167.8 seconds, so the maximum end-to-end delay of an IGRP route is 167 seconds.

Because IGRP uses bandwidth and delay as its default metrics, these quantities must be configured correctly and consistently on all interfaces of all IGRP routers.

Changing the bandwidth or delay of an interface should be done only for good reasons and only with a full understanding of the results of those changes. In most cases, it is best to leave the default values unchanged. A notable exception is serial interfaces. As noted earlier in this section, serial interfaces on Cisco routers have a default bandwidth of 1544 no matter what the bandwidth is of the connected link. The bandwidth command should be used to set the interface to the actual bandwidth of the serial link.

It is important to note that OSPF also uses the bandwidth statement to calculate its metric. Therefore, if IGRP (or EIGRP) metrics are to be manipulated in a network where OSPF is also running, use the delay to influence IGRP. Changing the bandwidth will affect both IGRP and OSPF.

Table 7-1 lists the bandwidths and delays for a few common interfaces. (The default bandwidth of a serial interface is always 1544; Table 7-1 shows the figures that would result from using the bandwidth command to reflect the actual connected bandwidth.)

### Table 7-1. Common $BW_{IGRP}$ and $DLY_{IGRP}$ quantities.

| Media | Bandwidth | $BW_{IGRP}$ | Delay | $DLY_{IGRP}$ |
|---|---|---|---|---|
| 100M ATM | 100000K | 100 | 100µS | 10 |
| Fast Ethernet | 100000K | 100 | 100µS | 10 |
| FDDI | 100000K | 100 | 100µS | 10 |
| HSSI | 45045K | 222 | 20000µS | 2000 |
| 16M Token Ring | 16000K | 625 | 630µS | 63 |
| Ethernet | 10000K | 1000 | 1000µS | 100 |
| T1 | 1544K | 6476 | 20000µS | 2000 |
| DS0 | 64K | 156250 | 20000µS | 2000 |
| 56K | 56K | 178571 | 20000µS | 2000 |
| Tunnel | 9K | 1111111 | 500000µS | 50000 |

Reliability is measured dynamically and is expressed as an eight-bit number, where 255 is a 100 percent reliable link and 1 is a minimally reliable link. In the output of show interface, reliability is shown as a fraction of 255, for example, 234/255 (see Example 7-3).

## Example 7-3. This interface shows a reliability of 234/255, or 91.8 percent.

```
Casablanca#show interface ethernet0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0000.0c76.5b7c (bia 0000.0c76.5b7c)
  Internet address is 172.20.1.1 255.255.255.0
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 234/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 4:00:00
  Last input 0:00:28, output 0:00:06, output hang never
  Last clearing of "show interface" counters 0:06:05
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     22 packets input, 3758 bytes, 0 no buffer
     Received 21 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     125 packets output, 11254 bytes, 0 underruns
     39 output errors, 694 collisions, 0 interface resets, 0 restarts
     0 output buffer failures, 0 output buffers swapped out
Casablanca#
```

Load, in an IGRP update, is an eight-bit number. Load is represented in the output of show interface as a fraction of 255, such as 40/255 (Example 7-4); 1 is a minimally loaded link, and 255 is a 100 percent loaded link.

## Example 7-4. This interface shows a load of 40/255, or 15.7 percent.

```
Yalta#show interface serial 1
Serial1 is up, line protocol is up
  Hardware is HD64570
  Internet address is 172.20.20.2 255.255.255.0
  MTU 1500 bytes, BW 56 Kbit, DLY 20000 usec, rely 255/255, load 40/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input 0:00:08, output 0:00:00, output hang never
  Last clearing of "show interface" counters 0:05:05
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 10000 bits/sec, 1 packets/sec
  5 minute output rate 9000 bits/sec, 1 packets/sec
     456 packets input, 397463 bytes, 0 no buffer
     Received 70 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     428 packets output, 395862 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets, 0 restarts
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=up DSR=up DTR=up RTS=up CTS=up
```

If reliability or load is to be used as a metric or as part of a composite metric, the algorithm for calculating the metric must not allow sudden changes in the error rate or channel occupancy to destabilize the network. As an example, if a "raw," or instantaneous, measure of load is used, a burst of heavy traffic could cause a route to go into holddown and an abrupt drop in traffic could trigger an update. To prevent frequent metric changes, reliability and load are calculated based on an exponentially weighted average with a five-minute time constant, which is updated every

five seconds.

The composite metric for each IGRP route is calculated as

$$\text{metric} = [k1*BW_{IGRP(min)} + (k2*BW_{IGRP(min)})/(256-LOAD) + k3*DLY_{IGRP(sum)}] \times [k5/(RELIABILITY + k4)],$$

where $BW_{IGRP(min)}$ is the minimum $BW_{IGRP}$ of all the outgoing interfaces along the route to the destination and $DLY_{IGRP(sum)}$ is the total $DLY_{IGRP}$ of the route.

The values k1 through k5 are configurable weights; their default values are k1=k3=1 and k2=k4=k5=0. These defaults can be changed with the command:[3]

[3] *tos* is a relic of the Cisco original intention to have IGRP do type of service routing; this plan was never adopted, and *tos* in this command is always set to zero.

**metric weights** tos k1 k2 k3 k4 k5[3]

If k5 is set to zero, the [k5/(RELIABILITY+k4)] term is not used.

Given the default values for k1 through k5, the composite metric calculation used by IGRP reduces to the default metric:

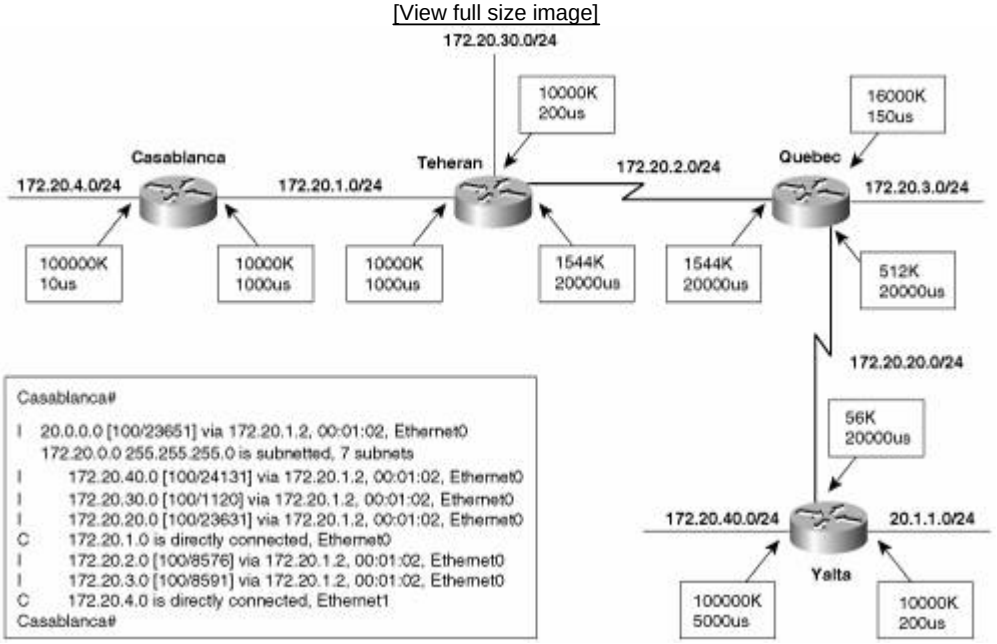
$$\text{metric} = BW_{IGRP(min)} + DLY_{IGRP(sum)}$$

The network example in Figure 7-3 shows the bandwidths and delays configured on each interface and a forwarding database from one of the routers with the derived IGRP metrics.[4]

[4] Also notice the administrative distance, which is 100 for IGRP.

Figure 7-3. By default, the total delay is added to the minimum bandwidth to derive the IGRP metric.

[View full size image]




The routing table itself shows only the derived metric, but the actual variables recorded by IGRP for each route can be seen by using the command show ip route *address*, as in Example 7-5.

Here the minimum bandwidth on the route from Casablanca to subnet 172.20.40.0/24 is 512K, at Quebec. The total delay of the route is (1000 + 20000 + 20000 + 5000) = 46000 microseconds:

$$BW_{IGRP(min)} = 10^7/512 = 19531$$

$$DLY_{IGRP(sum)} = 46000/10 = 4600$$

$$metric = BW_{IGRP(min)} + DLY_{IGRP(sum)} = 19531 + 4600 = 24131$$

### Example 7-5. The metric for the route from Casablanca to subnet 172.20.40.0 is calculated from the minimum bandwidth of 512K and the total delay of 46000 microseconds.

```
Casablanca#show ip route 172.20.40.0
Routing entry for 172.20.40.0 255.255.255.0
  Known via "igrp 1", distance 100, metric 24131
  Redistributing via igrp 1
  Advertised by igrp 1 (self originated)
  Last update from 172.20.1.2 on Ethernet0, 00:00:54 ago
  Routing Descriptor Blocks:
  * 172.20.1.2, from 172.20.1.2, 00:00:54 ago, via Ethernet0
      Route metric is 24131, traffic share count is 1
      Total delay is 46000 microseconds, minimum bandwidth is 512 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 1/255, Hops 2
```

Example 7-5 shows that IGRP also records the smallest MTU along the route in addition to the hop count. MTU is not used in the metric calculation. Hop count is the hop count reported by the next-hop router and is used only to limit the diameter of the network. By default, the maximum hop count is 100 and can be configured from 1 to 255 with the command `metric maximum-hops`. If the maximum hop count is exceeded, the route will be marked unreachable by setting the delay to 0xFFFFFF.

Note that all metrics are calculated from the outgoing interfaces along the route. For example, the metric for the route from Yalta to subnet 172.20.4.0/24 is different from the metric for the

PREV NEXT

# From IGRP to EIGRP

The original motivation for developing EIGRP was simply to make IGRP classless. But early in the development the engineers working on the project recalled some academic proposals for a new kind of convergence algorithm and decided to use that algorithm in their extension of IGRP. The result was a protocol that, while retaining some concepts introduced with IGRP such as multiple metrics, protocol domains, and unequal-cost load balancing, is distinctly different from IGRP.

EIGRP is occasionally described as a distance vector protocol that acts like a link-state protocol. To recap the extensive discussion in Chapter 4, "Dynamic Routing Protocols," a distance vector protocol shares everything it knows, but only with directly connected neighbors. Link-state protocols announce information only about their directly connected links, but they share the information with all routers in their routing domain or area.

All the distance vector protocols discussed so far run some variant of the Bellman-Ford (or Ford-Fulkerson) algorithm. These protocols are prone to routing loops and counting to infinity. As a result, they must implement loop-avoidance measures such as split horizon, route poisoning, and hold-down timers. Because each router must run the routing algorithm on received routes before passing those routes along to its neighbors, larger networks might be slow to converge. More important, distance vector protocols advertise routes; the change of a critical link might mean the advertisement of many changed routes.

Compared to distance vector protocols, link-state protocols are far less susceptible to routing loops and bad routing information. The forwarding of link-state packets is not dependent on performing the route calculations first, so large networks might converge faster. And only links or prefixes and their states are advertised, not routes, which means the change of a link will not cause the advertisement of all routes using that link.

Regardless of whether other routing protocols perform route calculations before sending distance vector updates to neighbors or after building a topological database, their common denominator is that they perform the calculations individually. In contrast to the Bellman-Ford algorithms used by most other distance vector protocols, EIGRP uses a system of *diffusing computations*route calculations that are performed in a coordinated fashion among multiple routersto attain fast convergence while remaining loop-free at every instant.

Although EIGRP updates are still vectors of distances transmitted to directly connected neighbors, they are nonperiodic, partial, and bounded. *Nonperiodic* means that updates are not sent at regular intervals; rather, updates are sent only when a metric or topology change occurs. *Partial* means that the updates will include only routes that have changed, not every entry in the route table. *Bounded* means that the updates are sent only to affected routers. These characteristics mean that EIGRP uses much less bandwidth than typical distance vector protocols use. This feature can be especially important on low-bandwidth, high-cost Wide Area Network (WAN) links.

Another concern when routing over low-bandwidth WAN links is the maximum amount of bandwidth used during periods of convergence, when routing traffic is high. By default, EIGRP uses no more than 50 percent of the bandwidth of a link. Later IOS releases allow this percentage to be changed with the command `ip bandwidth-percent eigrp`.

EIGRP is a classless protocol (that is, each route entry in an update includes a subnet mask). Variable-length subnet masks may be used with EIGRP not only for sub-subnetting as described in Chapter 6, "RIPv2, RIPng, and Classless Routing," but also for address aggregationthe summarization of a group of major network addresses.

EIGRP packets can be authenticated using an MD5 cryptographic checksum. The basics of authentication and MD5 are covered in Chapter 6; an example of configuring EIGRP

authentication is included in this chapter.

Finally, a major feature of EIGRP is that it can route not only IP but also IPX and AppleTalk.

PREV NEXT

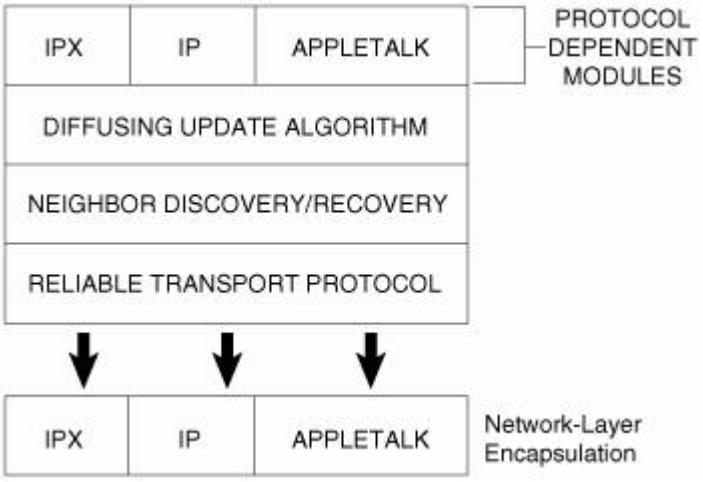
# Operation of EIGRP

EIGRP uses the same formula that IGRP uses to calculate its composite metric. However, EIGRP scales the metric components by 256 to achieve a finer metric granularity. So if the minimum configured bandwidth on the path to a destination is 512K and the total configured delay is 46000 microseconds, IGRP would calculate a composite metric of 24131. EIGRP, however, will multiply the bandwidth and delay components by 256 for a metric of 256 x 24131 = 6177536.

EIGRP has four components (Figure 7-4):

- Protocol-Dependent Modules
- Reliable Transport Protocol (RTP)
- Neighbor Discovery/Recovery
- Diffusing Update Algorithm (DUAL)

Figure 7-4. The four major components of EIGRP. RTP and neighbor discovery are lower-level protocols that enable the correct operation of DUAL. DUAL can perform route computations for multiple routed protocols.




This section examines each EIGRP component, with particular emphasis on DUAL, and ends with a discussion of address aggregation.

## Protocol-Dependent Modules

EIGRP implements modules for IP, IPX, and AppleTalk, which are responsible for the protocol-specific routing tasks. For example, the IPX EIGRP module is responsible for exchanging route information about IPX networks with other IPX EIGRP processes and for passing the information to the DUAL. Additionally, the IPX module will send and receive SAP information.

As Figure 7-4 shows, the traffic for the individual modules is encapsulated within their respective network layer protocols. EIGRP for IPX, for example, is carried in IPX packets.

EIGRP automatically redistributes with other protocols in many cases:

- IPX EIGRP automatically redistributes with IPX RIP and NLSP.
- AppleTalk EIGRP automatically redistributes with AppleTalk RTMP.
- IP EIGRP automatically redistributes routes with IGRP if the IGRP process is in the same autonomous system.

Redistribution with other IP routing protocols is the subject of Chapter 11.

Configuration of EIGRP for IPX and AppleTalk is outside the scope of this book. Refer to the *Cisco Configuration Guide* for more information.

## Reliable Transport Protocol

The Reliable Transport Protocol (RTP) manages the delivery and reception of EIGRP packets. *Reliable delivery* means that delivery is guaranteed and that packets will be delivered in order.

Guaranteed delivery is accomplished by means of a Cisco-proprietary algorithm known as *reliable multicast*, using the reserved class D address 224.0.0.10. Each neighbor receiving a reliably multicast packet unicasts an acknowledgment.

Ordered delivery is ensured by including two sequence numbers in the packet. Each packet includes a sequence number assigned by the sending router. This sequence number is incremented by one each time the router sends a new packet. In addition, the sending router places in the packet the sequence number of the last packet received from the destination router.

In some cases, RTP may use *unreliable delivery*. No acknowledgment is required, and no sequence number will be included for unreliably delivered EIGRP packets.

EIGRP uses multiple packet types, all of which are identified by protocol number 88 in the IP header:

- Hellos are used by the neighbor discovery and recovery process. Hello packets are multicast and use unreliable delivery.
- Acknowledgments (ACKs) are Hello packets with no data in them. ACKs are always unicast and use unreliable delivery.
- Updates convey route information. Unlike RIP and IGRP updates, these packets are transmitted only when necessary, contain only necessary information, and are sent only to routers that require the information. When updates are required by a specific router, they are unicast. When updates are required by multiple routers, such as upon a metric or topology change, they are multicast. Updates always use reliable delivery.
- Queries and Replies are used by the DUAL finite state machine to manage its diffusing computations. Queries can be multicast or unicast, and replies are always unicast. Both queries and replies use reliable delivery.
- Requests were a type of packet originally intended for use in route servers. This application was never implemented, and request packets are noted here only because they are mentioned in some older EIGRP documentation.

If any packet is reliably multicast and an ACK is not received from a neighbor, the packet will be

retransmitted as a unicast to that unresponding neighbor. If an ACK is not received after 16 of these unicast retransmissions, the neighbor will be declared dead.

The time to wait for an ACK before switching from multicast to unicast is specified by the *multicast flow timer*. The time between the subsequent unicasts is specified by the *retransmission timeout* (RTO). Both the multicast flow timer and the RTO are calculated for each neighbor from the *smooth round-trip time* (SRTT). The SRTT is the average elapsed time, measured in milliseconds, between the transmission of a packet to the neighbor and the receipt of an acknowledgment. The formulas for calculating the exact values of the SRTT, the RTO, and the multicast flow timer are proprietary.

The following two subsections discuss the EIGRP components that use the various packet types.

## Neighbor Discovery/Recovery

Because EIGRP updates are nonperiodic, it is especially important to have a process whereby neighborsEIGRP-speaking routers on directly connected networksare discovered and tracked. On most networks, Hellos are multicast every five seconds, minus a small random time to prevent synchronization. On multipoint X.25, Frame Relay, and ATM interfaces, with access link speeds of T1 or slower, Hellos are unicast every 60 seconds.[5] This longer Hello interval is also the default for ATM SVCs and for ISDN PRI interfaces. In all cases, the Hellos are unacknowledged. The default Hello interval can be changed on a per interface basis with the command `ip hello-interval eigrp`.

[5] Point-to-point subinterfaces send Hellos every 5 seconds.

When a router receives a Hello packet from a neighbor, the packet will include a *hold time*. The hold time tells the router the maximum time it should wait to receive subsequent Hellos. If the hold timer expires before a Hello is received, the neighbor is declared unreachable and DUAL is informed of the loss of a neighbor. By default, the hold time is three times the Hello interval180 seconds for low-speed nonbroadcast multiaccess (NBMA) networks and 15 seconds for all other networks. The default can be changed on a per interface basis with the command `ip hold-time eigrp`. The capability to detect a lost neighbor within 15 seconds, as opposed to 180 seconds for RIP and 270 seconds for IGRP, is one factor contributing to EIGRP's fast reconvergence.

Information about each neighbor is recorded in a neighbor table. As Example 7-6 shows, the *neighbor table* records the IP address of the neighbor and the interface on which the neighbor's Hellos are received. The hold time advertised by the neighbor is recorded, as is the SRTT and the *uptime*the time since the neighbor was added to the table. The RTO is the time, in milliseconds, that the router will wait for an acknowledgment of a unicast packet sent after a multicast has failed. If an EIGRP update, query, or reply is sent, a copy of the packet will be queued. If the RTO expires before an ACK is received, another copy of the queued packet is sent. The Q Count indicates the number of enqueued packets. The sequence number of the last update, query, or reply packet received from the neighbor is also recorded in the neighbor table. The RTP tracks these sequence numbers to ensure that packets from the neighbor are not received out of order. Finally, the H column records the order in which the neighbors were learned.

### Example 7-6. The command *show ip eigrp neighbors* is used to observe the IP EIGRP neighbor table.

```
Wright#show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address   Interface   Hold  Uptime   SRTT   RTO   Q   Seq
                          (sec)          (ms)        Cnt  Num
3   10.1.1.2  Et0           10  09:01:27   12   200   0   5
2   10.1.4.2  Se1           13  09:02:11   23   200   0   11
1   10.1.2.2  Et1           14  09:02:12    8   200   0   15
0   10.1.3.2  Se0           12  09:02:12   21   200   0   13
```

# Diffusing Update Algorithm

DUAL is a convergence algorithm that replaces the Bellman-Ford or Ford-Fulkerson algorithms used by other distance vector protocols. The design philosophy behind DUAL is that even temporary routing loops are detrimental to the performance of a network. DUAL uses diffusing computations, first proposed by E. W. Dijkstra and C. S. Scholten,[6] to perform distributed shortest-path routing while maintaining freedom from loops at every instant. Although many researchers have contributed to the development of DUAL, the most prominent work is that of J. J. Garcia-Luna-Aceves.[7]

[6] Edsger W. Dijkstra and C. S. Scholten. "Termination Detection for Diffusing Computations." Information Processing Letters, Vol. 11, No. 1, pp. 14: 29 August 1980.

[7] J. J. Garcia-Luna-Aceves. "A Unified Approach for Loop-Free Routing Using Link States or Distance Vectors," ACM SIGCOMM Computer Communications Review, Vol. 19, No. 4, pp. 212223: September 1989.

J. J. Garcia-Luna-Aceves. "Loop-Free Routing Using Diffusing Computations," IEEE/ACM Transactions on Networking, Vol. 1, No. 1, February 1993.

## DUAL: Preliminary Concepts

For DUAL to operate correctly, a lower-level protocol must ensure that the following conditions are met:[8]

[8] J. J. Garcia-Luna-Aceves. "Area-Based, Loop-Free Internet Routing." Proceedings of IEEE INFOCOMM 94. Toronto, Ontario, Canada, June 1994.

- A node detects within a finite time the existence of a few neighbor or the loss of connectivity with a neighbor.
- All messages transmitted over an operational link are received correctly and in the proper sequence within a finite time.
- All messages, changes in the cost of a link, link failures, and new-neighbor notifications are processed one at a time within a finite time and in the order in which they are detected.

The Cisco EIGRP uses Neighbor Discovery/Recovery and RTP to establish these preconditions.

Before the operation of DUAL can be examined, a few terms and concepts must be described.

Upon startup, a router uses Hellos to discover neighbors and to identify itself to neighbors. When a neighbor is discovered, EIGRP will attempt to form an adjacency with that neighbor. An *adjacency* is a logical association between two neighbors over which route information is exchanged. When adjacencies have been established, the router will receive updates from its neighbors. The updates will contain all routes known by the sending routers and the metrics of those routes. For each route, the router will calculate a distance based on the distance advertised by the neighbor and the cost of the link to that neighbor.

The lowest calculated metric to each destination will become the *feasible distance* (FD) of that destination. For example, a router may be informed of three different routes to subnet 172.16.5.0 and may calculate metrics of 380672, 12381440, and 660868 for the three routes. 380672 will become the FD because it is the lowest calculated distance.

The *feasibility condition* (FC) is a condition that is met if a neighbor's advertised distance to a destination is lower than the router's FD to that same destination.

If a neighbor's advertised distance to a destination meets the FC, the neighbor becomes a

*feasible successor*[9] for that destination. For example, if the FD to subnet 172.16.5.0 is 380672 and a neighbor advertises a route to that subnet with a distance of 355072, the neighbor will become a feasible successor; if the neighbor advertises a distance of 380928, it will not satisfy the FC and will not become a feasible successor.

[9] Successor simply means a router that is one hop closer to a destinationin other words, a next-hop router.

The concepts of feasible successors and the FC are central to loop avoidance. Because feasible successors are always "downstream," (that is, a shorter metric distance to the destination than the FD) a router will never choose a path that will lead back through itself, creating a loop. Such a path would have a distance larger than the FD.

Every destination for which one or more feasible successors exist will be recorded in a *topological table*, along with the following items:

- The destination's FD
- All feasible successors
- Each feasible successor's advertised distance to the destination
- The locally calculated distance to the destination via each feasible successor, based on the feasible successor's advertised distance and the cost of the link to that successor
- The interface connected to the network on which each feasible successor is found[10]

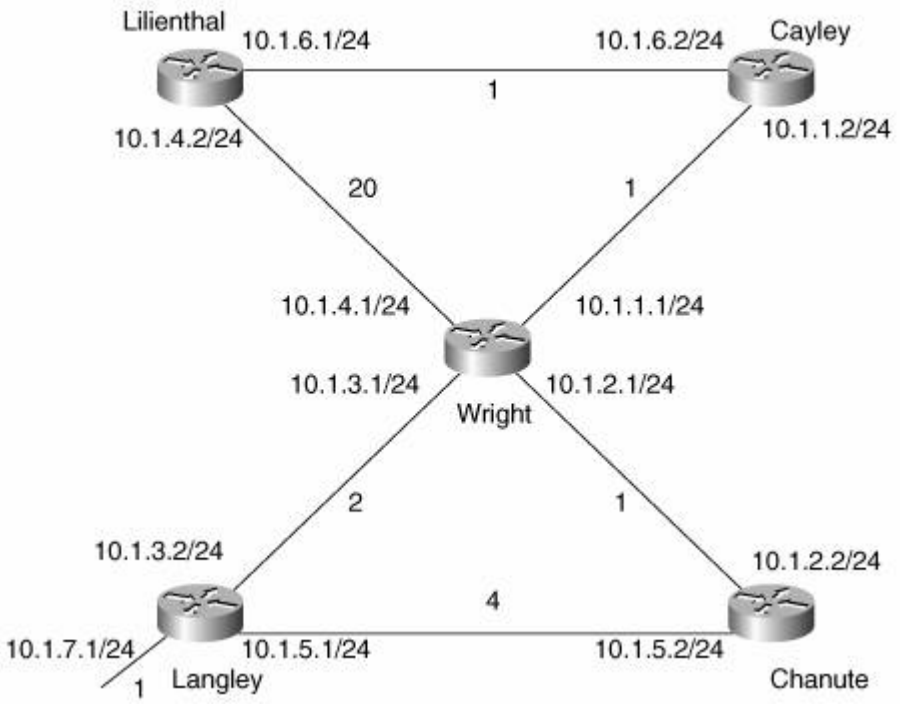
[10] Actually, the interface is not explicitly recorded in the route table. Rather, it is an attribute of the neighbor itself. This convention implies that the same router, seen across multiple parallel links, will be viewed by EIGRP as multiple neighbors.

For every destination listed in the topological table, the route with the lowest metric is chosen and placed into the route table. The neighbor advertising that route becomes the *successor*, or the next-hop router to which packets for that destination are sent.

An example will help clarify these terms, but first a brief discussion of the network used in the examples in this section is necessary. Figure 7-5 shows the EIGRP-based network that is used throughout this and the next three subsections.[11] The command `metric weights 0 0 0 1 0 0` has been added to the EIGRP process so that only delay is used in the metric calculations. The `delay` command has been used with the numbers shown at each link; for example, the interfaces of routers Wright and Langley, connected to subnet 10.1.3.0, have been configured with a delay of 2. These steps have been taken to simplify the examples that follow.

[11] Several of the illustrations in this and the following section, and in the network example used throughout, are adapted from Dr. Garcia-Luna-Aceves's "Loop-Free Routing Using Diffusing Computations," with his permission.

Figure 7-5. The examples and illustrations of this and the next two subsections are based on this EIGRP network.




It should be pointed out that although the delay parameters used here sacrifice realism for simplicity, the way the metrics are manipulated is realistic. Many parameters are calculated from an interface's `bandwidth` specification; some, such as the `ip bandwidth-percent eigrp`, apply directly to EIGRP. Others, such as OSPF cost, do not. As a result, changes of the configured bandwidth should be avoided except to set serial links to their actual bandwidth. If interface metrics need to be manipulated to influence EIGRP (or IGRP) routing, use `delay`. Many unexpected headaches can be avoided.

In Example 7-7, the command `show ip eigrp topology` is used to observe the topology table of router Langley. Each of the seven subnets shown in Figure 7-5 is listed, along with the feasible successors for the subnets. For example, the feasible successors for subnet 10.1.6.0 are 10.1.3.1 (Wright) and 10.1.5.2 (Chanute), via interfaces S0 and S1, respectively.

## Example 7-7. Topology table of router Langley.

```
Langley#show ip eigrp topology
IP-EIGRP Topology Table for process 1

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status

P 10.1.3.0/24, 1 successors, FD is 512
         via Connected, Serial0
P 10.1.2.0/24, 1 successors, FD is 768
         via 10.1.3.1 (768/256), Serial0
         via 10.1.5.2 (1280/256), Serial1
P 10.1.1.0/24, 1 successors, FD is 768
         via 10.1.3.1 (768/256), Serial0
         via 10.1.5.2 (1536/512), Serial1
P 10.1.7.0/24, 1 successors, FD is 256
         via Connected, Ethernet0
P 10.1.6.0/24, 1 successors, FD is 1024
         via 10.1.3.1 (1024/512), Serial0
         via 10.1.5.2 (1792/768), Serial1
P 10.1.5.0/24, 1 successors, FD is 1024
```

```
         via Connected, Serial1
P 10.1.4.0/24, 1 successors, FD is 5632
         via 10.1.3.1 (5632/5120), Serial0
         via 10.1.5.2 (6400/5376), Serial1
Langley#
```

Two metrics in parentheses are also associated with each feasible successor. The first number is the locally calculated metric from Langley to the destination. The second number is the metric advertised by the neighbor. For example, in Figure 7-5 the metric from Langley to subnet 10.1.6.0 via Wright is 256 x (2 + 1 + 1) = 1024, and the metric advertised by Wright for that destination is 256 x (1 + 1) = 512. The two metrics for the same destination via Chanute are 256 x (4 + 1 + 1 + 1) = 1792 and 256 x (1 + 1 + 1) = 768.

The lowest metric from Langley to subnet 10.1.6.0 is 1024, so that is the FD. Example 7-8 shows Langley's route table, with the chosen successors.

## Example 7-8. Langley's route table shows that a single successor has been chosen for each known destination, based on the lowest metric distance.

```
Langley#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
     10.0.0.0/8 is subnetted, 7 subnets
C       10.1.3.0 is directly connected, Serial0
D       10.1.2.0 [90/768] via 10.1.3.1, 00:32:06, Serial0
D       10.1.1.0 [90/768] via 10.1.3.1, 00:32:07, Serial0
C       10.1.7.0 is directly connected, Ethernet0
D       10.1.6.0 [90/1024] via 10.1.3.1, 00:32:07, Serial0
C       10.1.5.0 is directly connected, Serial1
D       10.1.4.0 [90/5632] via 10.1.3.1, 00:32:07, Serial0
Langley#
```

Langley has only one successor for every route. The topology table of Cayley (Example 7-9) shows that there are two successors for 10.1.4.0 because the locally calculated metric for both routes matches the FD. Both routes are entered into the route table (Example 7-10), and Cayley will perform equal-cost load balancing.

## Example 7-9. Topology table of Cayley, showing two successors to subnet 10.1.4.0.

```
Cayley#show ip eigrp topology
IP-EIGRP Topology Table for process 1
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status
P 10.1.3.0/24, 1 successors, FD is 768
         via 10.1.1.1 (768/512), Ethernet0
P 10.1.2.0/24, 1 successors, FD is 512
         via 10.1.1.1 (512/256), Ethernet0
P 10.1.1.0/24, 1 successors, FD is 256
         via Connected, Ethernet0
P 10.1.7.0/24, 1 successors, FD is 1024
```

```
         via 10.1.1.1 (1024/768), Ethernet0
P 10.1.6.0/24, 1 successors, FD is 256
         via Connected, Serial0
P 10.1.5.0/24, 1 successors, FD is 1536
         via 10.1.1.1 (1536/1280), Ethernet0
P 10.1.4.0/24, 2 successors, FD is 5376
         via 10.1.6.1 (5376/5120), Serial0
         via 10.1.1.1 (5376/5120), Ethernet0
Cayley#
```

## Example 7-10. Equal-cost load sharing will be performed between the two successors to 10.1.4.

```
Cayley#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
     10.0.0.0/24 is subnetted, 7 subnets
D       10.1.3.0 [90/768] via 10.1.1.1, 00:01:19, Ethernet0
D       10.1.2.0 [90/512] via 10.1.1.1, 00:01:19, Ethernet0
C       10.1.1.0 is directly connected, Ethernet0
D       10.1.7.0 [90/1024] via 10.1.1.1, 00:01:19, Ethernet0
C       10.1.6.0 is directly connected, Serial0
D       10.1.5.0 [90/1536] via 10.1.1.1, 00:01:19, Ethernet0
D       10.1.4.0 [90/5376] via 10.1.1.1, 00:01:19, Ethernet0
                 [90/5376] via 10.1.6.1, 00:01:19, Serial0
Cayley#
```

The topology table of Chanute (Example 7-11) shows several routes for which there is only one feasible successor. For example, the route to 10.1.6.0 has an FD of 768, and Wright (10.1.2.1) is the only feasible successor. Langley has a route to 10.1.6.0, but its metric is 256 x (2 + 1 + 1) = 1024, which is greater than the FD. Therefore, Langley's route to 10.1.6.0 does not satisfy the FC, and Langley does not qualify as a feasible successor.

## Example 7-11. Several of the subnets reachable from Chanute have only one feasible successor.

```
Chanute#show ip eigrp topology
IP-EIGRP Topology Table for process 1
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status
P 10.1.3.0/24, 1 successors, FD is 768
         via 10.1.2.1 (768/512), Ethernet0
         via 10.1.5.1 (1536/512), Serial0
P 10.1.2.0/24, 1 successors, FD is 256
         via Connected, Ethernet0
P 10.1.1.0/24, 1 successors, FD is 512
         via 10.1.2.1 (512/256), Ethernet0
P 10.1.7.0/24, 1 successors, FD is 1024
         via 10.1.2.1 (1024/768), Ethernet0
         via 10.1.5.1 (1280/256), Serial0
P 10.1.6.0/24, 1 successors, FD is 768
```

```
         via 10.1.2.1 (768/512), Ethernet0
P 10.1.5.0/24, 1 successors, FD is 1024
         via Connected, Serial0
P 10.1.4.0/24, 1 successors, FD is 5376
         via 10.1.2.1 (5376/5120), Ethernet0
Chanute#
```

If a feasible successor advertises a route for which the locally calculated metric is lower than the metric via the present successor, the feasible successor will become the successor. The following conditions can cause this situation to occur:

- A newly discovered route
- The cost of a successor's route increasing beyond that of a feasible successor
- The cost of a feasible successor's route decreasing to below the cost of the successor's route

For example, Example 7-12 shows that Lilienthal's successor to subnet 10.1.3.0 is Cayley (10.1.6.2). Suppose the cost of the link between Lilienthal and Wright is decreased to one. Wright (10.1.4.1) is advertising a distance of 512 to subnet 10.1.3.0; with the new cost of the link to Wright, Lilienthal's locally calculated metric to the subnet via that router is now 768. Wright will replace Cayley as the successor to subnet 10.1.3.0.

## Example 7-12. Topology table for Lilienthal.

```
Lilienthal#show ip eigrp topology
IP-EIGRP Topology Table for process 1
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status
P 10.1.3.0/24, 1 successors, FD is 1024
         via 10.1.6.2 (1024/768), Serial0
         via 10.1.4.1 (5632/512), Serial1
P 10.1.2.0/24, 1 successors, FD is 768
         via 10.1.6.2 (768/512), Serial0
         via 10.1.4.1 (5376/256), Serial1
P 10.1.1.0/24, 1 successors, FD is 512
         via 10.1.6.2 (512/256), Serial0
         via 10.1.4.1 (5376/256), Serial1
P 10.1.7.0/24, 1 successors, FD is 1280
         via 10.1.6.2 (1280/1024), Serial0
         via 10.1.4.1 (5888/768), Serial1
P 10.1.6.0/24, 1 successors, FD is 256
         via Connected, Serial0
P 10.1.5.0/24, 1 successors, FD is 1792
         via 10.1.6.2 (1792/1536), Serial0
         via 10.1.4.1 (6400/1280), Serial1
P 10.1.4.0/24, 1 successors, FD is 5120
         via Connected, Serial1
Lilienthal#
```

Next, suppose Lilienthal discovers a new neighbor that is advertising a distance of 256 to subnet 10.1.3.0. This distance is lower than the FD, so the new neighbor will become a feasible successor. Suppose further that the cost of the link to the new neighbor is 256. Lilienthal's locally calculated metric to 10.1.3.0 via the new neighbor will be 512. This metric is lower than the distance via Wright, so the new neighbor will become the successor to 10.1.3.0.

Feasible successors are important because they reduce the number of diffusing computations and therefore increase performance. Feasible successors also contribute to lower reconvergence times. If a link to a successor fails, or if the cost of the link increases beyond the FD, the router will first look into its topology table for a feasible successor. If one is found, it will become the successor; this selection normally occurs in the sub-second range. The router will begin a diffusing computation only if a feasible successor cannot be found. The key to successful EIGRP design, then, is ensuring that a feasible successor always exists for all destinations.

The following section gives a more formal set of rules for when and how a router will search for feasible successors. This set of rules is called the *DUAL finite state machine*.

## DUAL Finite State Machine

When an EIGRP router is performing no diffusing computations, each route is in the *passive state*. Referring to any of the topology tables in the previous section, a key to the left of each route indicates a passive state.

A router will reassess its list of feasible successors for a route, as described in the last section, any time an *input event* occurs. An input event can be the following:

- A change in the cost of a directly connected link
- A change in the state (up or down) of a directly connected link
- The reception of an update packet
- The reception of a query packet
- The reception of a reply packet

The first step in its reassessment is a *local computation* in which the distance to the destination is recalculated for all feasible successors. The possible results are

- If the feasible successor with the lowest distance is different from the existing successor, the feasible successor will become the successor.
- If the new distance is lower than the FD, the FD will be updated.
- If the new distance is different from the existing distance, updates will be sent to all neighbors.

While the router is performing a local computation, the route remains in the passive state. If a feasible successor is found, an update is sent to all neighbors and no state change occurs.

If a feasible successor cannot be found in the topology table, the router will begin a diffusing computation and the route will change to the *active state*. Until the diffusing computation is completed and the route transitions back to the passive state, the router cannot

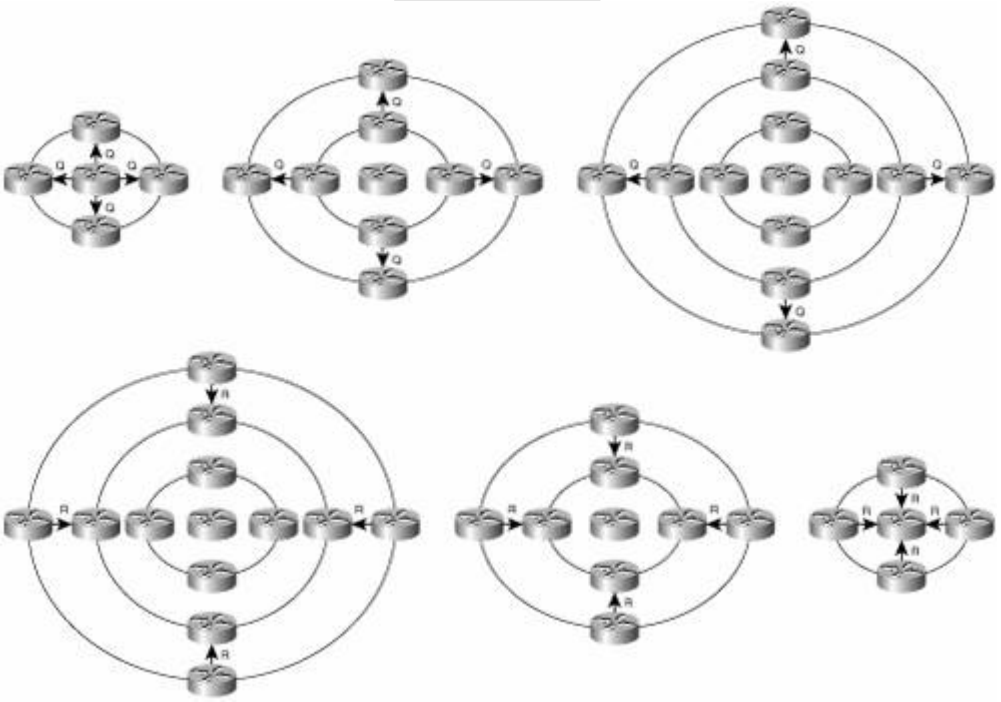
- Change the route's successor
- Change the distance it is advertising for the route
- Change the route's FD
- Begin another diffusing computation for the route

A router begins a diffusing computation by sending queries to all of its neighbors (Figure 7-6). The query will contain the new locally calculated distance to the destination. Each neighbor, upon receipt of the query, will perform its own local computation:

- If the neighbor has one or more feasible successors for the destination, it will send a reply to the originating router. The reply will contain that neighbor's minimum locally calculated distance to the destination.
- If the neighbor does not have a feasible successor, it too will change the route to the active state and will begin a diffusing computation.

Figure 7-6. A diffusing computation grows as queries are sent and shrinks as replies are received.

[View full size image]



For each neighbor to which a query is sent, the router will set a *reply status flag* (r) to keep track of all outstanding queries. The diffusing computation is complete when the router has received a reply to every query sent to every neighbor.

In some cases, a router does not receive a reply to every query sent. For example, this might happen in large networks with many low-bandwidth or low-quality links. At the beginning of the diffusing computation, an Active timer is set for three minutes. If all expected replies are not received before the Active time expires, the route is declared *stuck-in-active* (SIA). The neighbor or neighbors that did not reply will be removed from the neighbor table, and the diffusing computation will consider the neighbor to have responded with an infinite metric.

The default three-minute Active time can be changed or disabled with the command `timers active-time`. The deletion of a neighbor because of a lost query obviously can have disruptive results, and SIAs should never occur in a stable, well-designed network. The troubleshooting section of this chapter discusses SIAs in more detail. That same section describes a recent enhancement to the SIA procedures, using two new EIGRP messagesSIA Query and SIA Replythat both reduce the chance of occurrences of SIAs and, when they do occur, make them easier to troubleshoot.

At the completion of the diffusing computation, the originating router will set FD to infinity to ensure that any neighbor replying with a finite distance to the destination will meet the FC and become a feasible successor. For each of these replies, a metric is calculated based on the

distance advertised in the reply plus the cost of the link to the neighbor who sent the reply. A successor is selected based on the lowest metric, and FD is set to this metric. Any feasible successors that do not satisfy the FC for this new FD will be removed from the topology table. Note that a successor is not chosen until all replies have been received.

Because there are multiple types of input events that can cause a route to change state, some of which might occur while a route is active, DUAL defines multiple active states. A *query origin flag* (O) is used to indicate the current state. Figure 7-7 and Table 7-2 show the complete DUAL finite state machine.

Figure 7-7. The DUAL finite state machine. The query origin flag (O) marks the current state of the diffusing calculation. See Table 7-2 for a description of each input event (IE).

[View full size image]

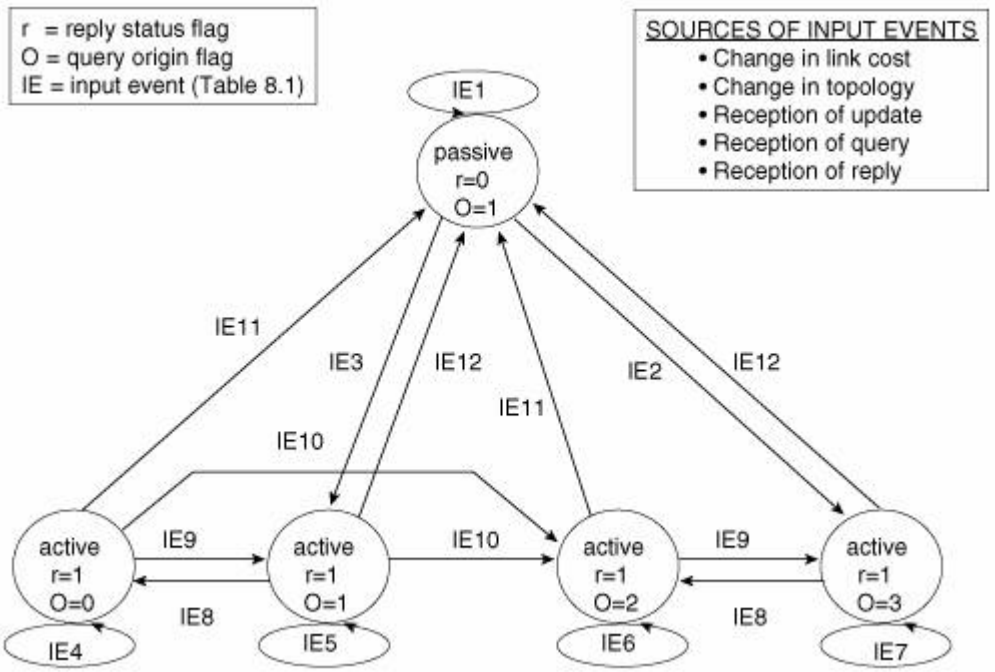



Table 7-2. Input events for the DUAL finite state machine.

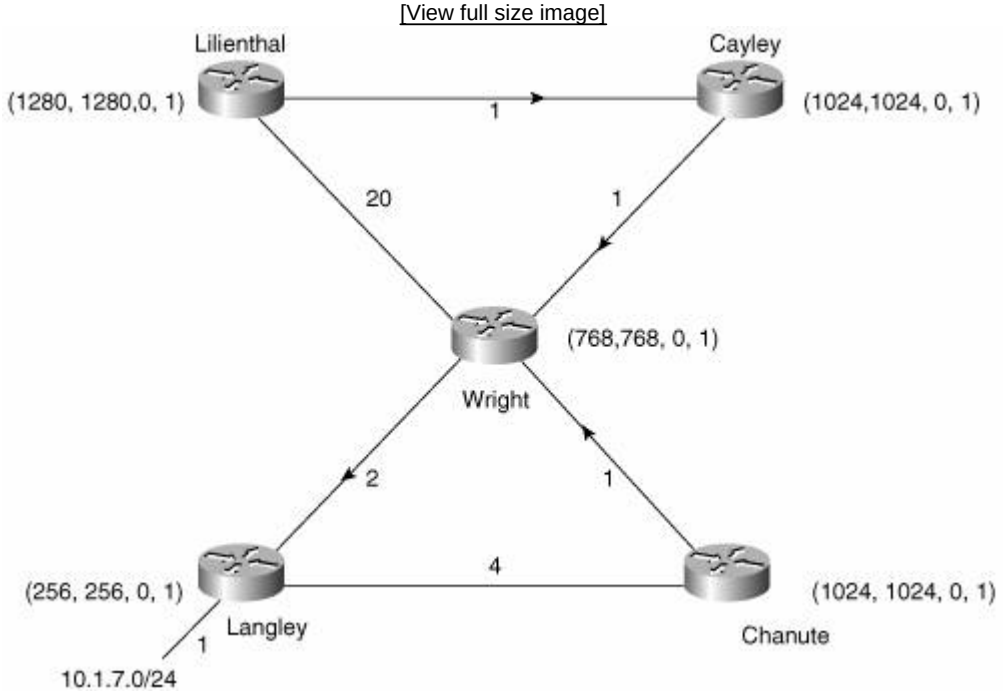
| Input Event | Description |
|---|---|
| IE1 | Any input event for which FC is satisfied or the destination is unreachable |
| IE2 | Query received from the successor; FC not satisfied |
| IE3 | Input event other than a query from the successor; FC not satisfied |
| IE4 | Input event other than last reply or a query from the successor |
| IE5 | Input event other than last reply, a query from the successor, or an increase in distance to destination |
| IE6 | Input event other than last reply |
| IE7 | Input event other than last reply or increase in distance to destination |

| Input Event | Description |
|---|---|
| IE8 | Increase in distance to destination |
| IE9 | Last reply received; FC not met with current FD |
| IE10 | Query received from the successor |
| IE11 | Last reply received; FC met with current FD |
| IE12 | Last reply received; set FD to infinity |

Two examples can help clarify the DUAL process. Figure 7-8 shows the example network, focusing only on each router's path to subnet 10.1.7.0; refer to Figure 7-5 for specific addresses. On the data links, an arrow indicates the successor each router is using to reach 10.1.7.0. In parentheses are each router's locally calculated distance to the subnet, the router's FD, the reply status flag (r), and the query origin flag (O), respectively. Active routers are indicated with a circle in subsequent figures and examples using this network.

Figure 7-8. All routes to subnet 10.1.7.0 are in the passive state, indicated by r = 0 and O = 1.

[View full size image]



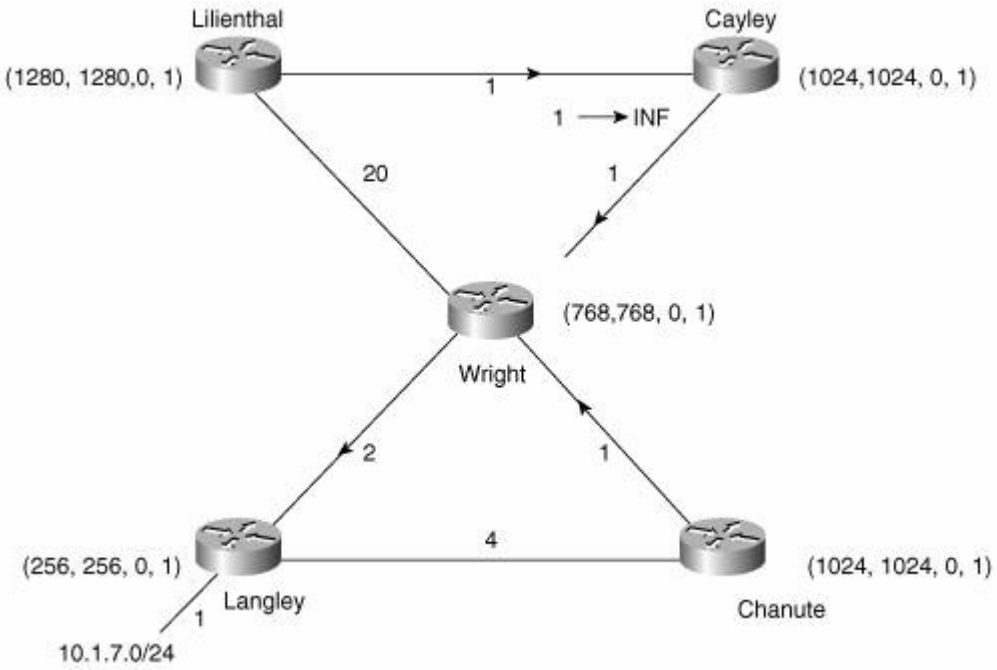

## Diffusing Computation: Example 1

This example focuses only on Cayley and its route to subnet 10.1.7.0. In Figure 7-9, the link between Cayley and Wright (10.1.1.1) has failed. EIGRP interprets the failure as a link with an infinite distance.[12] Cayley checks its topology table for a feasible successor to 10.1.7.0 and finds none (refer to Example 7-9).

[12] An infinite distance is indicated by a delay of 0xFFFFFFFF, or 4294967295.

Figure 7-9. The link between Wright and Cayley has failed, and Cayley does not have a feasible successor to subnet 10.1.7.0.

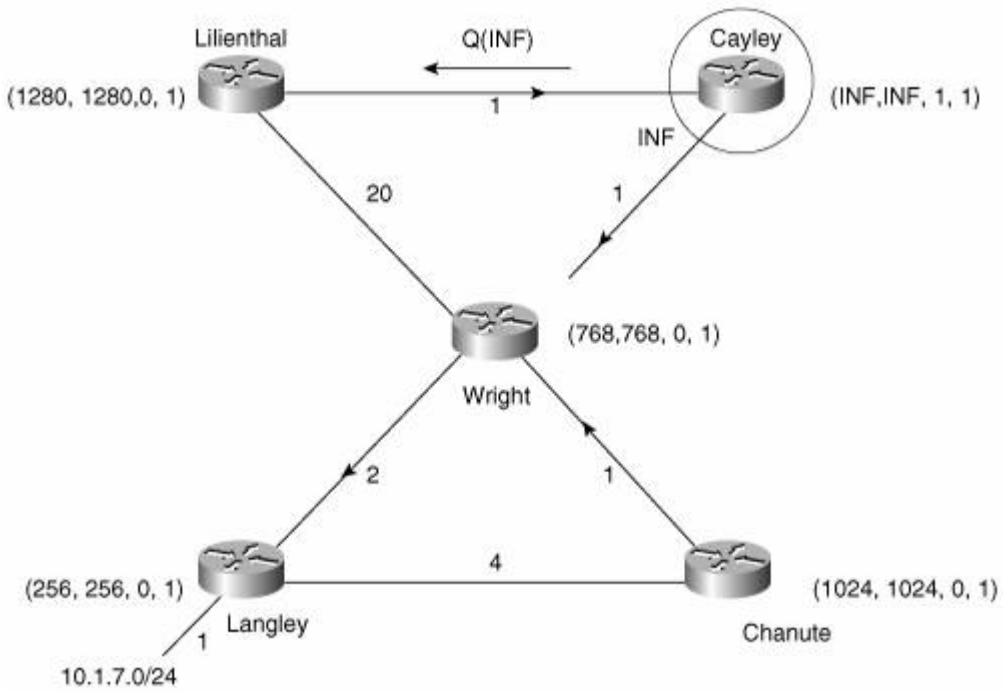
[View full size image]




Cayley's route becomes active (Figure 7-10). The distance and the FD of the route are changed to unreachable, and a query containing the new distance is sent to Cayley's neighbor, Lilienthal. Cayley's reply status flag for Lilienthal is set to one, indicating that a reply is expected. Because the input event was not the reception of a query (IE3), O=1.

Figure 7-10. Cayley's route to 10.1.7.0 transitions to active, and Lilienthal is queried for a feasible successor.

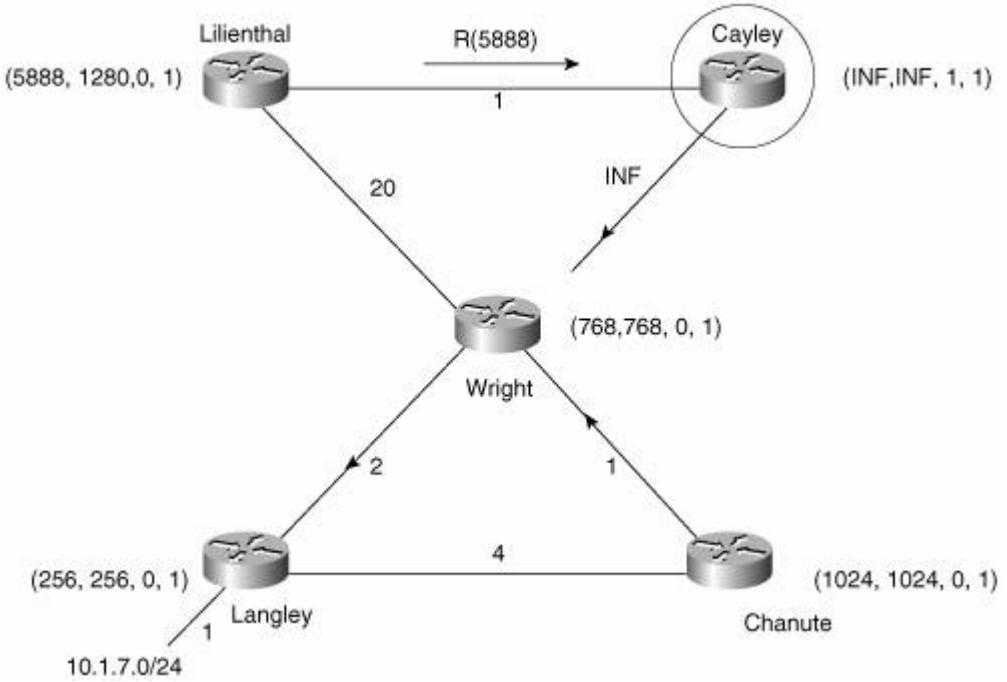
[View full size image]




Upon receipt of the query, Lilienthal performs a local computation (Figure 7-11). Because Lilienthal has a feasible successor for 10.1.7.0 (refer to Example 7-12), the route does not become active. Wright becomes the new successor, and a reply is sent with Lilienthal's distance to 10.1.7.0 via Wright. Because the distance to 10.1.7.0 has increased and the route did not become active, the FD is unchanged at Lilienthal.

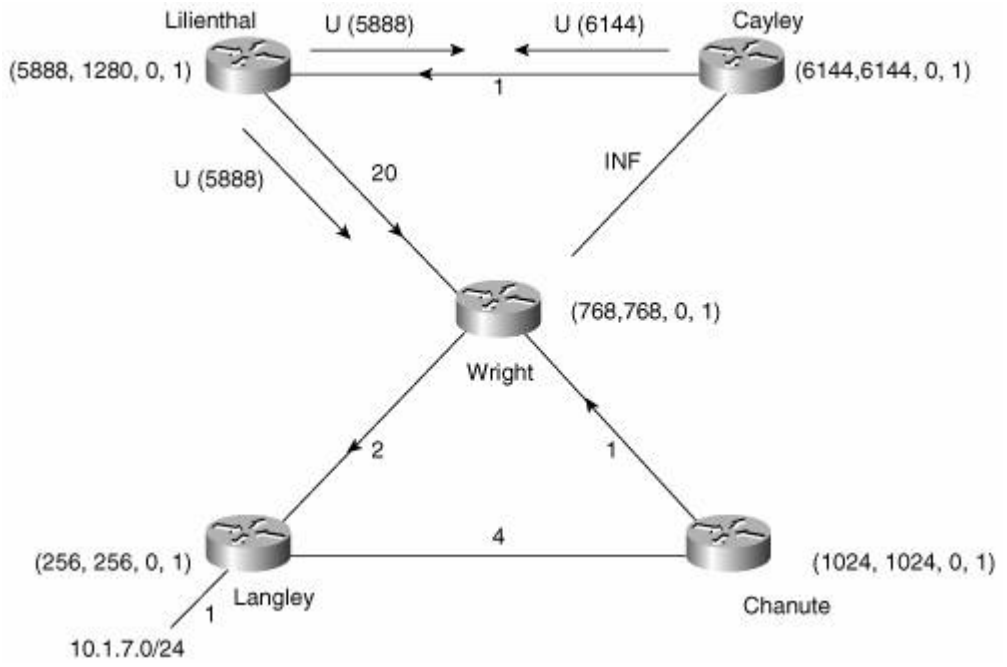
Figure 7-11. Lilienthal has a feasible successor to 10.1.7.0. A local computation is performed, a reply is sent to Cayley with the distance via Wright, and an update is sent to Wright.

[View full size image]



Upon receipt of the reply from Lilienthal, Cayley sets r=0 and the route becomes passive (Figure 7-12). Lilienthal becomes the new successor, and the FD is set to the new distance. Finally, an update is sent to Lilienthal with Cayley's locally calculated metric. Lilienthal will also send an update advertising its new metric.

Figure 7-12. Cayley's route to 10.1.7.0 becomes passive, and an update is sent to Lilienthal.




EIGRP packet activity can be observed with the debug command debug eigrp packets. By default, all EIGRP packets are displayed. Because Hellos and ACKs can make the debug output hard to follow, the command allows the use of optional keywords so that only specified packet types are displayed. In Example 7-13, debug eigrp packets query reply update is used to observe the packet activity at Cayley for the events described in this example.

Example 7-13. The EIGRP packet events described in this example can be observed in these debug messages.

```
Cayley#debug eigrp packet update query reply
EIGRP Packets debugging is on
    (UPDATE, QUERY, REPLY)
B#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state to down
EIGRP: Enqueueing QUERY on Serial0 iidbQ un/rely 0/1 serno 45-49
EIGRP: Enqueueing QUERY on Serial0 nbr 10.1.6.1 iidbQ un/rely 0/0 peerQ un/rely
0/0 serno 45-49
EIGRP: Sending QUERY on Serial0 nbr 10.1.6.1
  AS 1, Flags 0x0, Seq 45/64 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno
45-49
EIGRP: Received REPLY on Serial0 nbr 10.1.6.1
  AS 1, Flags 0x0, Seq 65/45 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
EIGRP: Enqueueing UPDATE on Serial0 iidbQ un/rely 0/1 serno 50-54
EIGRP: Enqueueing UPDATE on Serial0 nbr 10.1.6.1 iidbQ un/rely 0/0 peerQ un/rely
 0/0 serno 50-54
```

```
EIGRP: Sending UPDATE on Serial0 nbr 10.1.6.1
  AS 1, Flags 0x0, Seq 46/66 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno
50-54
EIGRP: Received UPDATE on Serial0 nbr 10.1.6.1
  AS 1, Flags 0x0, Seq 67/46 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
```

*Flags*, in the debug messages, indicate the state of the flags in the EIGRP packet header (see the section "EIGRP Packet Header" later in this chapter). 0x0 indicates that no flags are set. 0x1 indicates that the *initialization* bit is set. This flag is set when the enclosed route entries are the first in a new neighbor relationship. 0x2 indicates that the *conditional receive* bit is set. This flag is used in the proprietary Reliable Multicasting algorithm:

- Seq is the Packet Sequence Number/Acknowledged Sequence Number.
- idbq indicates packets in the input queue/packets in the output queue of the interface.
- iidbq indicates unreliable multicast packets awaiting transmission/reliable multicast packets awaiting transmission on the interface.
- peerQ indicates unreliable unicast packets awaiting transmission/reliable unicast packets awaiting transmission on the interface.
- serno is a pointer to a doubly linked serial number for the route. This is used by an internal (and proprietary) mechanism for tracking the correct route information in a rapidly changing topology.

## Diffusing Computation: Example 2

This example focuses on Wright and its route to subnet 10.1.7.0. Although the combination of input events portrayed here (the delay of a link changing twice during a diffusing computation) is unlikely to occur in real life, the example shows how DUAL handles multiple metric changes.

In Figure 7-13, the cost of the link between Wright and Langley changes from 2 to 10. The distance to 10.1.7.0 via Langley now exceeds Wright's FD, causing that router to begin a local computation. The metric is updated, and Wright sends updates to all its neighbors except the neighbor on the link whose cost changed (Figure 7-14).

Figure 7-13. Cayley's route to 10.1.7.0 becomes passive, and an update is sent to Lilienthal.

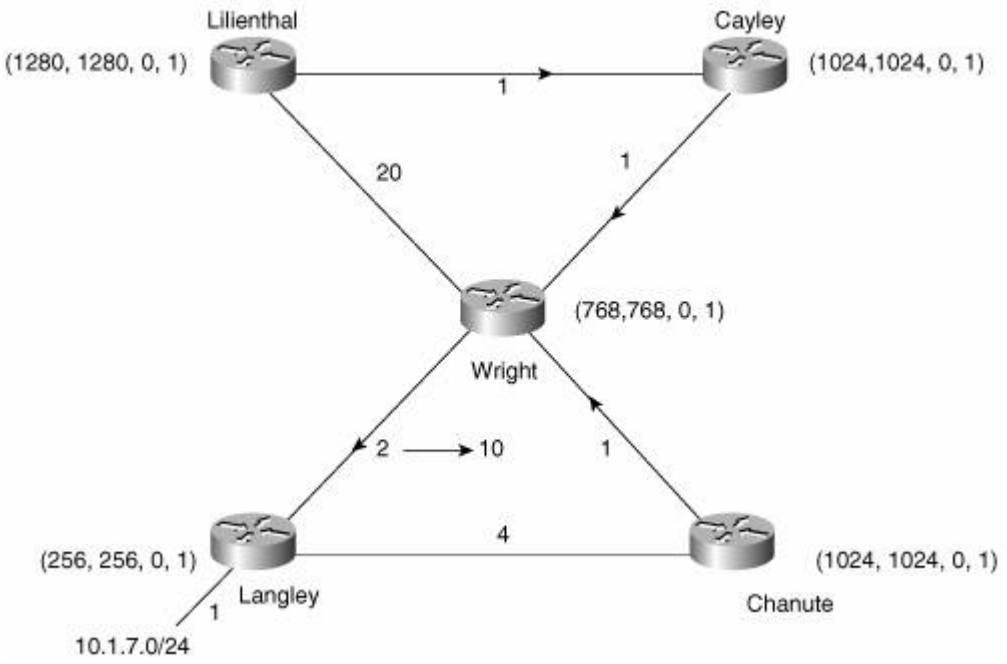
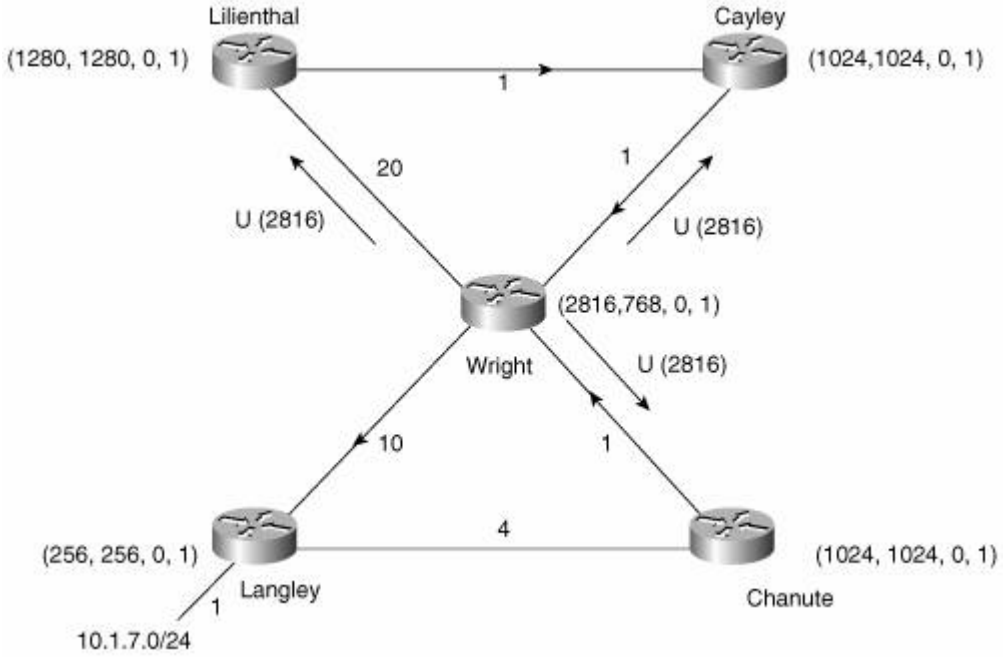


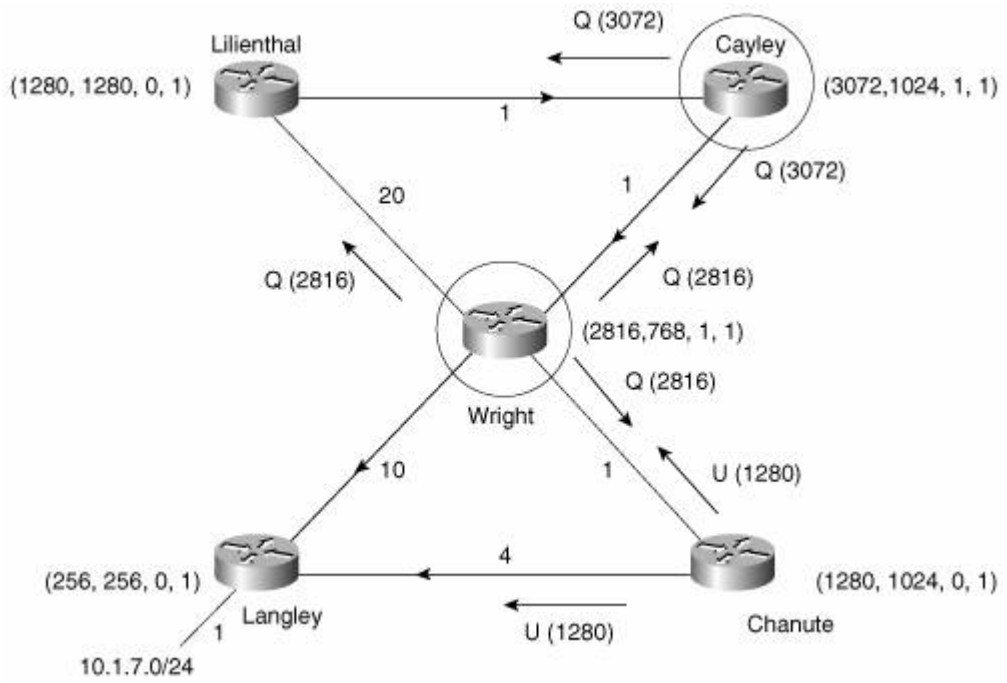

Figure 7-14. Wright sends updates containing the new metric to all neighbors except Langley.




Note that Langley was the only feasible successor to subnet 10.1.7.0 because Chanute's locally calculated metric is higher than Wright's FD (1024 > 768). The metric increase on the Wright-Langley link causes Wright to look in its topology table for a new successor. Because the only feasible successor that Wright can find in its topology table is Langley, the route becomes active. Queries are sent to the neighbors (Figure 7-15).

Figure 7-15. Wright's route to 10.1.7.0 becomes active, and it queries

its neighbors for a feasible successor. In response to the earlier update from Wright, Cayley makes its route active and queries its neighbors; also, Chanute changes its metric and sends updates.




At the same time, the updates sent by Wright in Figure 7-14 cause Cayley, Lilienthal, and Chanute to perform a local calculation.

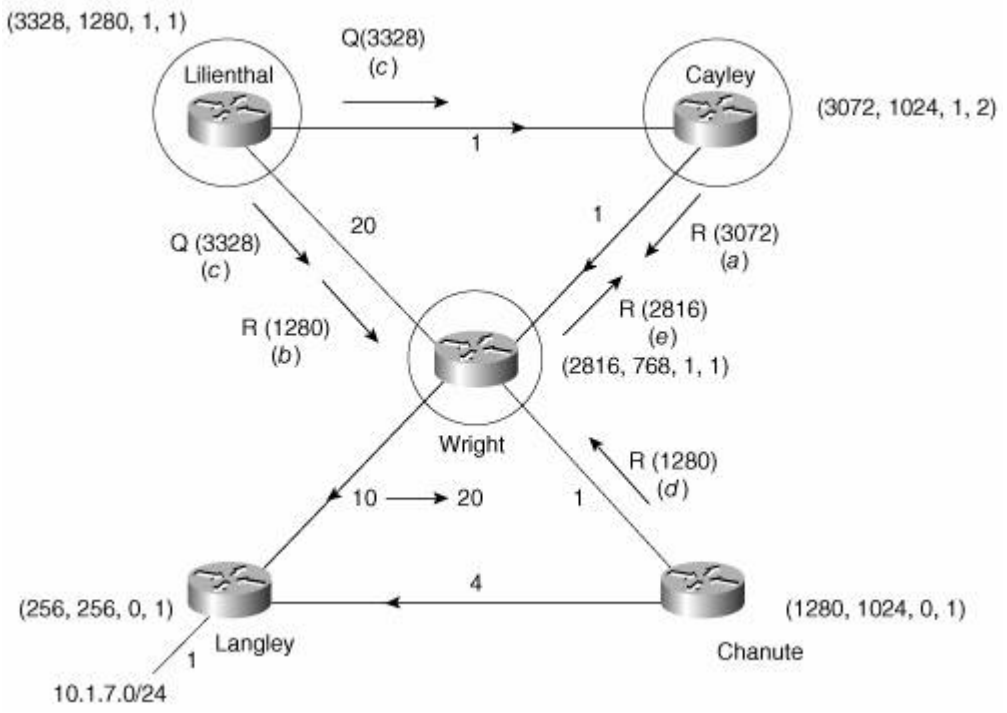
At Cayley, the route via Wright now exceeds Cayley's FD (2816 > 1024). The route goes active and queries are sent to the neighbors.

Lilienthal is using Cayley as a successor and in Figure 7-15 has not yet received the query from Cayley. Therefore, Lilienthal merely recalculates the metric of the path via Wright, finds that it no longer meets the FC, and drops the path from the topology table.

At Chanute, Wright is the successor. Because Wright's advertised distance no longer meets the FC at Chanute (2816 > 1024) and because Chanute does have a feasible successor (refer to Example 7-11), Wright is deleted from Chanute's topology table. Langley becomes the successor at Chanute; the metric is updated, and Chanute sends updates to its neighbors (refer to Figure 7-15). The route at Chanute never becomes active.

Cayley, Lilienthal, and Chanute each respond differently to the queries from Wright (Figure 7-16).

Figure 7-16. Cayley (a) replies to Wright's query. Lilienthal (b) replies to Wright's query and (c) goes active for the route, sending queries in response to Cayley's query. Chanute (d) replies to Wright's query. Wright (e) replies to Cayley's query.




Cayley is already active. Because the input event is a query from the successor, the query origin flag will be 2 (O=2) (refer to Figure 7-7 and Table 7-2).

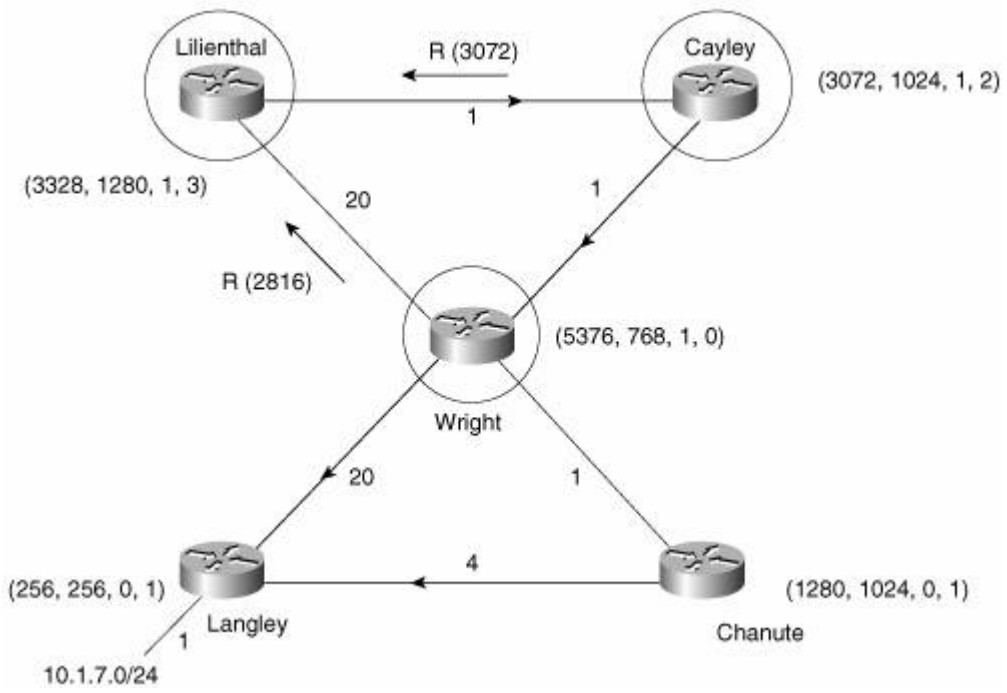
Lilienthal, upon the receipt of Wright's query, sends a response with its distance via Cayley. However, just after the reply is sent, Lilienthal receives the query from Cayley. The FD is exceeded, the metric is updated, and the route goes active. Lilienthal queries its neighbors.

Chanute, which has already switched to Langley as its successor, merely sends a reply.

While all this is going on, Figure 7-16 shows that the cost of the link between Wright and Langley again increases, from 10 to 20. Wright will recalculate the metric to 10.1.7.0 based on this new cost, but because the route is active, neither the FD nor the distance it advertises will change until the route becomes passive.

According to Figure 7-7 and Table 7-2, an increase in the distance to the destination while the route is active will cause O=0 (Figure 7-17). Wright responds to the query from Lilienthal. The distance it reports is the distance it had when the route first became active (remember, the advertised distance cannot change while the route is active). Cayley also sends a reply to Lilienthal's query.

Figure 7-17. Wright cannot change the metric it advertises until the route becomes passive.




Lilienthal, having received replies to all its queries, will transition the route to passive (Figure 7-18). A new FD is set for the route. Cayley remains the successor because its advertised route is lower than the FD at Lilienthal. Lilienthal also sends a reply to Cayley's query.

Figure 7-18. Having received the last expected reply, Lilienthal changes its route to the passive state (r=0, O=1).

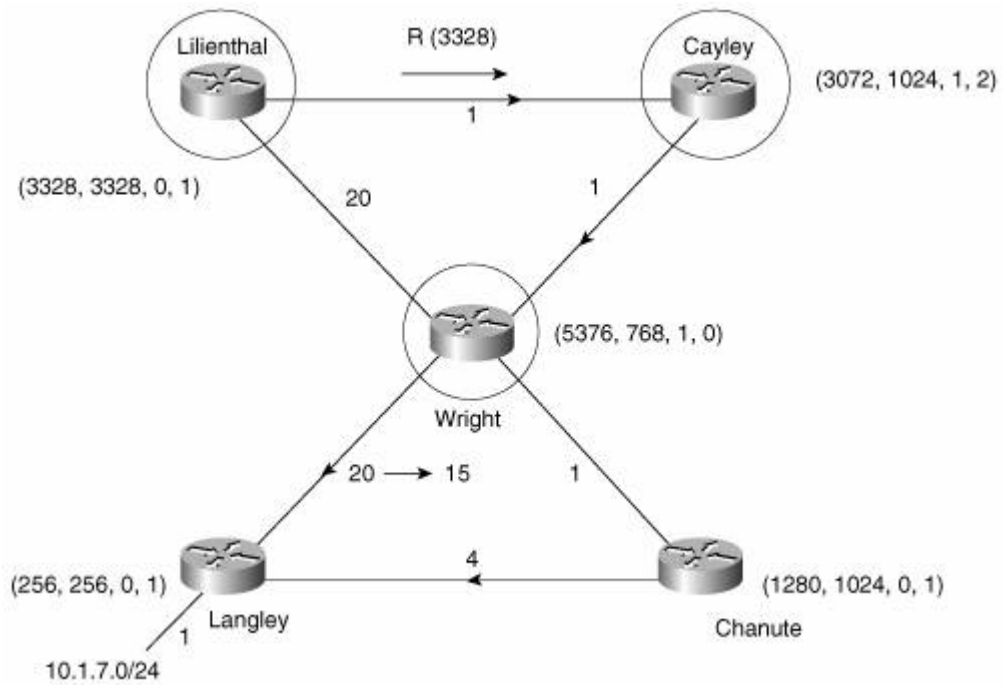
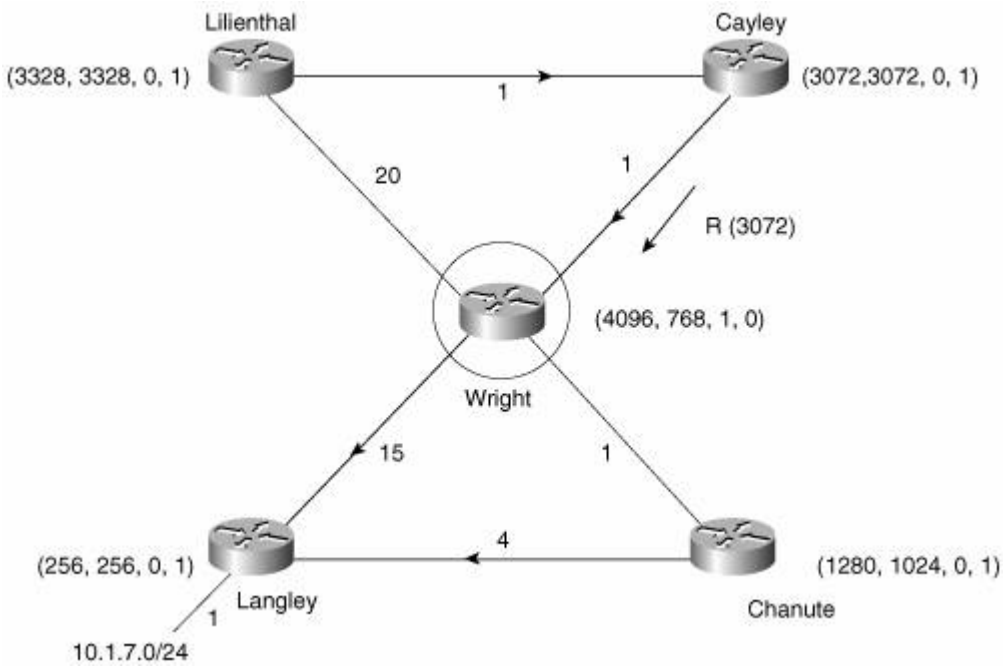



Figure 7-18 also shows that the distance has changed again, from 20 to 15. Wright recalculates its local distance for the route again, to 4096 (Figure 7-19). If it were to receive a query before going passive, the route would still be advertised with a distance of 2816the distance when the

route went active.

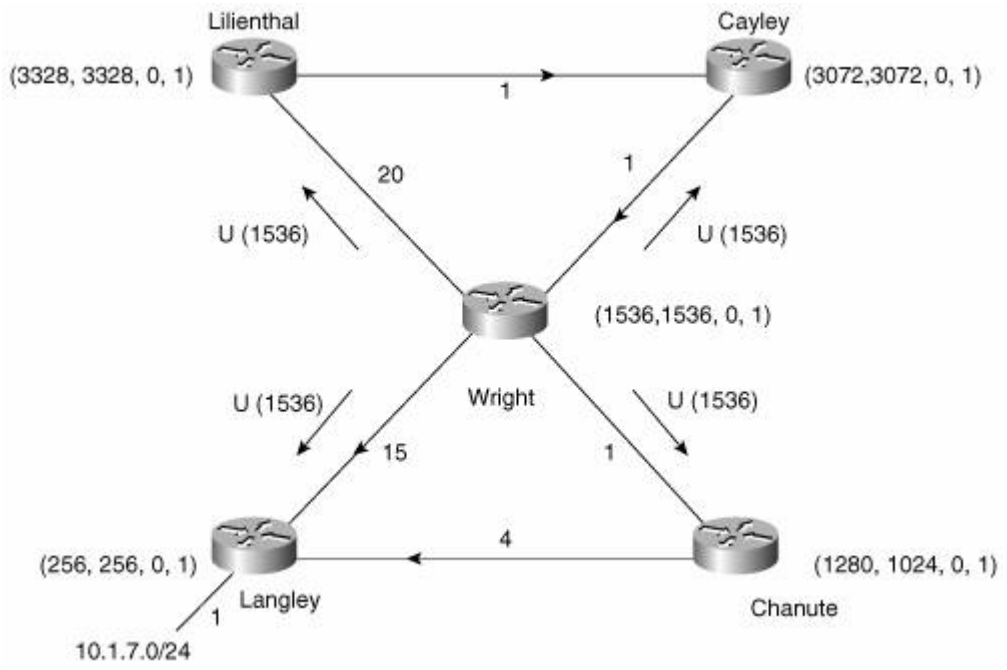
Figure 7-19. Having received its last expected reply, Cayley changes its route to the passive state.




When Cayley receives the reply to its query, its route to 10.1.7.0 also becomes passive (Figure 7-19) and a new FD is set. Although Wright's locally calculated metric is 4096, the last metric it advertised was 2816. Therefore, Wright meets the FC at Cayley and becomes the successor to 10.1.7.0. A reply is sent to Wright.

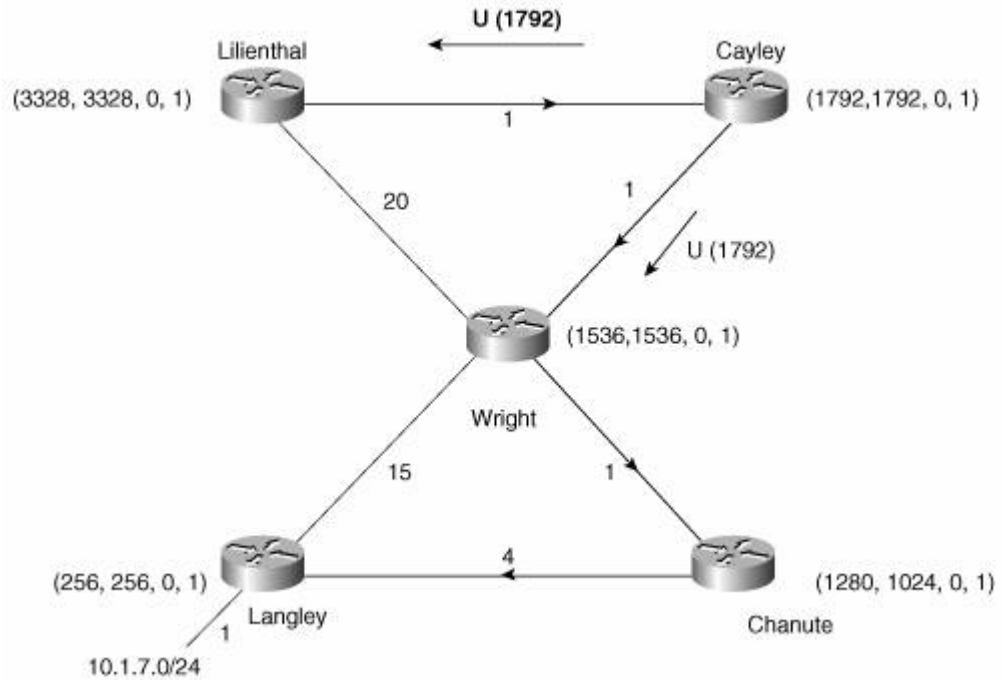
In Figure 7-20, Wright has received a reply to every query it sent, and its route becomes passive. It chooses Chanute as its new successor and changes the FD to the sum of Chanute's advertised distance and the cost of the link to that neighbor. Wright sends an update to all its neighbors, advertising the new locally calculated metric.

Figure 7-20. Wright transitions to passive, chooses Chanute as the successor, changes the FD, and updates all neighbors.



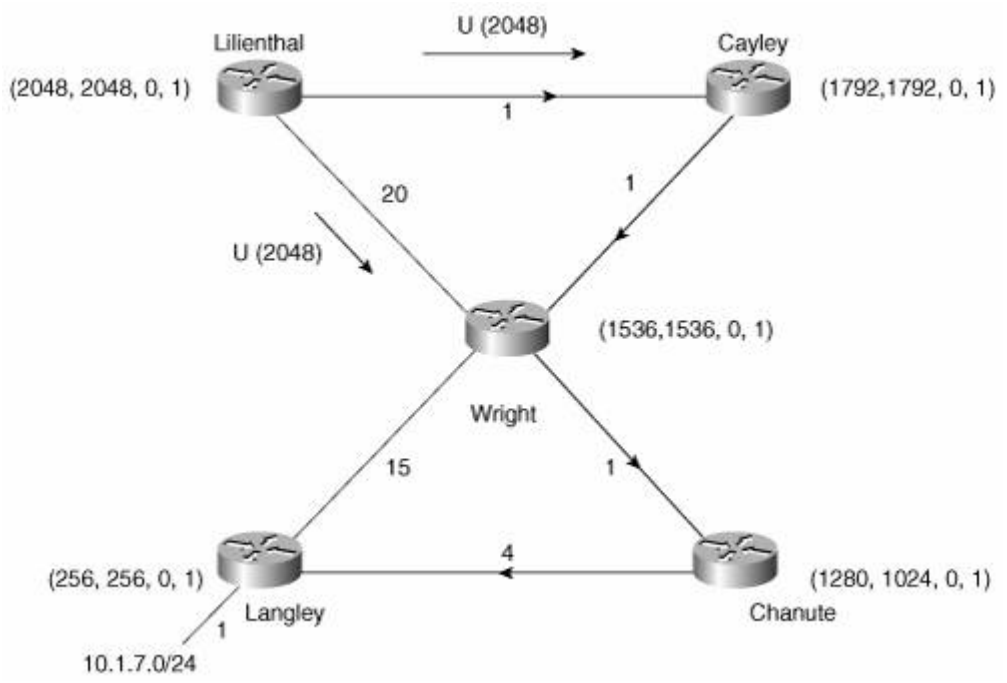

Cayley is already using Wright as the successor. When it receives the update from Wright with a lower cost, it changes its locally calculated metric and FD accordingly and updates its neighbors (Figure 7-21).

Figure 7-21. Cayley recalculates its metric, changes the FD based on the lower cost advertised by Wright, and updates its neighbors.




The update from Cayley has no effect at Wright because it does not satisfy the FC there. At Lilienthal the update causes a local computation. Lilienthal lowers the metric, lowers the FD, and updates its neighbors (Figure 7-22).

Figure 7-22. Lilienthal recalculates its metric, changes the FD based on the update from Cayley, and updates its neighbors.




Although they are rather elaborate and might take several readings to fully understand, this and the previous example contain the important core behavior of diffusing computations:

- Any time an input event occurs, perform a local calculation.
- If one or more feasible successors are found in the topology table, take the ones with the lowest metric cost as the successors.
- If no feasible successor can be found, make the route active and query the neighbors for a feasible successor.
- Keep the route active until all queries are answered by reply or by the expiration of the active timer.
- If the diffusing calculation does not result in the discovery of a feasible successor, declare the destination unreachable.

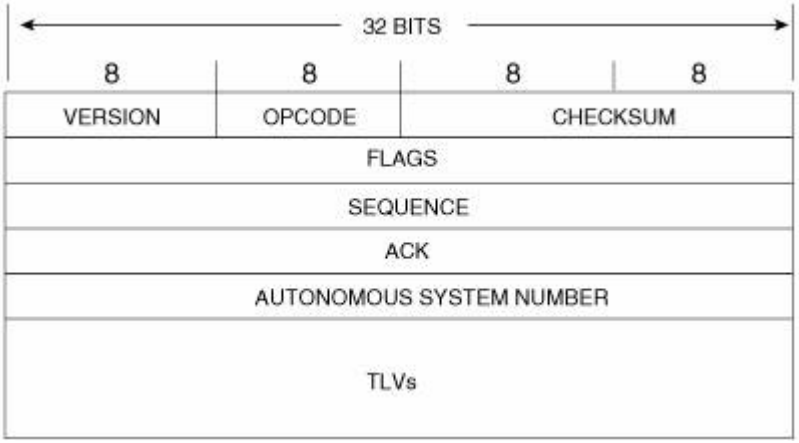
## EIGRP Packet Formats

The IP header of an EIGRP packet specifies protocol number 88, and the maximum length of the packet will be the IP maximum transmission unit (MTU) of the interface on which it is transmittedusually 1500 octets. Following the IP header is an EIGRP header followed by various *Type/Length/Value* (TLV) triplets. These TLVs will not only carry the route entries but also may provide fields for the management of the DUAL process, multicast sequencing, and IOS software versions.

### EIGRP Packet Header

Figure 7-23 shows the EIGRP header, which begins every EIGRP packet.

Figure 7-23. EIGRP packet header.




- Version specifies the particular version of the originating EIGRP process. The version of the EIGRP process itself has not changed since its release.
- Opcode specifies the EIGRP packet type, as shown in Table 7-3. Although the IPX SAP packet type is included in the table, a discussion of IPX EIGRP is outside the scope of this book.

Table 7-3. EIGRP packet types.

| Opcode | Type |
|---|---|
| 1 | Update |
| 3 | Query |
| 4 | Reply |
| 5 | Hello |
| 6 | IPX SAP |
| 10 | SIA Query |
| 11 | SIA Reply |

- Checksum is a standard IP checksum. It is calculated for the entire EIGRP packet, excluding the IP header.
- Flags currently include just two flags. The right-most bit is *Init*, which when set (0x00000001) indicates that the enclosed route entries are the first in a new neighbor relationship. The second bit (0x00000002) is the Conditional Receive bit, used in the proprietary Reliable Multicasting algorithm.

- Sequence is the 32-bit sequence number used by the RTP.
- ACK is the 32-bit sequence number last heard from the neighbor to which the packet is being sent. A Hello packet with a nonzero ACK field will be treated as an ACK packet rather than as a Hello. Note that an ACK field will only be nonzero if the packet itself is unicast because acknowledgments are never multicast.
- Autonomous System Number is the identification number of the EIGRP domain.

Following the header are the TLVs, whose various types are listed in Table 7-4. IPX and AppleTalk types are included, although they are not discussed in this book. Each TLV includes one of the two-octet type numbers listed in Table 7-4, a two-octet field specifying the length of the TLV, and a variable field whose format is determined by the type.

Table 7-4. Type/Length/Value (TLV) types.

| Number | TLV Type |
|---|---|
| *General TLV Types* | |
| 0x0001 | EIGRP Parameters |
| 0x0003 | Sequence |
| 0x0004 | Software Version[*] |
| 0x0005 | Next Multicast Sequence |
| *IP-Specific TLV Types* | |
| 0x0102 | IP Internal Routes |
| 0x0103 | IP External Routes |
| *AppleTalk-Specific TLV Types* | |
| 0x0202 | AppleTalk Internal Routes |
| 0X0203 | AppleTalk External Routes |
| 0x0204 | AppleTalk Cable Configuration |
| *IPX-Specific TLV Types* | |
| 0x0302 | IPX Internal Routes |
| 0x0303 | IPX External Routes |

[*] This packet indicates whether the older software release is running (software version 0) or the newer release, as of IOS 10.3(11), 11.0(8), and 11.1(3), is running (version 1).

## General TLV Fields

These TLVs carry EIGRP management information and are not specific to any one routed protocol. The Parameters TLV, which is used to convey metric weights and the hold time, is shown in Figure 7-24. The Sequence, Software Version, and Next Multicast Sequence TLVs are used by the Cisco proprietary Reliable Multicast algorithm and are beyond the scope of this book.

Figure 7-24. EIGRP Parameters TLV.

## IP-Specific TLV Fields

Each Internal and External Routes TLV contains one route entry. Every Update, Query, and Reply packet contains at least one Routes TLV.

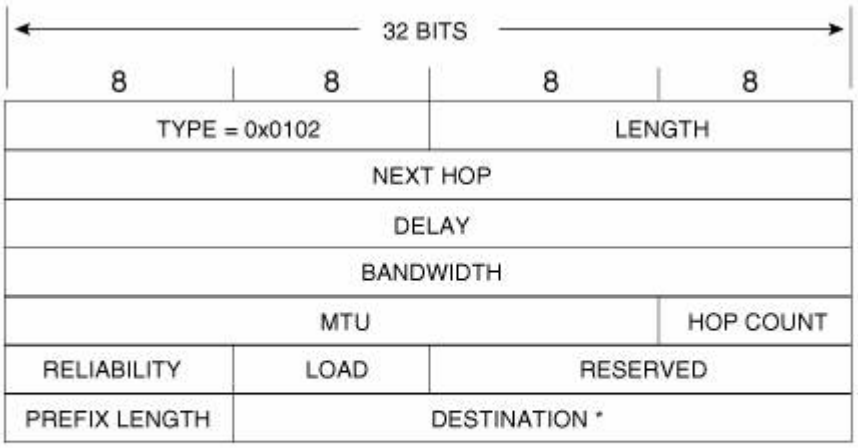
The Internal and External Routes TLVs include metric information for the route. As noted earlier, the metrics used by EIGRP are the same metrics used by IGRP, although scaled by 256.

## IP Internal Routes TLV

An internal route is a path to a destination within the EIGRP autonomous system. The format of the Internal Routes TLV is shown in Figure 7-25.

Figure 7-25. The IP Internal Routes TLV.

[View full size image]

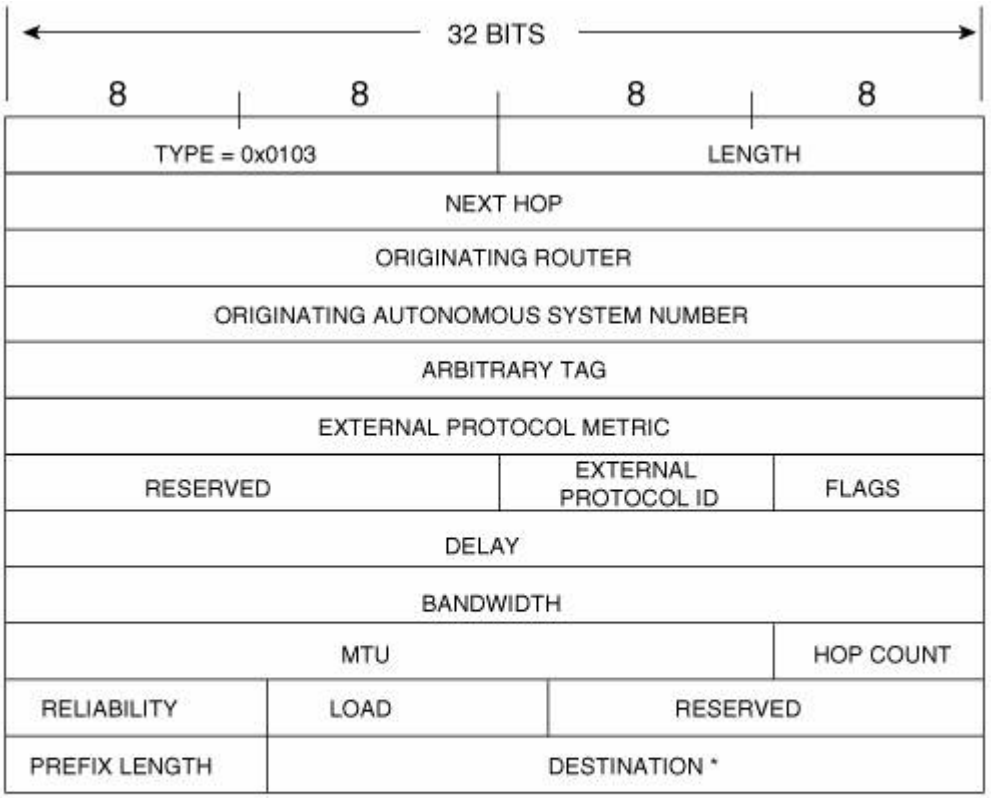



* This field is variable. If it is less than or more than 3 octets, the TLV will be padded with zeros to the next 4-octet boundary. For example, if the destination address is 10.1, the Destination field will be 2 octets and will be followed with a pad of 0x00. If the address is 192.168.16.64, the Destination field will be 4 octets and will be followed with a pad of 0x000000.

- Next Hop is the next-hop IP address. This address might or might not be the address of the originating router.
- Delay is the sum of the configured delays expressed in units of 10 microseconds. Notice that unlike the 24-bit delay field of the IGRP packet, this field is 32 bits. This larger field accommodates the 256 multiplier used by EIGRP. A delay of 0xFFFFFFFF indicates an unreachable route.
- Bandwidth is 256 x $BW_{IGRP(min)}$, or 2,560,000,000 divided by the lowest configured bandwidth of any interface along the route. Like Delay, this field is also eight bits larger than the IGRP field.
- MTU is the smallest Maximum Transmission Unit of any link along the route to the destination. Although an included parameter, it has never been used in the calculation of metrics.
- Hop Count is a number between 0x01 and 0xFF indicating the number of hops to the destination. A router will advertise a directly connected network with a hop count of 0; subsequent routers will record and advertise the route relative to the next-hop router.
- Reliability is a number between 0x01 and 0xFF that reflects the total outgoing error rates of the interfaces along the route, calculated on a five-minute exponentially weighted average. 0xFF indicates a 100 percent reliable link.
- Load is also a number between 0x01 and 0xFF, reflecting the total outgoing load of the interfaces along the route, calculated on a five-minute exponentially weighted average. 0x01 indicates a minimally loaded link.
- Reserved is an unused field and is always 0x0000.
- Prefix Length specifies the number of network bits of the address mask. *Destination* is the destination address of the route. Although the field is shown as a three-octet field in Figure 7-25 and Figure 7-26 (see next section), the field varies with the specific address. For example, if the route is to 10.1.0.0/16, the prefix length will be 16 and the destination will be a two-octet field containing 10.1. If the route is to 192.168.17.64/27, the prefix length will be 27 and the destination will be a four-octet field containing 192.168.16.64. If this field is not exactly three octets, the TLV will be padded with zeros to make it end on a four-octet boundary.

Figure 7-26. The IP External Routes TLV.

[View full size image]




* This field is variable. If it is less than or more than 3 octets, the TLV will be padded with zeros to the next 4-octet boundary. For example, if the destination address is 10.1, the Destination field will be 2 octets and will be followed with a pad of 0x00. If the address is 192.168.16.64, the Destination field will be 4 octets and will be followed with a pad of 0x000000.

## IP External Routes TLV

An external route is a path that leads to a destination outside of the EIGRP autonomous system and that has been redistributed into the EIGRP domain. Figure 7-26 shows the format of the External Routes TLV:

- Next Hop is the next-hop IP address. On a multiaccess network, the router advertising the route might not be the best next-hop router to the destination. For example, an EIGRP-speaking router on an Ethernet link might also be speaking BGP and might be advertising a BGP-learned route into the EIGRP autonomous system. Because other routers on the link do not speak BGP, they might have no way of knowing that the interface to the BGP speaker is the best next-hop address. The Next Hop field allows the "bilingual" router to tell its EIGRP neighbors, "Use address A.B.C.D as the next hop instead of using my interface address."
- Originating Router is the IP address or router ID of the router that redistributed the external route into the EIGRP autonomous system.
- Originating Autonomous System Number is the autonomous system number of the router originating the route.
- Arbitrary Tag may be used to carry a tag set by route maps. See Chapter 14 for information on he use of route maps.
- External Protocol Metric is, as the name implies, the metric of the external protocol. This field is used, when distributing with IGRP, to track the IGRP metric.

- Reserved is an unused field and is always 0x0000.
- External Protocol ID specifies the protocol from which the external route was learned. Table 7-5 lists the possible values of this field.

Table 7-5. Values of the External Protocol ID field.

| Code | External Protocol |
|---|---|
| 0x01 | IGRP |
| 0x02 | EIGRP |
| 0x03 | Static Route |
| 0x04 | RIP |
| 0x05 | Hello |
| 0x06 | OSPF |
| 0x07 | IS-IS |
| 0x08 | EGP |
| 0x09 | BGP |
| 0x0A | IDRP |
| 0x0B | Connected Link |

- Flags currently constitute just two flags. If the right-most bit of the eight-bit field is set (0x01), the route is an external route. If the second bit is set (0x02), the route is a candidate default route. Default routes are discussed in Chapter 12.

The remaining fields describe the metrics and the destination address. The descriptions of these fields are the same as those given in the discussion of the Internal Routes TLV.

## Address Aggregation

Chapter 1, "Routing Basics," introduced the practice of subnetting, in which the address mask is extended into the host space to address multiple data links under one major network address. Chapter 6 introduced the practice of variable-length subnet masking, in which the address mask is extended even more to create subnets within subnets.

From an opposite perspective, a subnet address may be thought of as a summarization of a group of sub-subnets. And a major network address may be thought of as a summarization of a group of subnet addresses. In each case, the summarization is achieved by reducing the length of the address mask.

Address aggregation takes summarization a step further by breaking the class limits of major network addresses. An aggregate address represents a numerically contiguous group of network addresses, known as a supernet.[13] Figure 7-27 shows an example of an aggregate address.

[13] An aggregate is, more correctly, any summarized group of addresses. For clarity, in this book the term aggregate refers

to a summarization of a group of major network addresses.

Figure 7-27. This group of network addresses can be represented with a single aggregate address, or subnet.

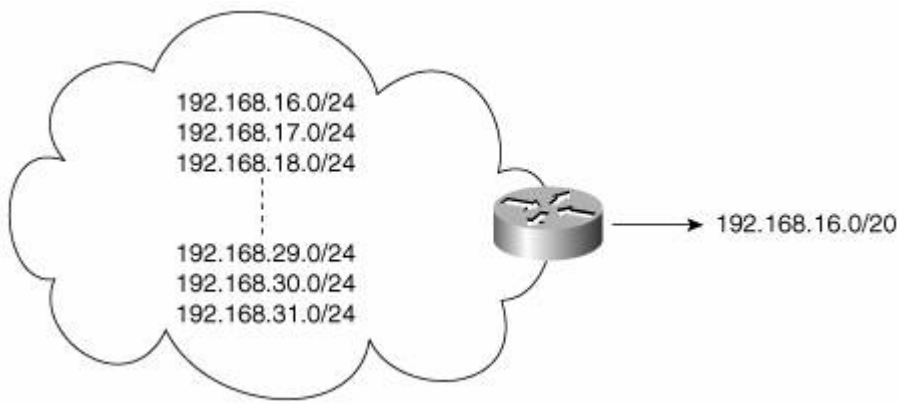



Figure 7-28 shows how the aggregate address of Figure 7-27 is derived. For a group of network addresses, find the bits that are common to all of the addresses and mask these bits. The masked portion is the aggregate address.

Figure 7-28. The aggregate address is derived by masking all the common bits of a group of numerically contiguous network addresses.

```
11111111111111111111111100000000 = 24-bit mask
11000000101010000001000000000000 = 192.168.16.0/24
11000000101010000001000100000000 = 192.168.17.0/24
11000000101010000001001000000000 = 192.168.18.0/24
11000000101010000001001100000000 = 192.168.19.0/24
11000000101010000001010000000000 = 192.168.20.0/24
11000000101010000001010100000000 = 192.168.21.0/24
11000000101010000001011000000000 = 192.168.22.0/24
11000000101010000001011100000000 = 192.168.23.0/24
11000000101010000001100000000000 = 192.168.24.0/24
11000000101010000001100100000000 = 192.168.25.0/24
11000000101010000001101000000000 = 192.168.26.0/24
11000000101010000001101100000000 = 192.168.27.0/24
11000000101010000001110000000000 = 192.168.28.0/24
11000000101010000001110100000000 = 192.168.29.0/24
11000000101010000001111000000000 = 192.168.30.0/24
11000000101010000001111100000000 = 192.168.31.0/24
11000000101010000001000000000000 = 192.168.16.0/20
```

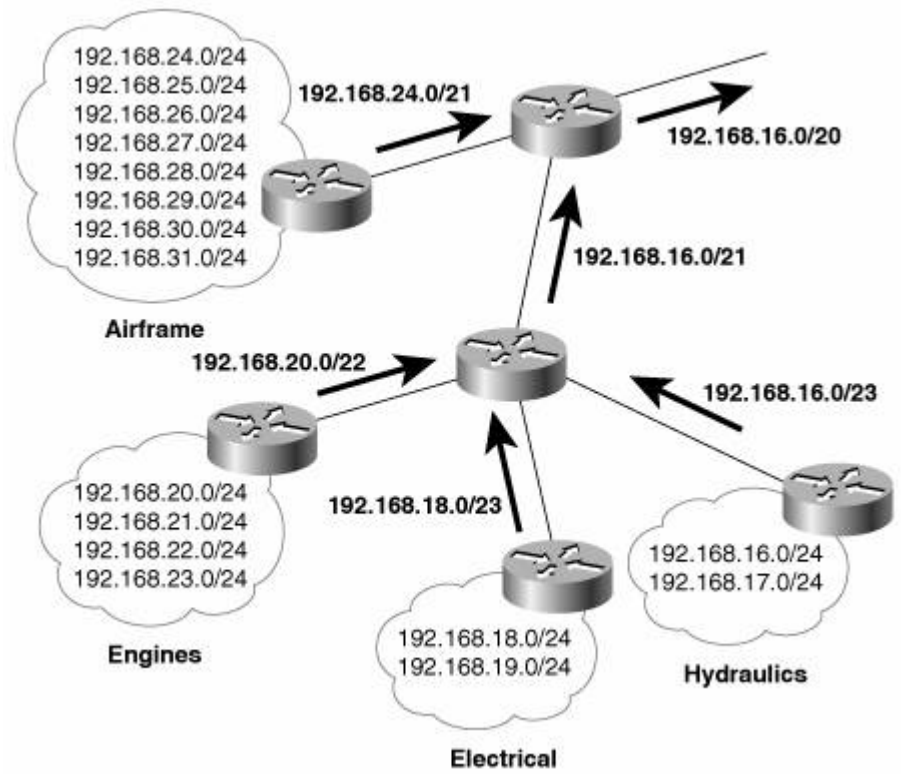
When designing a supernet, it is important that the member addresses should form a complete, contiguous set of the formerly masked bits. In Figure 7-28, for example, the 20-bit mask of the aggregate address is four bits less than the mask of the member addresses. Of the four "difference" bits, notice that they include every possible bit combination from 0000 to 1111. Failure to follow this design rule could make the addressing scheme confusing, could reduce the efficiency of aggregate routes, and might lead to routing loops and black holes.

The obvious advantage of summary addressing is the conservation of network resources.

Bandwidth is conserved by advertising fewer routes, and CPU cycles are conserved by processing fewer routes. Most important, memory is conserved by reducing the size of the route tables.

Classless routing, VLSM, and aggregate addressing together provide the means to maximize resource conservation by building address hierarchies. Unlike IGRP, EIGRP supports all of these addressing strategies. In Figure 7-29, the engineering division of Treetop Aviation has been assigned 16 class C addresses. These addresses have been assigned to the various departments according to need.

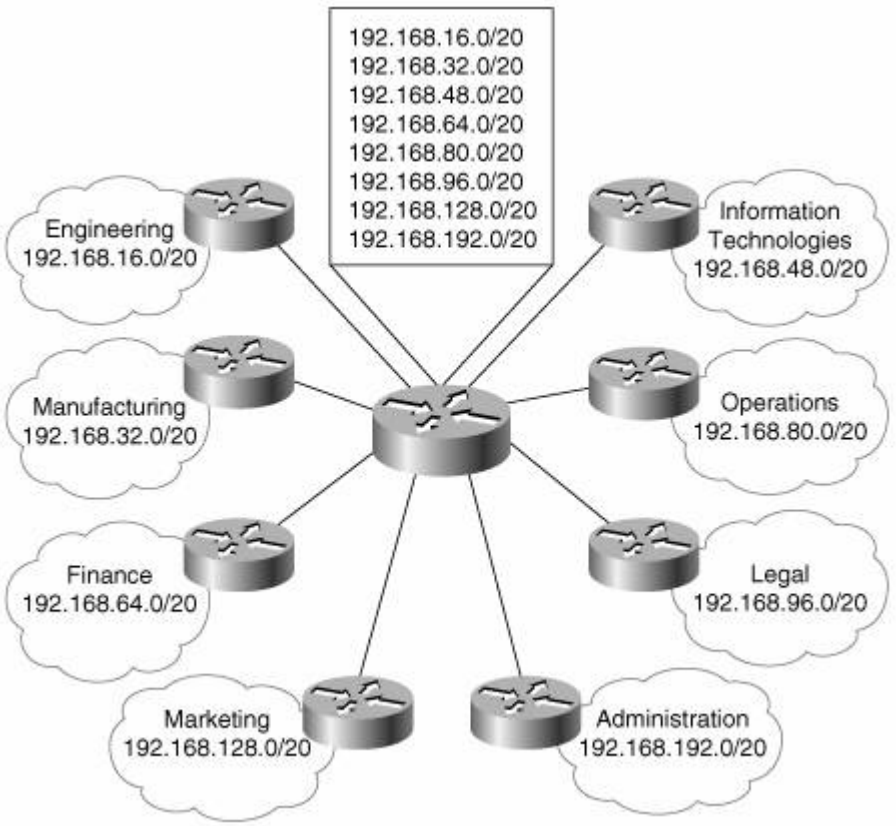
Figure 7-29. At Treetop Aviation, several departments within a larger division are aggregating addresses. In turn, the entire division can be advertised with a single aggregate address (192.168.16.0/20).




The aggregate addresses of the engines, electrical, and hydraulics departments are themselves aggregated into a single address, 192.168.16.0/21. That address and the aggregate address of the airframe department are aggregated into the single address 192.168.16.0/20, which represents the entire engineering division.

Other divisions may be similarly represented. For example, if Treetop Aviation had a total of eight divisions and if those divisions were all addressed similarly to the engineering division, the backbone router at the top of the hierarchy might have only eight routes (Figure 7-30).

Figure 7-30. Although there are 128 major network addresses and possibly more than 32,000 hosts in this network, the backbone router has only eight aggregate addresses in its route table.




The hierarchical design is continued within each department of each division by subnetting the individual network addresses. VLSM may be used to further divide the subnets. The routing protocols will automatically summarize the subnets at network boundaries, as discussed in previous chapters.

Address aggregation also allows both address conservation and address hierarchies in the Internet. With the exponential growth of the Internet, two increasing concerns have been the depletion of available IP addresses (particularly class B addresses) and the huge databases needed to store the Internet routing information.

A solution to this problem is *Classless Interdomain Routing* (CIDR).[14] Under CIDR, aggregates of class C addresses are allocated by the IANA to the various worldwide address assignment authorities such as Asia Pacific Network Information Centre (APNIC) in Asia, American Registry for Internet Numbers (ARIN) in North America, and Réseaux IP Européens (RIPE) in Europe. The aggregates are organized geographically, as shown in Table 7-6.

[14] V. Fuller, T. Li, J. I. Yu, and K. Varadhan. "Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy," RFC 1519. September 1993.

## Table 7-6. CIDR address allocations by geographic region.

| Region | Address Range |
|---|---|
| Multiregional | 192.0.0.0193.255.255.255 |
| Europe | 194.0.0.0195.255.255.255 |
| Others | 196.0.0.0197.255.255.255 |

| Region | Address Range |
|---|---|
| North America | 198.0.0.0199.255.255.255 |
| Central/South America | 200.0.0.0201.255.255.255 |
| Pacific Rim | 202.0.0.0203.255.255.255 |
| Others | 204.0.0.0205.255.255.255 |
| Others | 206.0.0.0207.255.255.255 |

The address assignment authorities in turn divide their portions among the regional Internet Service Providers (ISPs). When an organization applies for an IP address and requires addressing for fewer than 32 subnets and 4096 hosts, it will be given a contiguous group of class C addresses called a *CIDR block*.

In this way, the Internet routers of individual organizations might advertise a single summary address to their ISP. The ISP, in turn, aggregates all of its addresses, and ideally all ISPs in a region of the world may be summarized by the addresses of Table 7-6. Knowing that the current size of the Internet routing table is approaching 200,000 routes, it is obvious that CIDR has not been adhered to as well as intended. Nonetheless, the concept is a good one. Similar efforts are under way to constrain IPv6 address assignments geographically; we can only hope that the efforts are more successful than they were with IPv4.

PREV NEXT

# Configuring EIGRP

The case studies in this section demonstrate a basic EIGRP configuration and then examine summarization techniques and interoperability with IGRP.

## Case Study: A Basic EIGRP Configuration

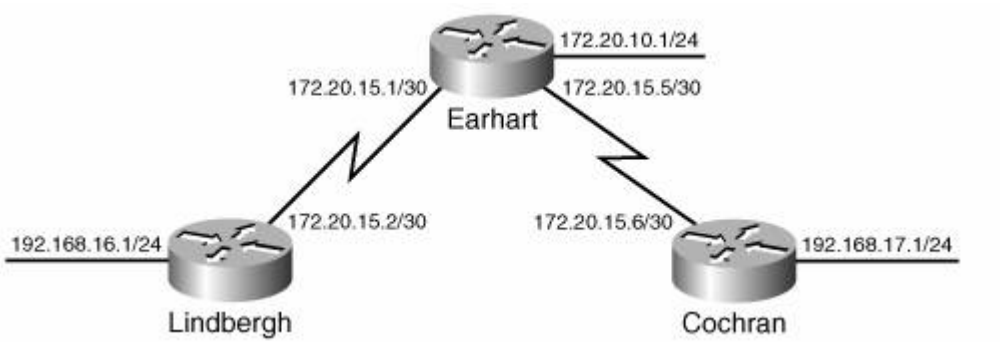
EIGRP requires only two steps to begin the routing process:

Step 1. Enable EIGRP with the command `router eigrp` *process-id*.

Step 2. Specify each major network on which to run EIGRP with the `network` command.

The process ID may be any number between 1 and 65535 (0 is not allowed), and it may be arbitrarily chosen by the network administrator, if it is the same for all EIGRP processes in all routers that must share information. Alternatively, the number might be a publicly assigned autonomous system number. Figure 7-31 shows a simple network; the configurations for the three routers are displayed in Example 7-14, Example 7-15, and Example 7-16.

Figure 7-31. Unlike IGRP, EIGRP will support the VLSM requirements of this network.




Example 7-14. Earhart.

```
router eigrp 15
 network 172.20.0.0
```

Example 7-15. Cochran.

```
router eigrp 15
 network 172.20.0.0
 network 192.167.17.0
```

### Example 7-16. Lindbergh.

```
router eigrp 15
 network 172.20.0.0
 network 192.167.16.0
```

Earhart's route table is shown in Example 7-17. The table shows that the default EIGRP administrative distance is 90 and that network 172.20.0.0 is variably subnetted.

### Example 7-17. Earhart's route table.

```
Earhart#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route
Gateway of last resort is not set
     172.20.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       172.20.10.0/24 is directly connected, Ethernet0
C       172.20.15.4/30 is directly connected, Serial0.1
C       172.20.15.0/30 is directly connected, Serial0.2
D    192.168.17.0/24 [90/2195456] via 172.20.15.6, 00:00:01, Serial0.1
D    192.168.16.0/24 [90/2195456] via 172.20.15.2, 00:00:01, Serial0.2
Earhart#
```

The network of Figure 7-31 uses default metrics, unlike the earlier examples in this chapter, so a review of the EIGRP metric calculation in a more realistic scenario might be useful.

Tracing the route from Earhart to network 192.168.16.0, the path traverses a serial interface and an Ethernet interface, each with default metric values. The minimum bandwidth of the route will be that of the serial interface,[15] and the delay will be the sum of the two interface delays. Referring back to Table 7-1:

[15] Remember that the default bandwidth of a serial interface is 1544K.

$BW_{EIGRP(min)} = 256 \times 6476 = 1657856$

$DLY_{EIGRP(sum)} = 256 \times (2000 + 100) = 537600$

Therefore,

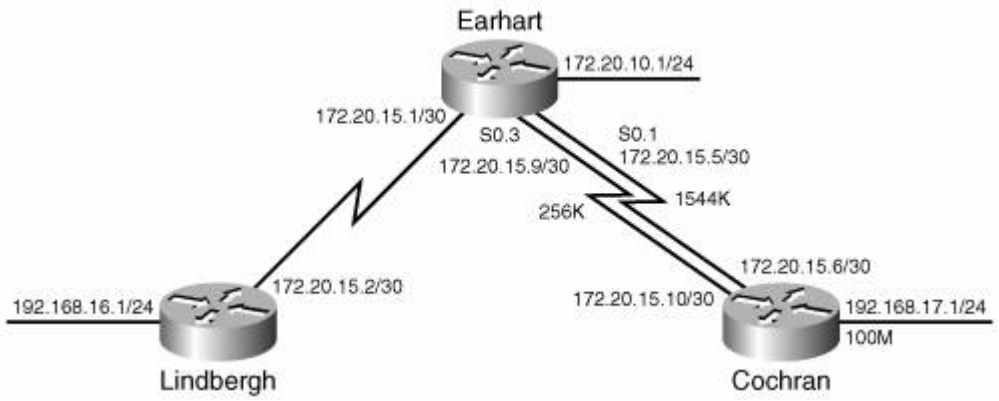
Metric = 1657856 + 537600 = 2195456

## Case Study: Unequal-Cost Load Balancing

Given up to 16 parallel routes of equal cost,[16] EIGRP will do equal-cost load balancing under the same CEF/fast/process switching constraints as RIP. Unlike RIP, EIGRP can also perform unequal-cost load balancing. An additional serial link has been added between Earhart and Cochran in Figure 7-32, with a configured bandwidth of 256K. The goal is to have Earhart perform unequal-cost load balancing across these two linksspreading the traffic load inversely proportional to the metrics of the link.

[16] The default is four paths. See the case study on setting maximum paths for further details.

Figure 7-32. EIGRP can be configured to perform unequal-cost load balancing across links such as the two between Earhart and Cochran.




Examining the route from Earhart's S0.1 interface to network 192.168.17.0, the minimum bandwidth is 1544K (assuming Cochran's Ethernet interface is using the default 100000K bandwidth for Fast Ethernet). Referring to Table 7-1, $DLY_{EIGRP(sum)}$ for the serial interface and the Fast Ethernet interface is 256 x ( 2000 + 10 ) = 514560. $BW_{EIGRP(min)}$ is 256 x ($10^7$/1544) = 1657856, so the composite metric of the route is 514560 + 1657856) = 2172416.

The minimum bandwidth on the route via Earhart's S0.3 to 192.168.17.0 is 256K; $DLY_{EIGRP(sum)}$ is the same as on the first route. Therefore, the composite metric for this route is 256 x ($10^7$/256) + 514560 = 10514432. Without further configuration, EIGRP will simply select the path with the lowest metric cost. Example 7-18 shows that Earhart is using only the path via Serial 0.1, with a metric of 2172416.

Example 7-18. Earhart is using only the lowest-cost link to network 192.168.17.0. Additional configuration is needed to enable unequal-cost load balancing.

```
Earhart#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E  EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o  ODR
       P - periodic downloaded static route
Gateway of last resort is not set
     172.20.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       172.20.10.0/27 is directly connected, Ethernet0
C       172.20.15.4/30 is directly connected, Serial0.1
C       172.20.15.0/30 is directly connected, Serial0.2
C       172.20.15.8/30 is directly connected, Serial0.3
D    192.168.17.0/24 [90/2172416] via 172.20.15.6, 00:00:09, Serial0.1
D    192.168.16.0/24 [90/2195456] via 172.20.15.2, 00:00:09, Serial0.2
Earhart#
```

The variance command is used to determine which routes are feasible for unequal-cost load sharing. Variance defines a multiplier by which a metric may differ, or vary, from the metric of

the lowest-cost route. Any route whose metric exceeds the metric of the lowest-cost route, multiplied by the variance, will not be considered a feasible route.

The default variance is one, meaning that the metrics of multiple routes must be equal, to load balance. Variance must be specified in whole numbers.

The metric of Earhart's route through S0.3 is 10514432/2172416 = 4.8 times larger than the metric of the S0.1 route. So to conduct unequal-cost load balancing over both links, the variance at Earhart should be five. The EIGRP configuration is in Example 7-19.

### Example 7-19. Earhart's configuration uses a variance of five to perform unequal-cost load sharing using EIGRP.

```
router eigrp 15
 network 172.20.0.0
 variance 5
```

After specifying a variance of five at Earhart, its route table will include the second, higher-cost route (Example 7-20). The following three conditions must be met for a route to be included in unequal-cost load sharing:

- The maximum-paths limit must not be exceeded as a result of adding the route to a load-sharing "group."
- The next-hop router must be metrically closer to the destination. That is, its metric for the route must be smaller than the local router's metric. A next-hop router, being closer to the destination, is often referred to as the *downstream* router.
- The metric of the lowest-cost route, when multiplied by the variance, must be greater than the metric of the route to be added.

### Example 7-20. The composite metric of the second path to 192.168.17.0 from Earhart is 10514432, or 4.8 times the metric of the lowest-cost route. EIGRP will enter the second path into the route table if the variance is set to at least five.

```
Earhart(config)#router eigrp 15
Earhart(config-router)#variance 5
Earhart(config-router)#^Z
Earhart#clear ip route *
Earhart#show ip route
Gateway of last resort is not set
     172.20.0.0/16 is variably subnetted, 4 subnets, 2 masks
C       172.20.10.0/24 is directly connected, Ethernet0
C       172.20.15.4/30 is directly connected, Serial0.1
C       172.20.15.0/30 is directly connected, Serial0.2
C       172.20.15.8/30 is directly connected, Serial0.3
D    192.168.17.0/24 [90/2172416] via 172.20.15.6, 00:00:02, Serial0.1
                     [90/10514432] via 172.20.15.10, 00:00:02, Serial0.3
D    192.168.16.0/24 [90/2195456] via 172.20.15.2, 00:00:02, Serial0.2
Earhart#
```

The rules concerning per destination versus per packet load sharing, discussed in Chapter 3, "Static Routing," apply here as well. Load sharing is per destination if the packet is fast switched or CEF switched using the default CEF configuration, and per packet if process switching is used

or if the CEF configuration was modified. Example 7-21 shows a debug output resulting from 20 ping packets being sent through Earhart; CEF and fast switching have been turned off with `no ip cef` and `no ip route-cache`, and the router is performing unequal-cost, per packet load balancing. For every five packets sent over the 1544K link (to next hop 172.20.15.6), one packet is sent over the 256K link (to next hop 172.20.15.10). This corresponds to the approximately five-to-one variance of the metrics of these two paths.

### Example 7-21. Per packet load sharing is being performed, with one packet being sent over the high-cost link for every five packets sent over the low-cost link.

```
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.3), g=172.20.15.10,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.3), g=172.20.15.10,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.1), g=172.20.15.6,
    len 100, forward
IP: s=172.20.15.2 (Serial 2), d=192.168.17.1 (Serial0.3), g=172.20.15.10,
    len 100, forward
Earhart#
```

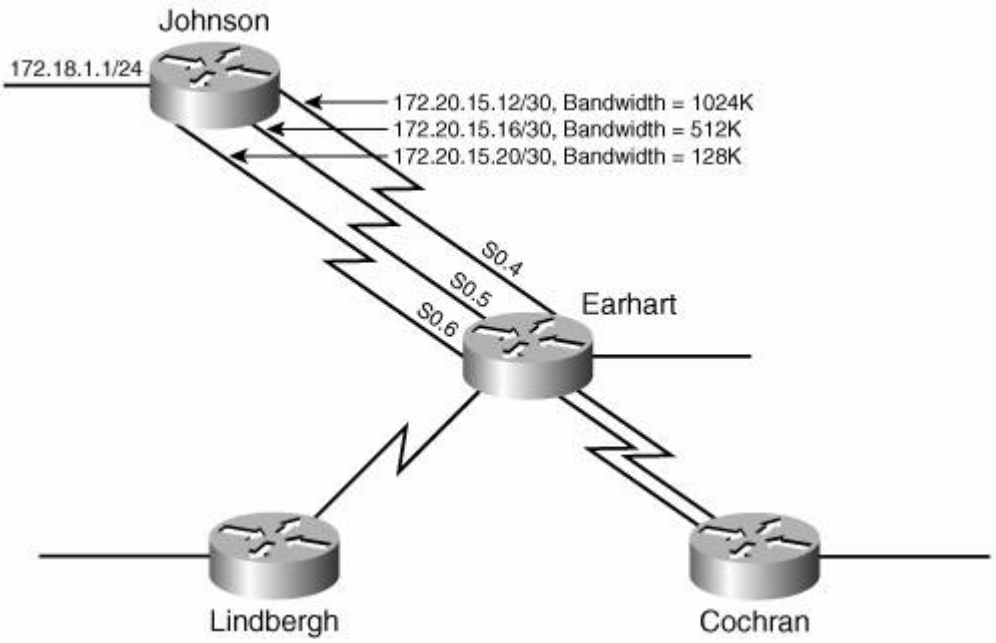
If variance is set at one, EIGRP enters only the lowest-cost route to a destination into the route table. In some situations, howeverfor example, to decrease reconvergence time or aid in troubleshootingall feasible routes should be entered into the table, even though no load balancing should occur. All packets should use the lowest-cost route and switch to the next-best path only if the primary fails. There is an implicit default command (that is, it exists, but is not observed in the configuration file) of `traffic-share balanced`. To configure the router to only use the minimum-cost path even when multiple paths are shown in the route table, change this default to `traffic-share min`. If there are multiple minimum-cost paths and `traffic-share min` is configured, EIGRP will perform equal-cost load balancing.

## Case Study: Setting Maximum Paths

The maximum number of routes over which EIGRP can load balance is set with the maximum-paths *paths* command. *paths* may be any number from 1 to 16 in IOS 12.3(2)T and later 12.3(T) releases and any number from 1 to 6 in earlier versions. The default for all versions is 4.

Figure 7-33 shows three parallel paths of varying costs from Earhart to address 172.18.0.0. The network administrator wants to load balance over a maximum of only two of these routes while ensuring that if either of these paths should fail, the third route will replace it.

Figure 7-33. The maximum-paths and variance commands can be used together to configure load balancing over only two of the three links between Earhart and Johnson. If either link fails, the third will take its place.




The metrics from Earhart are

Via S0.4: 256 x (9765 + (2000 + 10)) = 3014400

Via S0.5: 256 x (19531 + (2000 + 10)) = 5514496

Via S0.6: 256 x (78125 + (2000 + 10)) = 20514560

The metric of the S0.6 route is 6.8 times as large as the lowest-cost metric, so the variance is seven.

Earhart's EIGRP configuration is displayed in Example 7-22.

Example 7-22. Earhart's configuration uses the variance command and the maximum-paths command to provide unequal-cost load sharing over a specified maximum number of paths.

```
router eigrp 15
 variance 7
 network 172.20.0.0
 maximum-paths 2
```

The variance command ensures that any of the three routes to 172.18.0.0 is feasible; the maximum-paths command limits the load-sharing group to only the two best routes. The results of this configuration can be seen in Example 7-23. The first route table shows that Earhart was load balancing over the two links with the lowest of the three metrics, S0.4 and S0.5. After a failure of the S0.4 link, the second route table shows that the router is now load balancing over the S0.5 and S0.6 links. In each instance, the router will load balance inversely proportional to the metrics of the two paths.

Example 7-23. The route table for Earhart, before and after the failure of one of three links, shows the results of using the variance and maximum-paths commands to configure load sharing to 172.18.0.0.

```
Earhart#debug eigrp neighbor
EIGRP Neighbors debugging is on
Earhart#show ip route
Gateway of last resort is not set

D    172.18.0.0/16 [90/5514496] via 172.20.15.18, 00:00:16, Serial0.5
                   [90/3014400] via 172.20.15.14, 00:00:16, Serial0.4
     172.20.0.0/16 is variably subnetted, 7 subnets, 2 masks
C       172.20.15.20/30 is directly connected, Serial0.6
C       172.20.15.16/30 is directly connected, Serial0.5
C       172.20.10.0/24 is directly connected, Ethernet0
C       172.20.15.4/30 is directly connected, Serial0.1
C       172.20.15.0/30 is directly connected, Serial0.2
C       172.20.15.12/30 is directly connected, Serial0.4
C       172.20.15.8/30 is directly connected, Serial0.3
D    192.168.17.0/24 [90/2195456] via 172.20.15.6, 00:00:18, Serial0.1
                     [90/10537472] via 172.20.15.10, 00:00:18, Serial0.3
D    192.168.16.0/24 [90/2195456] via 172.20.15.2, 00:00:19, Serial0.2
Earhart#
1w6d: EIGRP: Holdtime expired
1w6d: EIGRP: Neighbor 172.20.15.14 went down on Serial0.4
Earhart#show ip route
Gateway of last resort is not set
D    172.18.0.0/16 [90/5514496] via 172.20.15.18, 00:00:09, Serial0.5
                   [90/20514560] via 172.20.15.22, 00:00:09, Serial0.6
     172.20.0.0/16 is variably subnetted, 7 subnets, 2 masks
C       172.20.15.20/30 is directly connected, Serial0.6
C       172.20.15.16/30 is directly connected, Serial0.5
C       172.20.10.0/24 is directly connected, Ethernet0
C       172.20.15.4/30 is directly connected, Serial0.1
C       172.20.15.0/30 is directly connected, Serial0.2
C       172.20.15.12/30 is directly connected, Serial0.4
C       172.20.15.8/30 is directly connected, Serial0.3
D    192.168.17.0/24 [90/2195456] via 172.20.15.6, 00:00:26, Serial0.1
                     [90/10537472] via 172.20.15.10, 00:00:26, Serial0.3
D    192.168.16.0/24 [90/2195456] via 172.20.15.2, 00:00:26, Serial0.2
Earhart#
```

Care should be taken when configuring multiple parallel paths in a network. Too many parallel paths can increase the amount of time it takes for EIGRP to converge when a link fails, because

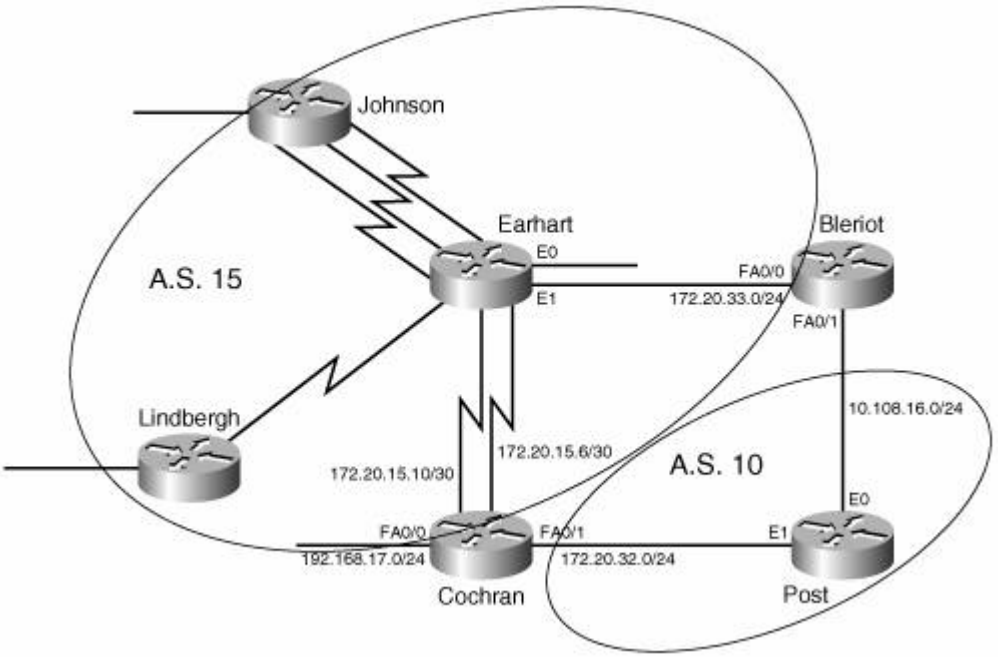
the number of neighboring routers has increased, therefore increasing the querying scope.

## Case Study: Multiple EIGRP Processes

Two new routers, Bleriot and Post, have been added to the network. A decision has been made to create two EIGRP process domains in the network with no communications between the two. Figure 7-34 shows the two autonomous systems and the related links for each.

Figure 7-34. The routers Bleriot and Cochran will each run multiple EIGRP processes to facilitate the creation of separate autonomous systems (AS 10 and AS 15) within this IGP.

[View full size image]




The configurations for Post, Johnson, Lindbergh, and Earhart are straightforward: Johnson, Earhart, and Lindbergh will run EIGRP 15, and Post will run EIGRP 10. At Bleriot, the configuration will be as displayed in Example 7-24.

Example 7-24. Bleriot's configuration for EIGRP 15 and EIGRP 10.

```
router eigrp 15
 network 172.20.0.0
!
router eigrp 10
 network 10.0.0.0
```

Each process will run only on the interfaces of the networks specified. At Cochran, all interfaces other then FA0/0 belong to network 172.20.0.0 (see Example 7-25).

Using the passive-interface command prevents EIGRP Hellos from being sent on data links where they don't belong. Note that because Cochran's interfaces are in network 172.20.0.0, the passive-interface command is used to restrict unnecessary routing protocol traffic. For EIGRP,

this command blocks unnecessary Hellos. No adjacencies will be formed; therefore, no other EIGRP traffic will be sent.

### Example 7-25. Cochran's EIGRP 15 and EIGRP 10 configuration requires passive interfaces because the same major network number is used on both of Cochran's connected interfaces.

```
router eigrp 15
 passive-interface Fastethernet0/1
 network 172.20.0.0
!
router eigrp 10
 passive-interface Serial0/0.1
 passive-interface Serial 0/0.2
 network 172.20.0.0
```

Bleriot's neighbor table (Example 7-26) shows that there is one neighbor for EIGRP process 10, 10.108.16.2, and one neighbor for process 15, 172.20.33.1.

### Example 7-26. The neighbors associated with each of the multiple EIGRP processes are displayed using the *show ip eigrp neighbor* command.

```
Bleriot#show ip eigrp neighbors
IP-EIGRP neighbors for process 15
H   Address                 Interface   Hold Uptime    SRTT   RTO   Q   Seq
                                        (sec)          (ms)        Cnt  Num
0   172.20.33.1             Fa0/0         11 00:31:19    34   204   0   44
IP-EIGRP neighbors for process 10
H   Address                 Interface   Hold Uptime    SRTT   RTO   Q   Seq
                                        (sec)          (ms)        Cnt  Num
0   10.108.16.2             Fa0/1         10 00:19:21   289  1734   0   6
Bleriot#
```

In lieu of passive interfaces, the network statement for EIGRP can be configured with wildcard bits. The wildcard bits specify which bits of the address are to be used when identifying interfaces to include in the EIGRP process.

Cochran's configuration is modified to that shown in Example 7-27.

### Example 7-27. Cochran's EIGRP 15 and EIGRP 10 configuration uses wildcard bits with the network to narrow down the interfaces that will run EIGRP for a given process.

```
router eigrp 15
 network 172.20.15.0 0.0.0.255
!
router eigrp 10
 network 172.20.32.0 0.0.0.255
```

Cochran's configuration in Example 7-27 specifies that interfaces with addresses with the first three octets equal to 172.20.15 run EIGRP process 15, and interfaces with the first three octets equal to 172.20.32 run EIGRP process 10.

## Case Study: Disabling Automatic Summarization

By default, EIGRP summarizes at network boundaries as do the protocols covered in previous chapters. Unlike those protocols, however, EIGRP's automatic summarization can be disabled.

Look at Bleriot's route table in Example 7-28. Notice that the entry for address 172.20.32.0 is known via Fast Ethernet 0/0, even though that path goes through one Fast Ethernet link and one serial link, from Bleriot, through Earhart to Cochran. The path via Post, attached to Fast Ethernet 0/1, is only one Fast Ethernet hop away. Look more closely at the route table, and you'll see the only entry for an address other then 10.108.16.0 known via Fast Ethernet 0/1 is 172.20.0.0/16. Post has summarized the 172.20.32.0 address before it advertises it to Bleriot over the 10.0.0.0 network boundary. To enable Bleriot to forward traffic to 172.20.32.0 via Post, disable automatic summarization on Post using the command `no auto-summary`.

### Example 7-28. EIGRP automatic summarization can cause, in some cases, sub-optimal route choices.

```
Bleriot#show ip route
Gateway of last resort is not set
D    172.18.0.0/16 [90/3016960] via 172.20.33.1, 00:26:42, FastEthernet0/0
     172.20.0.0/16 is variably subnetted, 10 subnets, 4 masks
D       172.20.32.0/24 [90/2174976] via 172.20.33.1, 00:14:46, FastEthernet0/0
C       172.20.33.0/24 is directly connected, FastEthernet0/0
D       172.20.15.20/30
           [90/20514560] via 172.20.33.1, 00:20:28, FastEthernet0/0
D       172.20.15.16/30
           [90/5514496] via 172.20.33.1, 00:20:28, FastEthernet0/0
D       172.20.10.0/27 [90/284160] via 172.20.33.1, 00:43:34, FastEthernet0/0
D       172.20.15.4/30 [90/2172416] via 172.20.33.1, 00:26:59, FastEthernet0/0
D       172.20.15.0/30 [90/2172416] via 172.20.33.1, 00:27:18, FastEthernet0/0
D       172.20.0.0/16 [90/284160] via 10.108.16.2, 00:14:50, FastEthernet0/1
D       172.20.15.12/30
           [90/3014400] via 172.20.33.1, 00:26:49, FastEthernet0/0
D       172.20.15.8/30
           [90/10514432] via 172.20.33.1, 00:20:47, FastEthernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.108.16.0 is directly connected, FastEthernet0/1
D    192.168.16.0/24 [90/2198017] via 172.20.33.1, 00:27:33, FastEthernet0/0
D    192.168.17.0/24 [90/2174976] via 172.20.33.1, 00:00:38, FastEthernet0/0
Bleriot#
```

Post's configuration is displayed in Example 7-29.

### Example 7-29. Post's configuration disables automatic summarization for EIGRP.

```
router eigrp 10
 network 10.0.0.0
 network 172.20.0.0
 no auto-summary
```

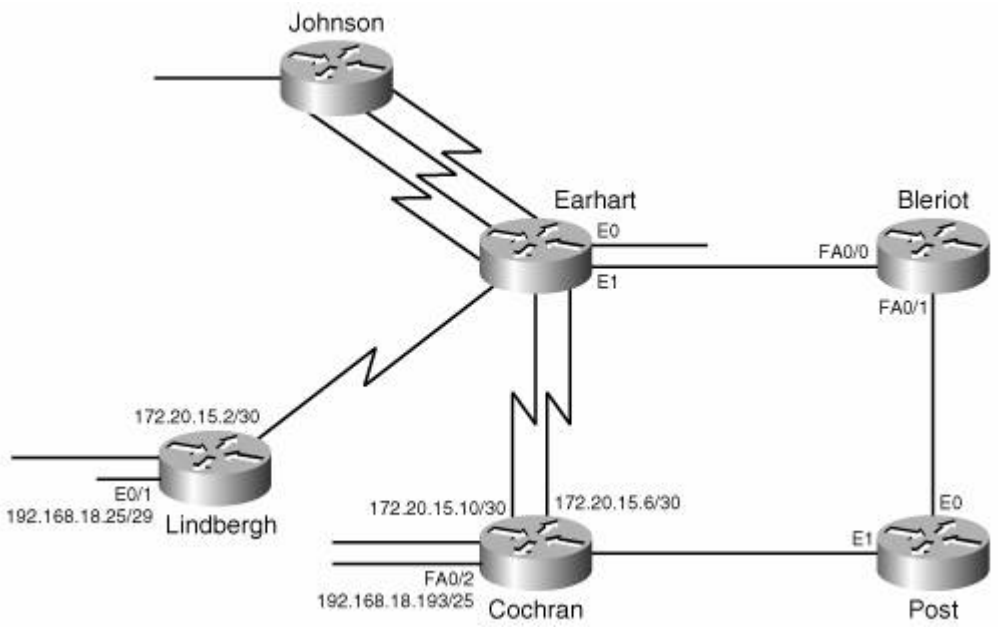
Bleriot's new route table is shown in Example 7-30.

Example 7-30. After disabling EIGRP automatic summarization, subnets of distant addresses can be seen in the route tables.

```
Bleriot#show ip route
Gateway of last resort is not set
D    172.18.0.0/16 [90/3016960] via 172.20.33.1, 00:35:27, FastEthernet0/0
     172.20.0.0/16 is variably subnetted, 9 subnets, 3 masks
D       172.20.32.0/24 [90/284160] via 10.108.16.2, 00:00:55, FastEthernet0/1
C       172.20.33.0/24 is directly connected, FastEthernet0/0
D       172.20.15.20/30
           [90/20514560] via 172.20.33.1, 00:29:13, FastEthernet0/0
D       172.20.15.16/30
           [90/5514496] via 172.20.33.1, 00:29:13, FastEthernet0/0
D       172.20.10.0/27 [90/284160] via 172.20.33.1, 00:52:19, FastEthernet0/0
D       172.20.15.4/30 [90/2172416] via 172.20.33.1, 00:35:44, FastEthernet0/0
D       172.20.15.0/30 [90/2172416] via 172.20.33.1, 00:36:03, FastEthernet0/0
D       172.20.15.12/30
           [90/3014400] via 172.20.33.1, 00:35:34, FastEthernet0/0
D       172.20.15.8/30
           [90/10514432] via 172.20.33.1, 00:29:20, FastEthernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.108.16.0 is directly connected, FastEthernet0/1
D    192.168.16.0/24 [90/2198016] via 172.20.33.1, 00:36:06, FastEthernet0/0
D    192.168.17.0/24 [90/2174976] via 172.20.33.1, 00:00:38, FastEthernet0/0
Bleriot#
```

Figure 7-35 shows another situation in which disabling summarization is useful.

Figure 7-35. Disabling automatic summarization at Cochran and Lindbergh prevents ambiguous routing to network 192.168.18.0.




New Ethernet links have been added to routers Cochran and Lindbergh, and their addresses create a discontiguous subnet. The default behavior of both routers is to see themselves as border routers between major networks 172.20.0.0 and 192.168.18.0. As a result, Earhart will

receive summary advertisements to 192.168.18.0 on its serial interfaces to both Lindbergh and Cochran. The result is an ambiguous routing situation in which Earhart records two equal-cost paths to 192.168.18.0; a packet destined for one of the subnets might or might not be routed to the correct link.

After disabling automatic summarization, Lindbergh's configuration will be as in Example 7-31.

Example 7-31. Lindbergh's configuration disables automatic summarization.

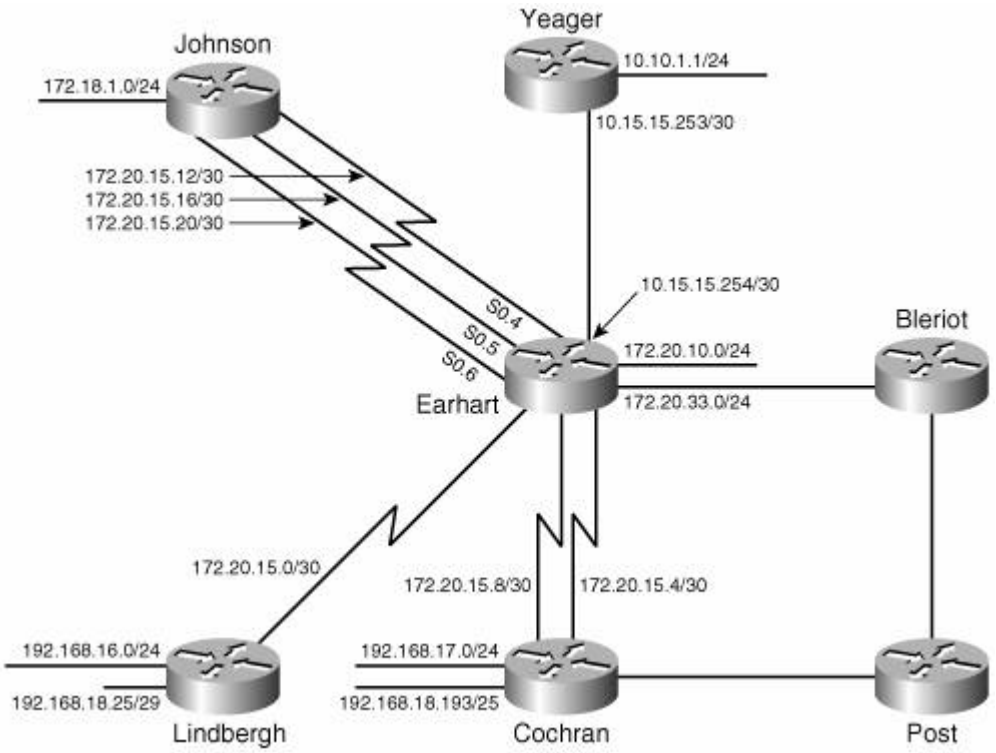
```
router eigrp 15
 network 172.20.0.0
 network 192.168.16.0
 network 192.168.18.0
 no auto-summary
```

By turning off summarization at Lindbergh and Cochran, the individual subnets 192.168.18.24/29 and 192.168.18.128/25 will be advertised into network 172.20.0.0, eliminating the ambiguities at Earhart.

## Case Study: Stub Routing

Recall the discussion of DUAL earlier in this chapter. When an entry in a router's EIGRP topology table changes for the worse (either the metric increases, or the successor is no longer accessible), if there is no feasible successor for the address, the entry goes into Active state, and the router sends query packets to all its neighbors. If Earhart's link to Yeager, in Figure 7-36, goes down, Earhart sends queries to all its neighbors, including Johnson and Lindbergh, to find out if any neighbors have a path to Yeager. Earhart cannot modify its active entries in the topology table until it hears responses from all its queries regarding that entry. If a problem develops on the link to Lindbergh before Earhart has received a response to the query it sent about Yeager's addresses, Yeager's addresses will remain Active, even if the link between Earhart and Yeager comes back up.

Figure 7-36. Yeager is added to the network with a single link to Earhart.




Johnson and Lindbergh, in Figure 7-36, do not have back-door routes to any other site in the network. They are spoke routers in a hub-and-spoke design. The routers are not used to provide transit paths to any addresses in the network. When Lindbergh or Johnson need to forward a packet to an address that is not local to its site, the packet is forwarded to Earhart. Lindbergh knows of one path to 172.20.10.0, for instance, and that path is via Earhart. There is no need to send Johnson queries about addresses in other locations of the network and risk causing network instabilities. Johnson and Lindbergh can be configured with stub routing.

A router that has EIGRP Stub neighbors will not send queries to the stubs, thereby eliminating the chance that a stub-configured remote site will cause stuck in active conditions, and routing instabilities in other parts of the network.

Johnson is configured as an EIGRP stub router. Johnson's stub router configuration is displayed in Example 7-32.

## Example 7-32. Johnson's EIGRP stub router configuration.

```
router eigrp 15
 eigrp stub
```

No configuration changes are required on Earhart, the hub router.

The command `eigrp stub` causes Johnson to send updates containing its connected and summary routes only. Johnson can be configured to include any combination of connected routes, summary routes, static routes, or routes that have been redistributed into EIGRP, with the command:

```
eigrp stub {connected | redistributed | static | summary | receive-only}
```

Johnson can also be configured to not send any route information in updates, with the `receive-only` option. With the `receive-only` option, the remote router will not include any addresses in

an update. Addresses connected to the Johnson router would have to be advertised to the rest of the network in some other way to ensure that traffic can reach the site, perhaps with static routes configured on Earhart.

To verify a neighbor is configured as a stub router, use the command `show ip eigrp neighbor detail` on the hub router, as shown in Example 7-33. Earhart's output shows that Johnson is configured as a stub.

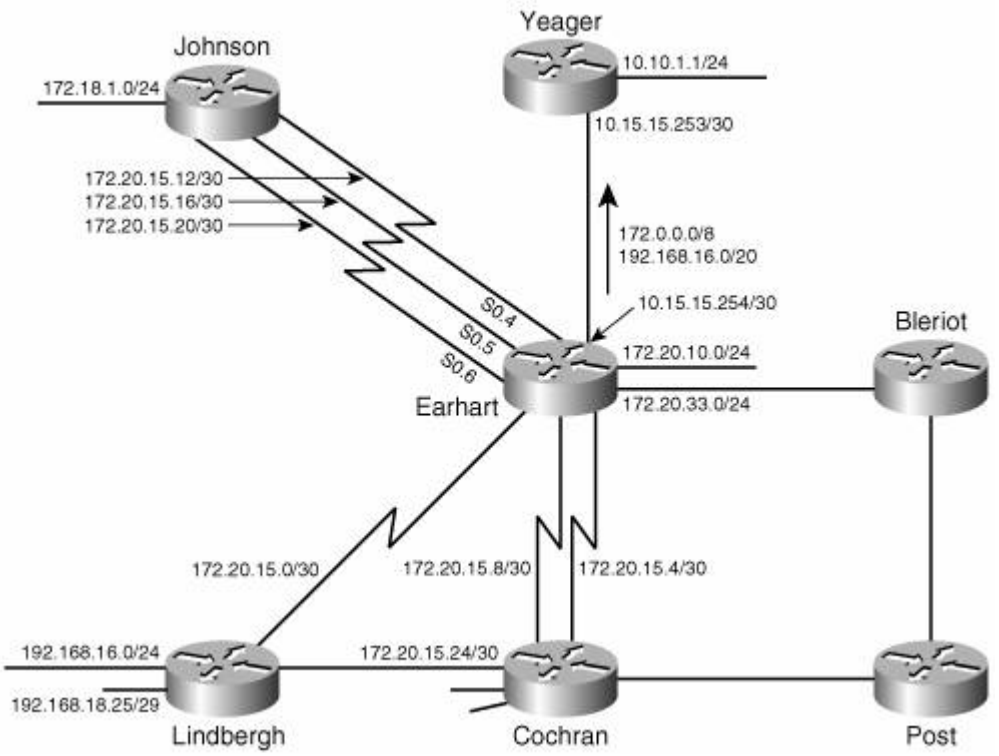
### Example 7-33. *show ip eigrp neighbor detail* displays which neighbor routers are configured as EIGRP stubs.

```
Earhart#show ip eigrp neighbor detail
IP-EIGRP neighbors for process 15
H   Address                 Interface   Hold Uptime    SRTT   RTO   Q   Seq Type
                                        (sec)          (ms)        Cnt  Num
7   172.20.33.2             Et1           11 00:00:10    12   200   0   13
   Version 12.1/1.2, Retrans: 0, Retries: 0
6   10.15.15.253            Et2           11 00:00:10    12   200   0   6
   Version 12.3/1.2, Retrans: 1, Retries: 0
5   172.20.15.22            Se0.6         11 00:00:13   298  1788   0   73
   Version 12.3/1.2, Retrans: 0, Retries: 0
   Stub Peer Advertising ( CONNECTED SUMMARY ) Routes
4   172.20.15.14            Se0.4         11 00:00:13   927  5000   0   81
   Version 12.3/1.2, Retrans: 0, Retries: 0
   Stub Peer Advertising ( CONNECTED SUMMARY ) Routes
3   172.20.15.18            Se0.5         10 00:00:13   817  4902   0   80
   Version 12.3/1.2, Retrans: 0, Retries: 0
   Stub Peer Advertising ( CONNECTED SUMMARY ) Routes
0   172.20.15.2             Se0.2         11 00:15:48   274  1644   0   13
   Version 12.3/1.2, Retrans: 0, Retries: 0
2   172.20.15.10            Se0.3         13 00:45:40    72   570   0   59
   Version 12.3/1.2, Retrans: 0, Retries: 0
1   172.20.15.6             Se0.1         11 00:46:01    61   366   0   57
   Version 12.3/1.2, Retrans: 0, Retries: 0
Earhart#
```

Earhart has three stub neighbors. They are connected to the three links to Johnson: 172.20.15.22, 172.20.15.14, and 172.20.15.17.

Now, suppose a new link is added between Lindbergh and Cochran to add redundancy for traffic connected to Lindbergh's LAN traveling into the core of the network, as shown in Figure 7-37. Before Lindbergh is configured as a stub, when the links between Cochran and Earhart fail, Cochran will send queries to Lindbergh regarding alternate paths to the addresses in its topology table, 172.20.10.0 for instance. Lindbergh responds with a positive reply, because Lindbergh has an entry in its topology table for 172.20.10.0 via Earhart. Traffic from Cochran to 172.20.10.1 travels via Lindbergh.

### Figure 7-37. A new link is added between Lindbergh and Cochran to add redundancy for traffic connected to Lindbergh's LAN traveling into the core of the network.




Although this is an alternate path, forwarding traffic through a remote site is not always desirable. In this case, Lindbergh's links are there to allow redundancy from Lindbergh's addresses to core addresses, not to enable Lindbergh to act as a transit router. The bandwidth might not be sufficient to provide transit routing. Stub routing easily solves this problem. As a stub, no queries are sent to Lindbergh, so Lindbergh will not make itself available to Cochran as an alternate path. Furthermore, Lindbergh only sends updates containing connected, summary, static, or redistributed routes, not remote routes, such as 172.20.10.0.

Example 7-34 shows Cochran's EIGRP neighbors when Lindbergh (connected to Serial 0/0.4) is not configured as a stub router. Cochran's two links to Earhart are brought down. Cochran's subsequent topology table (Example 7-35) shows all addresses are accessible via Lindbergh (Serial 0/0.4).

## Example 7-34. Cochran's EIGRP neighbor table shows its neighbors. Lindbergh is not a stub.

```
Cochran#show ip eigrp neighbor detail
IP-EIGRP neighbors for process 15
H   Address                 Interface       Hold Uptime    SRTT   RTO   Q   Seq
                                            (sec)          (ms)        Cnt  Num
2   172.20.15.26            Se0/0.4           10 00:00:18  1152  5000   0   34
   Version 12.3/1.2, Retrans: 0, Retries: 0
1   172.20.15.9             Se0/0.2           14 00:41:58   104   624   0   205
   Version 12.1/1.2, Retrans: 1, Retries: 0
0   172.20.15.5             Se0/0.1           11 00:49:12    99   594   0   206
   Version 12.1/1.2, Retrans: 2, Retries: 0
IP-EIGRP neighbors for process 10
H   Address                 Interface       Hold Uptime    SRTT   RTO   Q   Seq
                                            (sec)          (ms)        Cnt  Num
0   172.20.32.2             Fa0/1             13 00:42:01    60   360   0   14
   Version 12.1/1.2, Retrans: 2, Retries: 0
Cochran#
```

## Example 7-35. After failure of the primary links between Cochran and Earhart, all addresses are accessible via the dual connected spoke router, Lindbergh.

```
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 15: Neighbor 172.20.15.5 (Serial0/0.1) is down:
interface down
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 15: Neighbor 172.20.15.9 (Serial0/0.2) is down:
interface down
Cochran#show ip eigrp topology
IP-EIGRP Topology Table for AS(15)/ID(192.168.17.1)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.0.0.0/8, 1 successors, FD is 2707456
        via 172.20.15.26 (2707456/2195456), Serial0/0.4
P 192.168.18.24/29, 1 successors, FD is 2195456
        via 172.20.15.26 (2195456/281600), Serial0/0.4
P 192.168.16.0/24, 1 successors, FD is 2195456
        via 172.20.15.26 (2195456/281600), Serial0/0.4
P 192.168.17.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
P 172.20.32.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/1
P 172.20.33.0/24, 1 successors, FD is 2707456
        via 172.20.15.26 (2707456/2195456), Serial0/0.4
P 172.20.15.20/30, 1 successors, FD is 21536000
        via 172.20.15.26 (21536000/21024000), Serial0/0.4
P 172.20.15.16/30, 1 successors, FD is 6535936
        via 172.20.15.26 (6535936/6023936), Serial0/0.4
P 172.20.15.24/30, 1 successors, FD is 2169856
        via Connected, Serial0/0.4
P 172.20.10.0/27, 1 successors, FD is 2707456
        via 172.20.15.26 (2707456/2195456), Serial0/0.4
P 172.20.15.4/30, 1 successors, FD is 3193856
        via 172.20.15.26 (3193856/2681856), Serial0/0.4
P 172.20.15.0/30, 1 successors, FD is 2681856
        via 172.20.15.26 (2681856/2169856), Serial0/0.4
P 172.20.15.12/30, 1 successors, FD is 4035840
        via 172.20.15.26 (4035840/3523840), Serial0/0.4
P 172.18.0.0/16, 1 successors, FD is 4038400
        via 172.20.15.26 (4038400/3526400), Serial0/0.4
P 172.20.15.8/30, 1 successors, FD is 11535872
        via 172.20.15.26 (11535872/11023872), Serial0/0.4
P 192.168.18.128/25, 1 successors, FD is 28160
        via Connected, FastEthernet0/2
P 10.108.16.0/24, 1 successors, FD is 284160
        via 172.20.32.2 (284160/281600), FastEthernet0/1
P 172.20.15.24/30, 1 successors, FD is 2169856
        via Connected, Serial0/0.4
Cochran#
```

Now, Lindbergh is configured as an EIGRP stub in Example 7-36.

## Example 7-36. Lindbergh is configured as an EIGRP stub router.

```
router eigrp 15
 eigrp stub
```

Example 7-37 shows Cochran's EIGRP neighbor table, and Example 7-38 shows Cochran's EIGRP topology table after the same two links to Earhart fail again.

## Example 7-37. Cochran's EIGRP neighbor table shows its neighbors. Lindbergh is a stub. Cochran is running a later IOS release (12.3) then Earhart. Notice that the fact that queries are suppressed is explicitly stated.

```
Cochran#show ip eigrp neighbor detail
IP-EIGRP neighbors for process 15
H   Address                 Interface       Hold Uptime    SRTT   RTO   Q   Seq
                                            (sec)          (ms)        Cnt  Num
2   172.20.15.26            Se0/0.4           10 00:01:08    56   336   0   20
   Version 12.3/1.2, Retrans: 2, Retries: 0
   Stub Peer Advertising ( CONNECTED SUMMARY ) Routes
   Suppressing queries
1   172.20.15.9             Se0/0.2           11 00:29:46    96    576  0   111
   Version 12.1/1.2, Retrans: 1, Retries: 0
0   172.20.15.5             Se0/0.1           10 00:37:00    96    576  0   110
   Version 12.1/1.2, Retrans: 2, Retries: 0
IP-EIGRP neighbors for process 10
H   Address                 Interface       Hold Uptime    SRTT   RTO   Q   Seq
                                            (sec)          (ms)        Cnt  Num
0   172.20.32.2             Fa0/1             13 00:29:50    51   306   0   10
   Version 12.1/1.2, Retrans: 2, Retries: 0
Cochran#
```

## Example 7-38. After failure of the primary links between Cochran and Earhart, only addresses connected to Lindbergh are reachable via Serial 0/0.4.

```
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 15: Neighbor 172.20.15.5 (Serial0/0.1) is down:
interface down
%DUAL-5-NBRCHANGE: IP-EIGRP(0) 15: Neighbor 172.20.15.9 (Serial0/0.2) is down:
interface down

Cochran#show ip eigrp topology
IP-EIGRP Topology Table for AS(15)/ID(192.168.17.1)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 192.168.18.24/29, 1 successors, FD is 2195456
        via 172.20.15.26 (2195456/281600), Serial0/0.4
P 192.168.16.0/24, 1 successors, FD is 2195456
        via 172.20.15.26 (2195456/281600), Serial0/0.4
P 192.168.17.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
P 172.20.32.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/1
P 172.20.15.24/30, 1 successors, FD is 2169856
        via Connected, Serial0/0.4
P 172.20.15.0/30, 1 successors, FD is 2681856
        via 172.20.15.26 (2681856/2169856), Serial0/0.4
P 192.168.18.128/25, 1 successors, FD is 28160
        via Connected, FastEthernet0/2
```

```
P 10.108.16.0/24, 1 successors, FD is 284160
        via 172.20.32.2 (284160/281600), FastEthernet0/1
P 172.20.15.24/30, 1 successors, FD is 2169856
        via Connected, Serial0/0.4
Cochran#
```

Configuring Lindbergh as an EIGRP stub prevents it from becoming a transit router during a failure of core links.

Configuring stub routing with EIGRP greatly increases the scalability of an EIGRP network, by minimizing queries, and thus the amount of time that network outages require addresses to be in an active state.

Stub routing eliminates queries sent to the stub router, but it does nothing to hide the topology of the rest of the network from the stub's point of view. Earhart can hide the topology of the rest of the network from the stubs. They don't need to know about every individual subnet because all packets for each of the subnets are always forwarded to the hub. Earhart can accomplish this by summarizing addresses.

## Case Study: Address Summarization

Router Yeager, shown in the network in Figure 7-37, has a single link to Earhart. The six addresses that Earhart must advertise to Yeager can be summarized with two aggregate addresses. Earhart's configuration will be as shown in Example 7-39.

### Example 7-39. Earhart's configuration summarizes routes to Yeager.

```
interface Ethernet2
 ip address 10.15.15.254 255.255.255.252
 ip summary-address eigrp 15 172.0.0.0 255.0.0.0
 ip summary-address eigrp 15 192.168.16.0 255.255.240.0
```

The ip summary-address eigrp command will automatically suppress the advertisement of the more specific networks to Yeager. Example 7-40 shows the route table of Yeager before and after the aggregate addresses are configured. Even in this small network, the number of EIGRP-learned entries has been reduced by half; in a large network, the impact on route tables and the memory required to store them can be significant.

### Example 7-40. Yeager's route table before and after aggregate addresses are configured at Earhart.

```
Yeager#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

D    172.18.0.0/16 [90/3040000] via 10.15.15.254, 00:13:07, Ethernet0
D    172.20.0.0/16 [90/307200] via 10.15.15.254, 00:13:07, Ethernet0
```

```
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.10.1.0/24 is directly connected, Ethernet1
C       10.15.15.252/30 is directly connected, Ethernet0
     192.168.17.0/27 is subnetted, 1 subnets
D       192.168.17.0 [90/2198016] via 10.15.15.254, 00:03:57, Ethernet0
D    192.168.16.0/24 [90/2221056] via 10.15.15.254, 00:01:51, Ethernet0
     192.168.18.0/24 is variably subnetted, 2 subnets, 2 masks
D       192.168.18.24/29 [90/2221056] via 10.15.15.254, 00:13:09, Ethernet0
D       192.168.18.128/25 [90/2198016] via 10.15.15.254, 00:13:09, Ethernet0

Yeager#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.10.1.0/24 is directly connected, Ethernet1
C       10.15.15.252/30 is directly connected, Ethernet0
D    172.0.0.0/8 [90/307200] via 10.15.15.254, 00:00:57, Ethernet0
D    192.168.16.0/20 [90/2198016] via 10.15.15.254, 00:00:57, Ethernet0
```

## Authentication

MD5 cryptographic checksums are the only authentication supported in EIGRP, which on first consideration might seem less flexible than RIPv2 and OSPF, which support both MD5 and clear-text passwords. However, clear-text password authentication should be used only when a neighboring device does not support the more secure MD5. Because EIGRP will be spoken only between two Cisco devices, this situation will never arise.

The steps for configuring EIGRP authentication are

Step 1. Define a key chain with a name.

Step 2. Define the key or keys on the key chain.

Step 3. Enable authentication on an interface and specify the key chain to be used.

Step 4. Optionally configure key management.

Key-chain configuration and management are described in Chapter 6. EIGRP authentication is enabled and linked to a key chain on an interface with the commands `ip authentication key-chain eigrp` and `ip authentication mode eigrp md5`.[17]

[17] Although MD5 is the only authentication mode available, the **ip authentication mode eigrp md5** command anticipates the possibility of another mode being available in the future.

Referring to Figure 7-37, the configuration in Example 7-41 enables EIGRP authentication on Cochran's interface to Earhart.

Example 7-41. Cochran is configured to use MD5 authentication with Earhart.

```
key chain Edwards
 key 1
 key-string PanchoBarnes
!
interface Serial0/0.1
 ip address 172.20.15.6 255.255.255.252
 ip authentication key-chain eigrp 15 Edwards
 ip authentication mode eigrp 15 md5
interface Serial0/0.2
 ip address 172.20.15.10 255.255.255.252
 ip authentication key-chain eigrp 15 Edwards
 ip authentication mode eigrp 15 md5
```

PREV NEXT

# Troubleshooting EIGRP

Troubleshooting the exchange of RIP route information is a reasonably simple procedure. Routing updates are either propagated or they are not, and they either contain accurate information or they do not. The added complexity of EIGRP means an added complexity to the troubleshooting procedure. Neighbor tables and adjacencies must be verified, the query/response procedure of DUAL must be followed, and the influences of VLSM on automatic summarization must be considered.

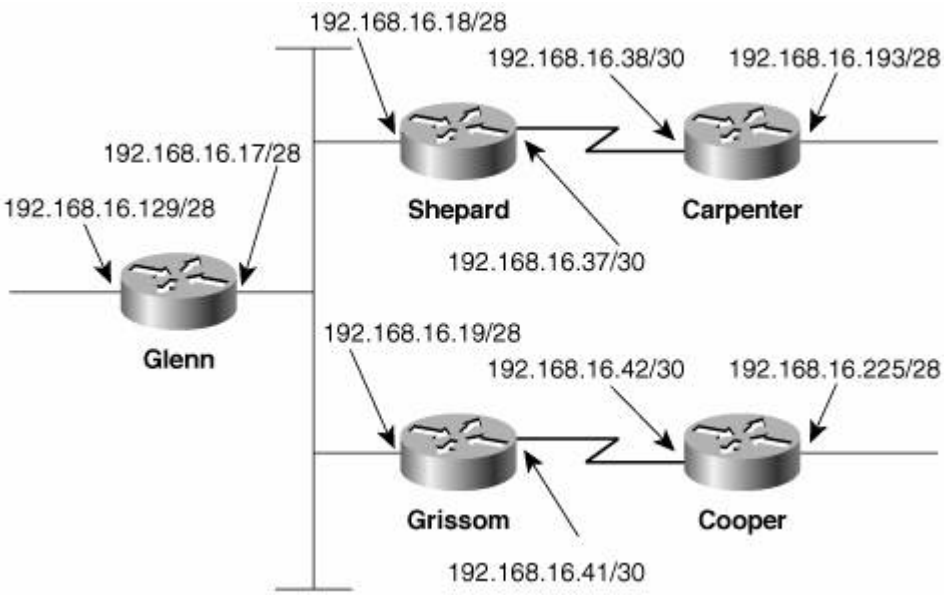
This section's case study describes a sequence of events that typically can be used when pursuing an EIGRP problem. Following the case study is a discussion of an occasional cause of instabilities in larger EIGRP internets.

## Case Study: A Missing Neighbor

Figure 7-38 shows a small EIGRP network. Users are complaining that subnet 192.168.16.224/28 is unreachable. An examination of the route tables reveals that something is wrong at router Grissom (Example 7-42).[18]

[18] When troubleshooting a network, it is a good practice to verify that the addresses of all router interfaces belong to the correct subnet.

Figure 7-38. Subnet 192.168.16.224/28 is not reachable through Grissom in this example of an EIGRP network.




Example 7-42. The route tables of Shepard and Grissom show that Grissom's EIGRP process is not advertising or receiving routes on subnet 192.168.16.16/28.

```
Grissom#show ip route
```

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
     192.168.16.0/24 is variably subnetted, 3 subnets, 2 masks
C       192.168.16.40/30 is directly connected, Serial0
C       192.168.16.16/28 is directly connected, Ethernet0
D       192.168.16.224/28 [90/2195456] via 192.168.16.42, 01:07:26, Serial0
```

---

```
Shepard#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
     192.168.16.0/24 is variably subnetted, 4 subnets, 2 masks
C       192.168.16.36/30 is directly connected, Serial0
C       192.168.16.16/28 is directly connected, Ethernet0
D       192.168.16.192/28 [90/2297856] via 192.168.16.38, 01:07:20, Serial0
D       192.168.16.128/28 [90/307200] via 192.168.16.17, 01:07:20, Ethernet0
```

The following observations are made from the two route tables of Example 7-42:

- Shepard does not have subnets 192.168.16.40/30 and 192.168.16.224/28 in its route table, although Grissom does.
- Grissom's route table does not contain any of the subnets that should be advertised by Glenn or Shepard.
- Shepard's route table contains the subnets advertised by Glenn (and Glenn's table contains the subnets advertised by Shepard, although its route table is not included in Example 7-42).

The conclusion to be drawn from these observations is that Grissom is not advertising or receiving routes correctly over subnet 192.168.16.16/28.

Among the possible causes, the simplest causes should be examined first. These follow:

- An incorrect interface address or mask
- An incorrect EIGRP process ID
- A missing or incorrect network statement

In this case, there are no EIGRP or address configuration errors.

Next, the neighbor tables should be examined. Looking at the neighbor tables at Grissom, Shepard, and Glenn (Example 7-43), two facts stand out:

- Grissom (192.168.16.19) is in its neighbors' tables, but its neighbors are not in Grissom's neighbor table.
- The entire network has been up for more than five hours; this information is reflected in the *uptime* statistic for all neighbors except Grissom. However, Grissom's uptime shows

approximately one minute.

### Example 7-43. Shepard and Glenn see Grissom as a neighbor, but Grissom does not see them. This suggests that Shepard and Glenn are receiving Hellos from Grissom, but Grissom is not receiving Hellos from Shepard and Glenn.

```
Grissom#show ip eigrp neighbors
IP-EIGRP neighbors for process 75
H   Address           Interface   Hold    Uptime    SRTT    RTO    Q    Seq
                                  (sec)             (ms)          Cnt   Num
0   192.168.16.42     Se0           11    05:27:11   23     200    0    8
```

---

```
Shepard#show ip eigrp neighbors
IP-EIGRP neighbors for process 75
H   Address           Interface   Hold    Uptime    SRTT    RTO    Q    Seq
                                  (sec)             (ms)          Cnt   Num
1   192.168.16.19     Et0           12    00:01:01    0    5000    1    0
2   192.168.16.17     Et0           11    05:27:33    8     200    0    6
0   192.168.16.38     Se0           14    05:27:34   22     200    0    10
```

---

```
Glenn#show ip eigrp neighbors
IP-EIGRP neighbors for process 75
H   Address           Interface   Hold    Uptime    SRTT    RTO    Q    Seq
                                  (sec)             (ms)          Cnt   Num
1   192.168.16.19     Et0           14    00:00:59    0    8000    1    0
2   192.168.16.18     Et0           10    05:30:11    9      20    0    7
0   192.168.16.130    Et1          12    05:30:58    6      20    0    7
```

If Grissom is in Shepard's neighbor table, Shepard must be receiving Hellos from it. Grissom, however, is apparently not receiving Hellos from Shepard. Without this two-way exchange of Hello packets, an adjacency will not be established and route information will not be exchanged.

A closer examination of Shepard's and Glenn's neighbor tables reinforces this hypothesis:

- The SRTT for Grissom is 0, indicating that a packet has never made the round trip.
- The RTO for Grissom has increased to five and eight seconds, respectively.
- There is a packet enqueued for Grissom (Q Cnt).
- The sequence number recorded for Grissom is 0, indicating that no reliable packets have ever been received from it.

These factors indicate that the two routers are trying to send a packet reliably to Grissom, but are not receiving an ACK.

In Example 7-44, `debug eigrp packets` is used at Shepard to get a better look at what is happening. All EIGRP packet types will be displayed, but a second debug command is used with it: `debug ip eigrp neighbor 75 192.168.16.19`. This command adds a filter to the first command. It tells `debug eigrp packet` to display only IP packets of EIGRP 75 (the process ID of the routers in Figure 7-38) and only those packets that concern neighbor 192.168.16.19 (Grissom).

### Example 7-44. The command *debug ip eigrp neighbor* is used to control the packets displayed by debug eigrp packets.

```
Shepard#debug eigrp packets
EIGRP Packets debugging is on
   (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK)
Shepard#debug ip eigrp neighbor 75 192.168.16.19
IP Neighbor target enabled on AS 75 for 192.168.16.19
IP-EIGRP Neighbor Target Events debugging is on
EIGRP: Sending UPDATE on Ethernet0 nbr 192.168.16.19, retry 14, RTO 5000
   AS 75, Flags 0x1, Seq 22/0 idbQ 1/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno
1-4
EIGRP: Received HELLO on Ethernet0 nbr 192.168.16.19
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Sending UPDATE on Ethernet0 nbr 192.168.16.19, retry 15, RTO 5000
AS 75, Flags 0x1, Seq 22/0 idbQ 1/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno
1-4
EIGRP: Received HELLO on Ethernet0 nbr 192.168.16.19
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Sending UPDATE on Ethernet0 nbr 192.168.16.19, retry 16, RTO 5000
   AS 75, Flags 0x1, Seq 22/0 idbQ 1/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno
1-4
EIGRP: Received HELLO on Ethernet0 nbr 192.168.16.19
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/1
EIGRP: Retransmission retry limit exceeded
EIGRP: Received HELLO on Ethernet0 nbr 192.168.16.19
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0
EIGRP: Enqueueing UPDATE on Ethernet0 nbr 192.168.16.19 iidbQ un/rely 0/1 peerQ
un/rely 0/0 serno 1-4
EIGRP: Sending UPDATE on Ethernet0 nbr 192.168.16.19
   AS 75, Flags 0x1, Seq 23/0 idbQ 1/0 iidbQ un/rely 0/0 peerQ un/rely 0/1 serno
1-4
```

Example 7-44 shows that Hello packets are being received from Grissom. It also shows that Shepard is attempting to send updates to Grissom; Grissom is not acknowledging them. After the 16th retry, the message "Retransmission retry limit exceeded" is displayed. This exceeded limit accounts for the low uptime shown for Grissom in the neighbor tableswhen the retransmission retry limit is exceeded, Grissom is removed from the neighbor table. But because Hellos are still being received from Grissom, it quickly reappears in the table and the process begins again.

Example 7-45 shows the output from `debug eigrp neighbors` at Shepard. This command is not IP specific, but instead shows EIGRP neighbor events. Here, two instances of the events described in the previous paragraph are displayed: Grissom is declared dead as the retransmission limit is exceeded but is immediately "revived" when its next Hello is received.

## Example 7-45. *debug eigrp neighbors* displays neighbor events.

```
Shepard#debug eigrp neighbors
EIGRP Neighbors debugging is on
Shepard#
EIGRP: Retransmission retry limit exceeded
EIGRP: Holdtime expired
EIGRP: Neighbor 192.168.16.19 went down on Ethernet0
EIGRP: New peer 192.168.16.19
EIGRP: Retransmission retry limit exceeded
EIGRP: Holdtime expired
EIGRP: Neighbor 192.168.16.19 went down on Ethernet0
EIGRP: New peer 192.168.16.19
```

Although Example 7-44 shows that update packets are being sent to Grissom, observation of EIGRP packets at that router shows that they are not being received (Example 7-46).

## Example 7-46. Grissom is exchanging Hellos with Cooper via interface S0 and is sending Hellos out E0. However, Grissom is not receiving any EIGRP packets on interface EO.

```
Grissom#debug eigrp packets
EIGRP Packets debugging is on
     (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK)
Grissom#
EIGRP: Sending HELLO on Serial0
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Received HELLO on Serial0 nbr 192.168.16.42
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
EIGRP: Sending HELLO on Ethernet0
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Sending HELLO on Serial0
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Received HELLO on Serial0 nbr 192.168.16.42
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
EIGRP: Sending HELLO on Ethernet0
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Sending HELLO on Serial0
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Sending HELLO on Ethernet0
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Received HELLO on Serial0 nbr 192.168.16.42
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0 peerQ un/rely 0/0
EIGRP: Sending HELLO on Serial0
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
EIGRP: Sending HELLO on Ethernet0
   AS 75, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ un/rely 0/0
```

Because Grissom is successfully exchanging Hellos with Cooper, Grissom's EIGRP process must be working. Suspicion therefore falls on Grissom's Ethernet interface. An inspection of the configuration file shows that an access list is configured as an incoming filter on E0 in Example 7-47.

## Example 7-47. An incoming access-list is denying EIGRP packets.

```
interface Ethernet0
 ip address 192.168.16.19 255.255.255.240
 ip access-group 150 in
!
!
access-list 150 permit tcp any any established
access-list 150 permit tcp any host 192.168.16.238 eq ftp
access-list 150 permit tcp host 192.168.16.201 any eq telnet
access-list 150 permit tcp any host 192.168.16.230 eq pop3
access-list 150 permit udp any any eq snmp
access-list 150 permit icmp any 192.168.16.224 0.0.0.15
```

When EIGRP packets are received at Grissom's E0 interface, they are first filtered through access list 150. They will not match any entry on the list and are therefore being dropped. The problem is resolved (Example 7-48) by adding the following entry to the access list:

```
access-list 150 permit eigrp 192.168.16.16 0.0.0.15 any
```

Example 7-48. When an entry is added to the access list to permit EIGRP packets, Grissom's neighbor and route tables show that it now has routes to all subnets.

```
Grissom#show ip eigrp neighbors
IP-EIGRP neighbors for process 75
H   Address                 Interface    Hold Uptime    SRTT   RTO   Q   Seq
                                         (sec)          (ms)        Cnt  Num
2   192.168.16.17           Et0             10 00:06:20    4   200   0   41
1   192.168.16.18           Et0             14 00:06:24   15   200   0   85
0   192.168.16.42           Se0             10 06:22:56   22   200   0   12
Grissom#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
   192.168.16.0/24 is variably subnetted, 6 subnets, 2 masks
C    192.168.16.40/30 is directly connected, Serial0
D    192.168.16.36/30 [90/2195456] via 192.168.16.18, 00:06:27, Ethernet0
C    192.168.16.16/28 is directly connected, Ethernet0
D    192.168.16.224/28 [90/2195456] via 192.168.16.42, 00:06:12, Serial0
D    192.168.16.192/28 [90/2323456] via 192.168.16.18, 00:06:27, Ethernet0
D    192.168.16.128/28 [90/307200] via 192.168.16.17, 00:06:12, Ethernet0
Grissom#
```

## Stuck-in-Active Neighbors

When a route goes active and queries are sent to neighbors, the route will remain active until a reply is received for every query. But what happens if a neighbor is dead or otherwise incapacitated and cannot reply? The route would stay permanently active. The active timer and SIA-retransmit timer are designed to prevent this situation. Both the active timer and the SIA-retransmit timer are set when a query is sent. If the SIA-retransmit timer is not supported by the router's IOS (IOS versions earlier then 12.2[4.1]), only the active timer is used. If the timers expire before a reply to the query is received, the route is declared *stuck-in-active,* the neighbor is presumed dead, and it is flushed from the neighbor table.[19] The SIA route and any other routes via that neighbor are eliminated from the route table. DUAL will be satisfied by considering the neighbor to have replied with an infinite metric.

[19] As mentioned previously, the default active time is three minutes. It can be changed with the command **timers active-time**.

In reality, this sequence of events should never happen. The loss of Hellos should identify a disabled neighbor long before the active timer expires.

But what happens in large EIGRP networks where a query might, like the bunny in the battery advertisement, keep going and going? Remember that queries cause the diffusing calculation to grow larger, whereas replies cause it to grow smaller (refer to Figure 7-6). Queries must eventually reach the edge of the network, and replies must eventually begin coming back, but if the diameter of the diffusing calculation grows large enough, an active timer might expire before all replies are received. The result, flushing a legitimate neighbor from the neighbor table, is obviously destabilizing.

When neighbors mysteriously disappear from neighbor tables and then reappear, or users complain of intermittently unreachable destinations, SIA routes might be the culprit. Checking the error logs of routers is a good way to find out whether SIAs have occurred (Example 7-49).

## Example 7-49. The final entry of this error log shows a SIA message.

```
Gagarin#show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level debugging, 3369 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 71 message lines logged
    Buffer logging: level debugging, 3369 messages logged
Log Buffer (4096 bytes):
   ...
   ...
   ...
DUAL: dual_rcvupdate(): 10.51.1.0/24 via 10.1.2.1 metric 409600/128256
DUAL: Find FS for dest 10.51.1.0/24. FD is 4294967295, RD is 4294967295 found
DUAL: RT installed 10.51.1.0/24 via 10.1.2.1
DUAL: Send update about 10.51.1.0/24. Reason: metric chg
DUAL: Send update about 10.51.1.0/24. Reason: new if
DUAL: dual_rcvupdate(): 10.52.1.0/24 via 10.1.2.1 metric 409600/128256
DUAL: Find FS for dest 10.52.1.0/24. FD is 4294967295, RD is 4294967295 found
%DUAL-3-SIA: Route 10.11.1.0/24 stuck-in-active state in IP-EIGRP 1. Cleaning up
Gagarin#
```

When chasing the cause of SIAs, close attention should be paid to the topology table in routers. If routes can be "caught" in the active state, the neighbors from whom queries have not yet been received should be noted. For example, Example 7-50 shows a topology table in which several routes are active. Notice that most of them have been active for 15 seconds and that one (10.6.1.0) has been active for 41 seconds.

## Example 7-50. This topology table shows several active routes, all of which are waiting for a reply from neighbor 10.1.2.1.

```
Gagarin#show ip eigrp topology
IP-EIGRP Topology Table for process 1
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply
       r - Reply Status
A 10.11.1.0/24, 0 successors, FD is 3072128000, Q
    1 replies, active 00:00:15, query-origin: Local origin
    Remaining replies:
        via 10.1.2.1, r, Ethernet0
A 10.10.1.0/24, 0 successors, FD is 3584128000, Q
    1 replies, active 00:00:15, query-origin: Local origin
    Remaining replies:
        via 10.1.2.1, r, Ethernet0
A 10.9.1.0/24, 0 successors, FD is 4096128000, Q
    1 replies, active 00:00:15, query-origin: Local origin
    Remaining replies:
        via 10.1.2.1, r, Ethernet0
A 10.2.1.0/24, 1 successors, FD is Inaccessible, Q
    1 replies, active ve 00:00:15, query-origin: Local origin
    Remaining res:
        via 10.1.2.1, r, Ethernet0
P 10.1.2.0/24, 1 successors, FD is 281600
        via Connected, Ethernet0
```

```
A 10.6.1.0/24, 0 successors, FD is 3385160704, Q
    1 replies, active 00:00:41, query-origin: Local origin
    Remaining replies:
        via 10.1.2.1, r, Ethernet0
A 10.27.1.0/24, 0 successors, FD is 3897160704, Q
--More-
```

Notice also that in each case, the neighbor 10.1.2.1 has its reply status flag (r) set. That is the neighbor from which replies have not yet been received. There might be no problem with the neighbor itself or with the link to the neighbor, but this information points to the direction within the network topology in which the investigation should proceed.

Common causes of SIAs in larger EIGRP networks are heavily congested, low-bandwidth data links and routers with low memory or overutilized CPUs. The problem will be exacerbated if these limited resources must handle very large numbers of queries.

The careless adjustment of the bandwidth parameter on interfaces might be another cause of SIAs. Recall that EIGRP is designed to use no more than 50 percent of the available bandwidth of a link. This restriction means that EIGRP's pacing is keyed to the configured bandwidth. If the bandwidth is set artificially low in an attempt to manipulate routing choices, the EIGRP process might be starved. If IOS 11.2 or later is being run, the command ip bandwidth-percent eigrp may be used to adjust the percentage of bandwidth used.

For example, suppose that an interface is connected to a 56K serial link, but the bandwidth is set to 14K. EIGRP would limit itself to 50 percent of this amount, or 7K. The commands in Example 7-51 adjust the EIGRP bandwidth percent to 200 percent200 percent of 14K, which is 50 percent of the actual bandwidth of the 56K link.

## Example 7-51. Router configuration adjusts the percentage of the configured bandwidth that EIGRP will use.

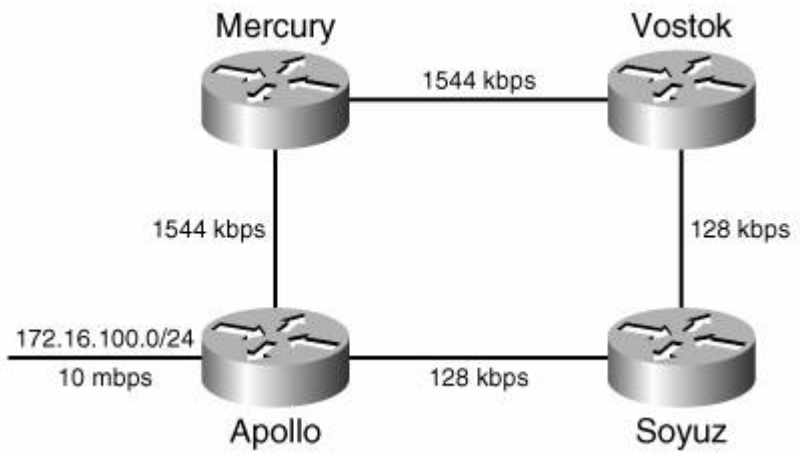
```
interface Serial 3
 ip address 172.18.107.210 255.255.255.240
 bandwidth 14
 ip bandwidth-percent eigrp 1 200
```

Increasing the active timer period with the timers active-time command might help avoid SIAs in some situations, but this step should not be taken without careful consideration of the effects it might have on reconvergence.

A new timer, the SIA-retransmit timer, and the two new EIGRP packet types, SIA-query and SIA-reply, help to minimize SIAs and to push the reset of the neighbor to link that is actually having the problem responding to queries.

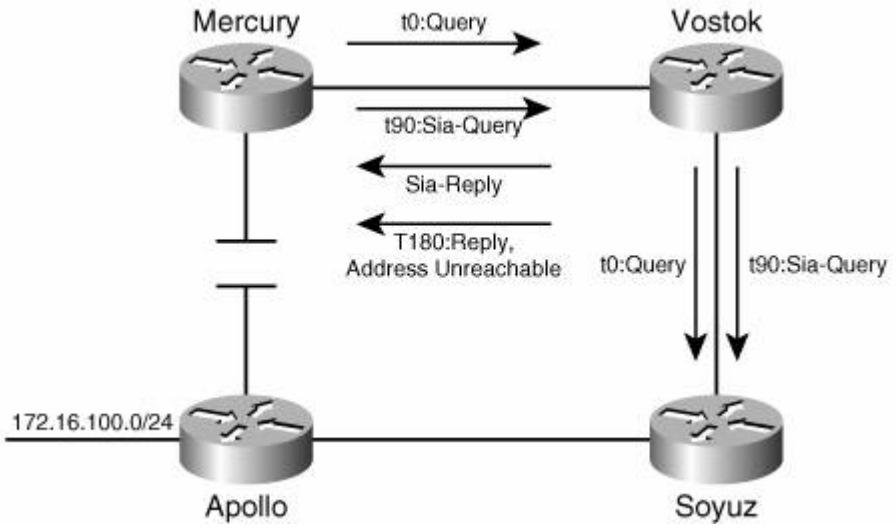
Consider the network in Figure 7-39. From router Mercury, EIGRP will route traffic to network 172.16.100.0 via Apollo. Vostok is not a feasible successor because the metric from Vostok to 172.16.100.0 is too high. Vostok routes traffic to 172.16.100.0 via Mercury and Apollo. Soyuz is not a feasible successor because the metric from Soyuz to 172.16.100.0 is too high.

## Figure 7-39. Mercury does not list Vostok as a feasible successor to 172.16.100.0/24.




When the link between Mercury and Apollo fails, as shown in Figure 7-40, Mercury places the address 172.16.100.0 (and any other address known via neighbor Apollo) into Active, and sends a query to Vostok. Vostok also places the address into Active state, and sends a query to Soyuz. The Active timers are set. In addition, the SIA-retransmit timers are set. The SIA-retransmit timer is set to one-half the value of the Active timer, typically 90 seconds.

Figure 7-40. SIA-queries and SIA-replies are used to avoid SIA conditions.




When the SIA-retransmit timer expires, Mercury sends an SIA-query to Vostok. Vostok sends an SIA-query to Soyuz. Vostok responds to the SIA-query from Mercury with an SIA-reply. Mercury resets the Active timer and the SIA-retransmit timer. The routers will send up to three SIA-queries (assuming no reply has been received from the original address query) as long as SIA-replies are received, before resetting a neighbor. So as long as a neighbor router responds to the SIA-queries, it won't be declared *stuck-in-active* and reset, for six minutes, assuming a default Active time of 180 seconds. This gives ample time for a large network to respond to queries.

But, say there is a problem on the link from Vostok to Soyuz that is allowing enough Hellos to get through to keep the neighbors active, but the SIA-reply is not received by Vostok within the SIA-retransmit time. If no SIA-reply is received within 90 seconds of a SIA-query, and no response to the original address query has been received, Vostok will reset neighbor Soyuz and reply to Mercury's original query that the address is unreachable.

The SIA-retransmit timer does two things. If neighbors are responding to SIA-queries, large networks are given more time to respond to address queries. If neighbors are not responding, the neighbor is reset. Only the router that is not receiving responses from its neighbor will reset the adjacency. Before the SIA-retransmit timer was introduced, any router that did not receive a response to an active query after the Active timer expired would reset the neighbor adjacency, even if the problem was somewhere downstream in the network.

A good network design is the best solution to instabilities such as SIA routes. By using a combination of intelligent address assignment, route filtering, default routes, stub routing, and summarization, boundaries may be constructed in a large EIGRP network to restrict the size and scope of diffusing computations. Chapter 13 includes an example of such a design.

PREV NEXT

# Looking Ahead

When comparing EIGRP and OSPF, it is often said that an advantage of EIGRP is that it is simpler to configure. This observation is true for many networks, but this chapter's discussion of troubleshooting shows that as a network grows, efforts must be made to "compartmentalize" the EIGRP topology. Ironically, the very complexity of OSPF might make it easier to configure in large networks, as the next chapter shows.

# Summary Table: Chapter 7 Command Review

| Command | Description |
|---|---|
| accept-lifetime *start-time* {infinite \| *end-time* \| duration *seconds*} | Specifies the time period during which the authentication key on a key chain is received as valid. |
| auto-summary | Enables automatic summarization at network boundaries. This command is enabled by default. |
| bandwidth *kilobits* | Specifies the bandwidth parameter, in kilobits per second, on an interface. |
| debug eigrp packets | Displays EIGRP packet activity. |
| debug ip eigrp neighbor *process-id address* | Adds a filter to the debug eigrp packets command, telling it to display only IP packets for the indicated process and neighbor. |
| delay *tens-of-microseconds* | Specifies the delay parameter, in tens of microseconds, on an interface. |
| eigrp stub {connected \| redistributed \| static \| summary \| receive-only} | Configures a spoke router as an EIGRP stub. |
| ip authentication key-chain eigrp *process-id key-chain* | Configures a key chain on an EIGRP interface and specifies the name of the key chain to be used. |
| ip authentication mode eigrp *process-id md5* | Enables EIGRP authentication on an interface. |
| ip bandwidth-percent eigrp *process-id percent* | Configures the percentage of bandwidth used by EIGRP; the default is 50 percent. |
| ip hello-interval eigrp *process-id seconds* | Configures the EIGRP hello interval. |
| ip hold-time eigrp *process-id seconds* | Configures the EIGRP hold time. |
| ip summary-address eigrp *process-id address mask* | Configures a router to send a summary EIGRP advertisement. |
| key *number* | Specifies a key on a key chain. |
| key chain *name-of-chain* | Specifies a group of authentication keys. |
| key-string *text* | Specifies the authentication string, or password, used by a key. |
| metric weights *tos k1 k2 k3 k4 k5* | Specifies how much weight the bandwidth, load, delay, reliability, and MTU size parameters should be given in the IGRP and EIGRP metric calculations. |
| network *network-number* | Specifies the network address of one or more interfaces on which IGRP, EIGRP, or RIP processes should be enabled. |
| passive-interface *type number* | Disables the transmission of broadcast or multicast routing updates on an interface. |

| Command | Description |
| --- | --- |
| router eigrp *process-id* | Enables an EIGRP process. |
| send-lifetime *start-time* {infinite \| *end-time* \| duration *seconds*} | Specifies the time period during which the authentication key on a key chain may be sent. |
| show ip eigrp neighbors [*type number*] | Displays the EIGRP neighbor table. |
| show ip eigrp topology [*process-id* \| [[*ip address*] *mask*]] | Displays the EIGRP topology table. |
| timers active-time {*minutes* \| disabled} | Changes or disables the default 3-minute active time. |
| traffic-share {balanced \| min} | Specifies whether an IGRP or EIGRP routing process should use unequal-cost load balancing or equal-cost load balancing only. |
| variance *multiplier* | Specifies a route multiplier by which a route metric can vary from the lowest-cost metric and still be included in an unequal-cost load balancing group. |

PREV NEXT

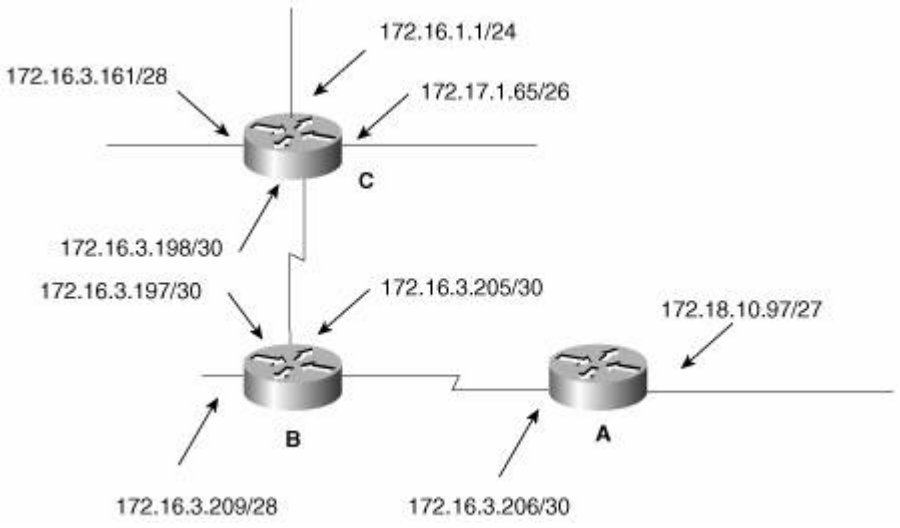
# Review Questions

1. Is EIGRP a distance vector or a link-state routing protocol?
2. What is the maximum configured bandwidth EIGRP will use on a link? Can this percentage be changed?
3. How do EIGRP and IGRP differ in the way they calculate the composite metric?
4. What are the four basic components of EIGRP?
5. In the context of EIGRP, what does the term *reliable delivery* mean? Which two methods ensure reliable delivery of EIGRP packets?
6. Which mechanism ensures that a router is accepting the most recent route entry?
7. What is the multicast IP address used by EIGRP?
8. What are the packet types used by EIGRP?
9. At what interval, by default, are EIGRP Hello packets sent?
10. What is the default hold time?
11. What is the difference between the neighbor table and the topology table?
12. What is a feasible distance?
13. What is the feasibility condition?
14. What is a feasible successor?
15. What is a successor?
16. What is the difference between an active route and a passive route?
17. What causes a passive route to become active?
18. What causes an active route to become passive?
19. What does stuck-in-active mean?
20. What is the difference between subnetting and address aggregation?

# Configuration Exercises

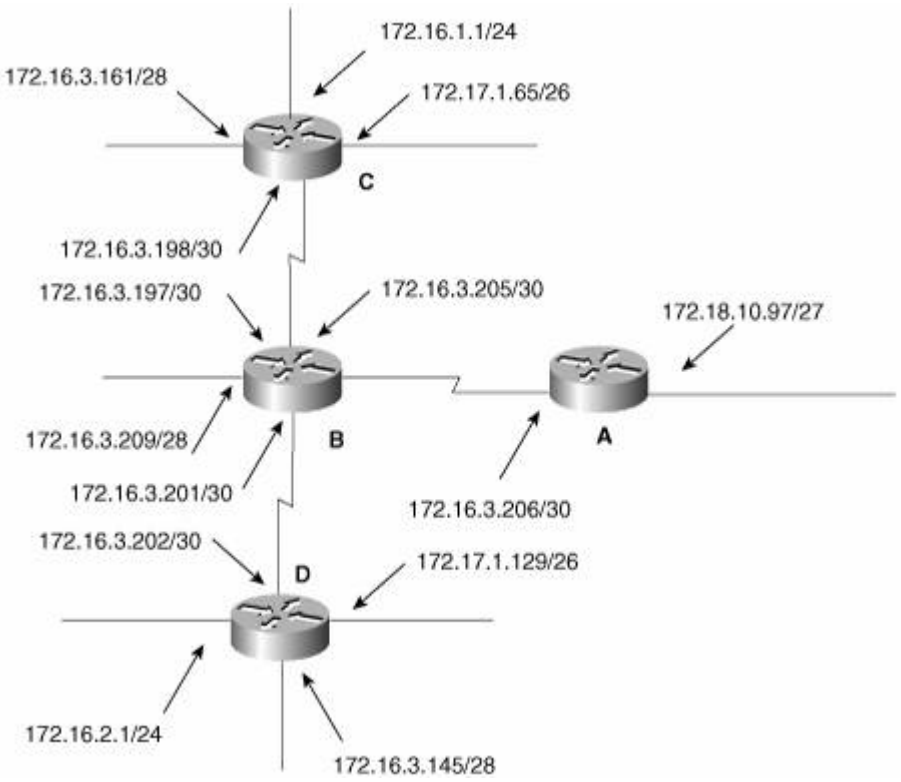
1 Write EIGRP configurations for Routers A, B, and C in Figure 7-41. Use process ID 5.

Figure 7-41. Network for Configuration Exercises 1 and 2.




2 The serial interfaces connecting Routers A and B in Figure 7-41 are both S0. Configure authentication between these two routers, using the first key two days from today's date. Configure a second key to be used beginning 30 days after the first key.

Figure 7-42. Network for Configuration Exercise 3.

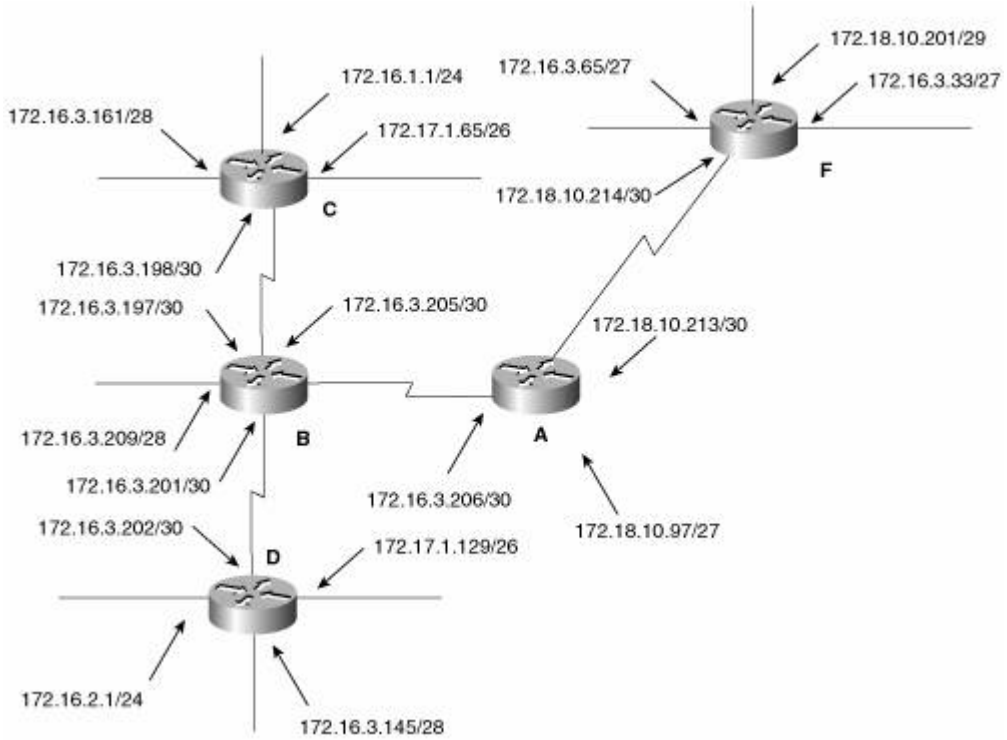



3 Router D is added in Figure 7-42. Add this router to the configurations written in Configuration Exercise 2.

4 Router F has been added in Figure 7-43. Configure this router to run EIGRP with the routers configured in Configuration Exercises 2 and 3.

Figure 7-43. Network for Configuration Exercises 4 and 5.

[View full size image]




5 Configure route summarization wherever possible in the network of Figure 7-43.

PREV NEXT

# Troubleshooting Exercises

1 Table 7-7 shows the values displayed in the show interface command for every interface in Figure 7-44. Which router will router F use as the successor to subnet A?

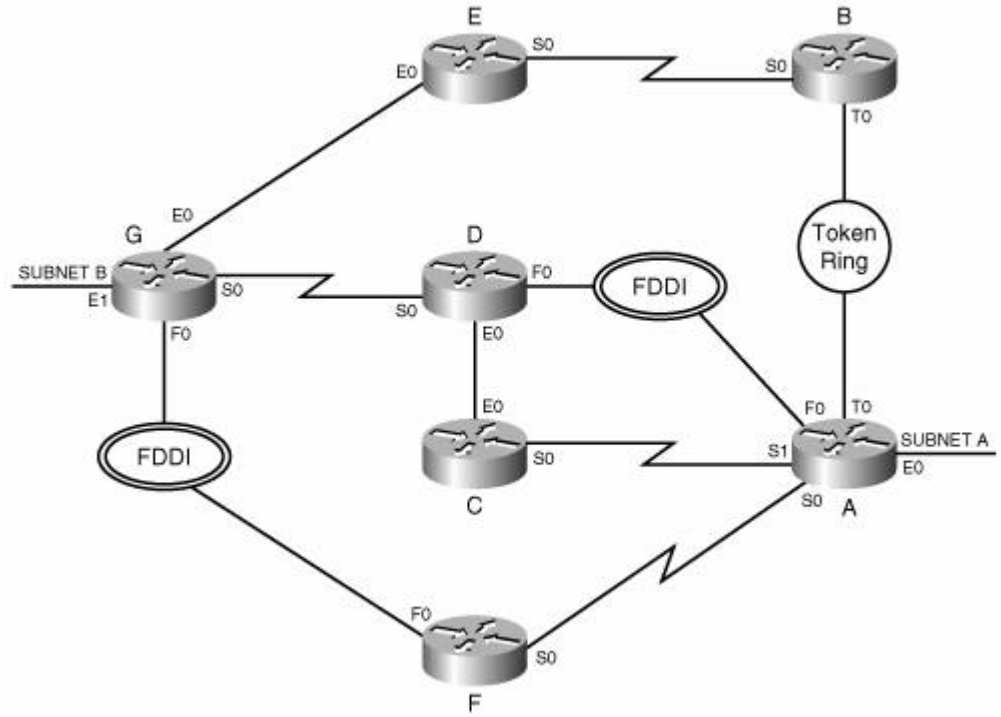
Table 7-7. Metric values for all interfaces in Figure 7-44, as displayed in the show interface command.

| Router | Interface | BW(k) | DLY(mS) |
|---|---|---|---|
| A | E0 | 10000 | 1000 |
| | F0 | 100000 | 100 |
| | T0 | 16000 | 630 |
| | S0 | 512 | 20000 |
| | S1 | 1544 | 20000 |
| B | TO | 16000 | 630 |
| | S0 | 1544 | 20000 |
| C | E0 | 10000 | 1000 |
| | S0 | 1544 | 20000 |
| D | E0 | 10000 | 1000 |
| | F0 | 100000 | 100 |
| | S0 | 1544 | 20000 |
| E | E0 | 10000 | 1000 |
| | S0 | 1544 | 20000 |
| F | F0 | 100000 | 100 |
| | S0 | 512 | 20000 |
| G | E0 | 10000 | 1000 |
| | E1 | 10000 | 1000 |
| | F0 | 100000 | 100 |
| | S0 | 56 | 20000 |

Figure 7-44. Network for Troubleshooting Exercises 1 through

5.

[View full size image]




2 In Figure 7-44, what is router C's feasible distance to subnet A?

3 In Figure 7-44, what is router G's feasible distance to subnet A?

4 In Figure 7-44, which routers will router G show in its topology table as feasible successors?

PREV NEXT

# Chapter 8. OSPFv2

This chapter covers the following subjects:

- Operation of OSPF
- Configuring OSPF
- Troubleshooting OSPF

Open Shortest Path First (OSPF) was developed by the Internet Engineering Task Force (IETF) as a replacement for the problematic RIP and is now the IETF-recommended Interior Gateway Protocol (IGP). OSPF is a link-state protocol that, as the name implies, uses Dijkstra's Shortest Path First (SPF) algorithm and that is *open*that is, it isn't proprietary to any vendor or organization. OSPF has evolved through several RFCs, all of which were written by John Moy. Version 1 of the protocol was specified in RFC 1131; this version never progressed beyond the experimental stage. Version 2, which is still the current version for IPv4, was first specified in RFC 1247, and the most recent specification is RFC 2328.

Like all link-state protocols, OSPF's major advantages over distance vector protocols are fast reconvergence, scalability to much larger networks, and less susceptibility to bad routing information. Other features of OSPF are

- The use of areas, which reduces the protocol's impact on CPU and memory, contains the flow of routing protocol traffic, and makes possible the construction of hierarchical network topologies
- Fully classless behavior, eliminating such classful problems as discontiguous subnets
- Support of classless route table lookups, VLSM, and supernetting for efficient address management
- A dimensionless, arbitrary metric
- Equal-cost load balancing for more efficient use of multiple paths[1]

  [1] More accurately, the RFC calls for equal-cost multipath, the discovery and use of multiple equal-cost paths, without prescribing how the protocol should route individual packets across these multiple paths. The Cisco OSPF implementation performs equal-cost load balancing as described in previous chapters.

- The use of reserved multicast addresses to reduce the impact on non-OSPFspeaking devices
- Support of authentication for more secure routing
- The use of route tagging for the tracking of external routes

OSPF also has the capability of supporting Type of Service (TOS) routing, although it was never widely implemented. RFC 2328 has deleted the TOS routing option for this reason.

# Operation of OSPF[2]

[2] Because of the interrelationship of OSPF terms and concepts, this chapter frequently uses terms before they are fully defined. The reader is advised to read this section more than once to ensure a complete understanding of OSPF operation. It will also be useful to review the section "Link State Routing Protocols" in Chapter 4, "Dynamic Routing Protocols."

At a very high level, the operation of OSPF is easily explained:

1. OSPF-speaking routers send Hello packets out all OSPF-enabled interfaces. If two routers sharing a common data link agree on certain parameters specified in their respective Hello packets, they will become *neighbors*.
2. *Adjacencies*, which can be thought of as virtual point-to-point links, are formed between some neighbors. OSPF defines several network types and several router types. The establishment of an adjacency is determined by the types of routers exchanging Hellos and the type of network over which the Hellos are exchanged.
3. Each router sends *link-state advertisements* (LSAs) over all adjacencies. The LSAs describe all of the router's links, or interfaces, the router's neighbors, and the state of the links. These links might be to stub networks (networks with no other router attached), to other OSPF routers, to networks in other areas, or to external networks (networks learned from another routing process). Because of the varying types of link-state information, OSPF defines multiple LSA types.
4. Each router receiving an LSA from a neighbor records the LSA in its *link-state database* and sends a copy of the LSA to all of its other neighbors.
5. By flooding LSAs throughout an area, all routers will build identical link-state databases.
6. When the databases are complete, each router uses the SPF algorithm to calculate a loop-free graph describing the shortest (lowest cost) path to every known destination, with itself as the root. This graph is the SPF tree.
7. Each router builds its route table from its SPF tree.[3]

[3] This fundamental procedure of calculating routes from the link-state database, rather than by exchanging routes with neighbors, has repercussions for route filtering. See Chapter 13, "Route Filtering," for more information.

When all link-state information has been flooded to all routers in an area and neighbors have verified that their databases are identicalthat is, the link-state databases have been synchronizedand the route tables have been built, OSPF is a quiet protocol. Hello packets are exchanged between neighbors as keepalives, and LSAs are retransmitted every 30 minutes. If the network topology is stable, no other activity should occur.

## Neighbors and Adjacencies

Before any LSAs can be sent, OSPF routers must discover their neighbors and establish adjacencies. The neighbors will be recorded in a *neighbor table*, along with the link (interface) on which each neighbor is located and which contains other information necessary for the maintenance of the neighbor (Example 8-1).

Example 8-1. The neighbor table records all OSPF-speaking neighbors.

```
Monet#show ip ospf neighbor
Neighbor ID     Pri   State     Dead Time    Address         Interface
192.168.30.70     1   FULL/DR   00:00:34     192.168.17.73   Ethernet0
192.168.30.254    1   FULL/DR   00:00:34     192.168.32.2    Ethernet1
192.168.30.70     1   FULL/BDR  00:00:34     192.168.32.4    Ethernet1
192.168.30.30     1   FULL/ -   00:00:33     192.168.17.50   Serial0.23
192.168.30.10     1   FULL/ -   00:00:32     192.168.17.9    Serial1
192.168.30.68     1   FULL/ -   00:00:39     192.168.21.134  Serial2.824
192.168.30.18     1   FULL/ -   00:00:30     192.168.21.142  Serial2.826
192.168.30.78     1   FULL/ -   00:00:36     192.168.21.170  Serial2.836
```

The tracking of other OSPF routers requires that each router have a *Router ID*, an IP address by which the router is uniquely identified within the OSPF domain. Cisco routers derive their Router IDs by the following means:

1. If the Router ID has been manually configured using the `router-id` command, that Router ID is used.
2. If no Router ID has been manually configured, the router chooses the numerically highest IP address on any of its loopback interfaces.
3. If no loopback interfaces are configured with IP addresses, the router chooses the numerically highest IP address on any of its physical interfaces. The interface from which the Router ID is taken does not have to be running OSPF.

Using addresses associated with loopback interfaces has two advantages:

- The loopback interface is more stable than any physical interface. It is active when the router boots up, and it only fails if the entire router fails.
- The network administrator has more leeway in assigning predictable or recognizable addresses as the Router IDs.

The Cisco OSPF will continue to use a Router ID learned from a physical interface even if the interface subsequently fails or is deleted (see "Case Study: Setting Router IDs with Loopback Interfaces," later in this chapter). Therefore, the stability of a loopback interface is only a minor advantage. The primary benefit is the ability to control the Router ID.

The OSPF router begins a neighbor relationship by advertising its Router ID in Hello packets.

## Hello Protocol

The Hello protocol serves several purposes:

- It is the means by which neighbors are discovered.
- It advertises several parameters on which two routers must agree before they can become neighbors.
- Hello packets act as keepalives between neighbors.
- It ensures bidirectional communication between neighbors.
- It elects Designated Routers (DRs) and Backup Designated Routers (BDRs) on Broadcast and Nonbroadcast Multiaccess (NBMA) networks.

OSPF-speaking routers periodically send a Hello packet out each OSPF-enabled interface. This period is known as the *HelloInterval* and is configured on a per interface basis. Cisco uses a default HelloInterval of 10 seconds for broadcast networks and 30 seconds for non-broadcast; the value can be changed with the command `ip ospf hello-interval`. If a router has not heard a Hello from a neighbor within a period of time known as the RouterDeadInterval, it will declare the neighbor down. The Cisco default RouterDeadInterval is four times the HelloInterval and can be changed with the command `ip ospf dead-interval`.[4]

[4] RFC 2328 does not set a required value for either the HelloInterval or the RouterDeadInterval, although it does suggest respective values of 10 seconds and 4X HelloInterval.

Each Hello packet contains the following information:

- Router ID of the originating router.
- Area ID of the originating router interface.
- Address mask of the originating interface.
- Authentication type and authentication information for the originating interface.
- HelloInterval of the originating interface.
- RouterDeadInterval of the originating interface.
- Router Priority.
- DR and BDR.
- Five flag bits signifying optional capabilities.
- Router IDs of the originating router's neighbors. This list contains only routers from which Hellos were heard on the originating interface within the last RouterDeadInterval.

This section overviews the meaning and use of most of the information listed. Subsequent sections discuss the DR, BDR, and Router Priority, and illustrate the precise format of the Hello packet. When a router receives a Hello from a neighbor, it will verify that the Area ID, Authentication, Network Mask, HelloInterval, RouterDeadInterval, and Options values match the values configured on the receiving interface. If they do not, the packet is dropped and no adjacency is established.

If everything matches, the Hello packet is declared valid. If the ID of the originating router is already listed in the neighbor table for that receiving interface, the RouterDeadInterval timer is reset. If the Router ID is not listed, it is added to the neighbor table.

Whenever a router sends a Hello, it includes in the packet the Router IDs of all neighbors listed for the link on which the packet is to be transmitted. If a router receives a valid Hello in which it finds its own Router ID listed, the router knows that two-way communication has been established.

After two-way communication has been established, adjacencies may be established. However, as mentioned earlier, not all neighbors will become adjacent. Whether an adjacency is formed or not depends on the type of network to which the two neighbors are attached. Network types also influence the way in which OSPF packets are transmitted; therefore, before discussing adjacencies, it is necessary to discuss network types.

## Network Types

OSPF defines five network types:

- Point-to-point networks
- Broadcast networks
- Nonbroadcast Multiaccess (NBMA) networks
- Point-to-multipoint networks
- Virtual links

*Point-to-point* networks, such as a T1, DS-3, or SONET link, connect a single pair of routers. Valid neighbors on point-to-point networks will always become adjacent. The destination address of OSPF packets on these networks will always be the reserved class D address 224.0.0.5, known as *AllSPFRouters*.[5]

[5] The exception to this rule is retransmitted LSAs, which are always unicast on all network types. This exception is covered later, in the section "Reliable Flooding: Acknowledgments."

*Broadcast* networks, such as Ethernet, Token Ring, and FDDI, might be better defined as broadcast multi-access networks to distinguish them from NBMA networks. Broadcast networks are multi-access in that they are capable of connecting more than two devices, and they are broadcast in that all attached devices can receive a single transmitted packet. OSPF routers on broadcast networks will elect a DR and a BDR, as described in the next section, "Designated Routers and Backup Designated Routers." Hello packets are multicast with the AllSPFRouters destination address 224.0.0.5, as are all OSPF packets originated by the DR and BDR. The destination Media Access Control (MAC) identifier of the frames carrying these packets is 0100.5E00.0005. All other routers will multicast link-state update and link-state acknowledgment packets (described later) to the reserved class D address 224.0.0.6, known as *AllDRouters*. The destination MAC identifier of the frames carrying these packets is 0100.5E00.0006.

*NBMA* networks, such as X.25, Frame Relay, and ATM, are capable of connecting more than two routers but have no broadcast capability. A packet sent by one of the attached routers would not be received by all other attached routers. As a result, extra configuration might be necessary for routers on these networks to acquire their neighbors. OSPF routers on NBMA networks elect a DR and BDR, and all OSPF packets are unicast.

*Point-to-multipoint* networks are a special configuration of NBMA networks in which the networks are treated as a collection of point-to-point links. Routers on these networks do not elect a DR and BDR, and the OSPF packets are unicast to each known neighbor.

*Virtual links*, described in a later section, are special configurations that are interpreted by the router as unnumbered point-to-point networks. OSPF packets are unicast over virtual links.

In addition to these five network types, it should be noted that all networks fall into one of two more-general types:

- Transit networks have two or more attached routers. They might carry packets that are "just passing through"packets that were originated on and are destined for a network other than the transit network.
- Stub networks have only a single attached router.[6] Packets on a stub network always have either a source or a destination address belonging to that network. That is, all packets were either originated by a device on the network or are destined for a device on the network. OSPF advertises host routes (routes with a mask of 255.255.255.255) as stub networks. Loopback interfaces are also considered stub networks and are advertised as host routes.[7]

[6] Do not confuse stub networks with stub areas, discussed later in the chapter.

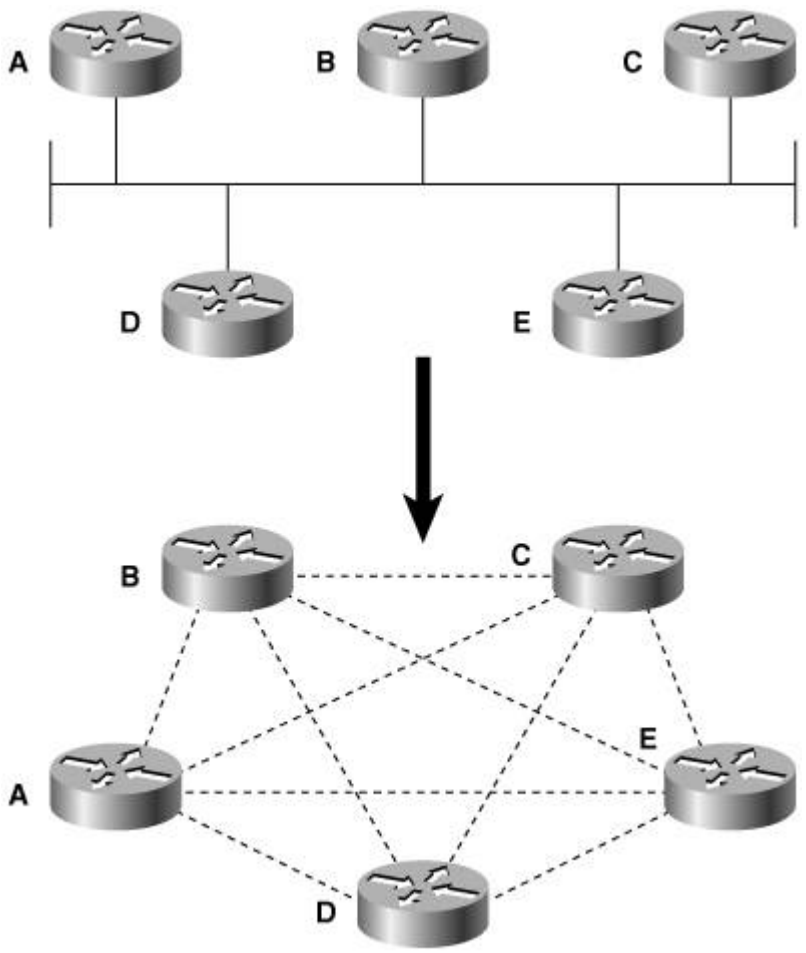
[7] Beginning with IOS 11.3, this default behavior can be changed by adding the command **ip ospf network point-to-point** to the loopback interface. This will cause the loopback interface's address to be advertised as a subnet route.

## Designated Routers and Backup Designated Routers

Multiaccess networks present two problems for OSPF, relating to the flooding of LSAs (described in a later section):

- The formation of an adjacency between every attached router would create many unnecessary LSAs. If *n* is the number of routers on a multiaccess network, there would be *n*(*n* 1)/2 adjacencies (Figure 8-1). Each router would flood *n* 1 LSAs for its adjacent neighbors, plus one LSA for the network, resulting in $n^2$ LSAs originating from the network.

Figure 8-1. Ten adjacencies would be required for each of the five routers on this OSPF network to become fully adjacent with all of its neighbors; 25 LSAs would be originated from the network.




- Flooding on the network itself would be chaotic and excessive. A router would flood an LSA to all its adjacent neighbors, which in turn would flood it to all their adjacent neighbors, creating many copies of the same LSA on the same network.

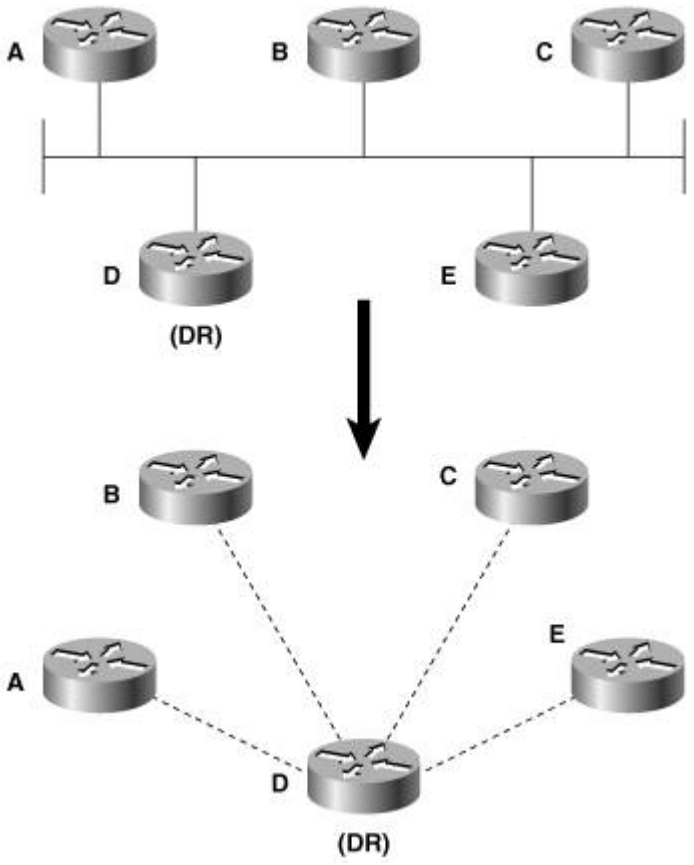
To prevent these problems, a DR is elected on multi-access networks. The DR has the following duties:

- To represent the multi-access network and its attached routers to the rest of the OSPF area
- To manage the flooding process on the multi-access network

The concept behind the DR is that the broadcast link itself is considered a "pseudonode," or a virtual router. When the SPF tree is calculated, the link appears as a node and the routers attached to the link are attached to that node. The cost from an attached router to the pseudonode is the outgoing cost of that router's interface to the broadcast link, but the cost from the pseudonode to any attached router is 0. This way, the overall path cost is not affected by the pseudonode.

Each router on the network forms an adjacency with the DR (Figure 8-2), which represents the pseudonode with a special Network LSA. Keep in mind that a router might be a DR on one of its attached multi-access networks, and it might not be the DR on another of its attached multi-access networks. In other words, the DR is a property of a router's interface, not the entire router.

Figure 8-2. The DR represents the multi-access network. Other routers on the network will form adjacencies with the DR, not with each other.




A significant problem with the DR scheme as described so far is that if the DR fails, a new DR must be elected. New adjacencies must be established, and all routers on the network must synchronize their databases with the new DR (part of the adjacency-building process). While all this is happening, the network is unavailable for transit packets.

To prevent this problem, a BDR is elected in addition to the DR. All routers form adjacencies not only with the DR but also with the BDR. The DR and BDR also become adjacent with each other. If the DR fails, the BDR becomes the new DR. Because the other routers on the network are already adjacent with the BDR, network unavailability is minimized.

The election of the DR and BDR is triggered by the interface state machine, which is described in a later section. For the election process to function properly, the following preconditions must

exist:

- Each multi-access interface of each router has a *Router Priority*, which is an 8-bit unsigned integer ranging from 0 to 255. The default priority on Cisco routers is 1 and can be changed on a per multi-access-interface basis with the command `ip ospf priority`. Routers with a priority of 0 are ineligible to become the DR or BDR.
- Hello packets include fields for the originating router to specify its Router Priority and for the IP addresses of the connected interfaces of the routers it considers the DR and BDR.
- When an interface first becomes active on a multi-access network, it sets the DR and BDR to 0.0.0.0. It also sets a *wait timer* with a value equal to the RouterDeadInterval.
- Existing interfaces on a multi-access network record the addresses of the DR and the BDR in the interface data structure, described in a later section.

The election procedure of the DR and BDR is as follows:

1. After two-way communication has been established with one or more neighbors, examine the Priority, DR, and BDR fields of each neighbor's Hello. List all routers eligible for election (that is, routers with priority greater than 0 and whose neighbor state is at least two-way); all routers declaring themselves to be the DR (their own interface address is in the DR field of the Hello packet); and all routers declaring themselves to be the BDR (their own interface address is in the BDR field of the Hello packet). The calculating router will include itself on this list unless it is ineligible.

2. From the list of eligible routers, create a subset of all routers not claiming to be the DR (routers declaring themselves to be the DR cannot be elected BDR).

3. If one or more neighbors in this subset include its own interface address in the BDR field, the neighbor with the highest priority will be declared the BDR. In a tie, the neighbor with the highest Router ID will be chosen.

4. If no router in the subset claims to be the BDR, the neighbor with the highest priority will become the BDR. In a tie, the neighbor with the highest Router ID will be chosen.

5. If one or more of the eligible routers include their own address in the DR field, the neighbor with the highest priority will be declared the DR. In a tie, the neighbor with the highest Router ID will be chosen.

6. If no router has declared itself the DR, the newly elected BDR will become the DR.

7. If the router performing the calculation is the newly elected DR or BDR, or if it is no longer the DR or BDR, repeat steps 2 through 6.

In simpler language, when an OSPF router becomes active and discovers its neighbors, it checks for an active DR and BDR. If a DR and BDR exist, the router accepts them. If there is no BDR, an election is held in which the router with the highest priority becomes the BDR. If more than one router has the same priority, the one with the numerically highest Router ID wins. If there is no active DR, the BDR is promoted to DR and a new election is held for the BDR.

It should be noted that the priority can influence an election, but will not override an active DR or BDR. That is, if a router with a higher priority becomes active after a DR and BDR have been elected, the new router will not replace either of them. So the first two DR-eligible routers to initialize on a multiaccess network will become the DR and BDR.

After the DR and BDR have been elected, the other routers (known as DRothers) will establish adjacencies with the DR and BDR only. All routers continue to multicast Hellos to the

AllSPFRouters address 224.0.0.5 so that they can track neighbors, but DRothers multicast update packets to the AllDRouters address 224.0.0.6. Only the DR and BDR will listen to this address; in turn, the DR will flood the updates to the DRothers on 224.0.0.5.

Note that if only one eligible router is attached to a multiaccess network, that router will become the DR and there will be no BDR. Any other routers will form adjacencies only with the DR. If none of the routers attached to a multi-access network are eligible, there will be no DR or BDR and no adjacencies will form. The neighbor states of all routers will remain two-way (explained later, in "Neighbor State Machine").

The duties performed by the DR and BDR are described more fully in subsequent sections.

## OSPF Interfaces

The essence of a link-state protocol is that it is concerned with links and the state of those links. Before Hellos can be sent, before adjacencies can be formed, and before LSAs can be sent, an OSPF router must understand its own links. A router's interfaces are the means by which OSPF interprets links. As a result, when speaking of OSPF, it is not uncommon to hear the terms *interface* and *link* used synonymously. This section examines the data structure OSPF associates with each interface and the various states of an OSPF interface.

## Interface Data Structure

An OSPF router maintains a data structure for each OSPF-enabled interface. In Example 8-2, the command show ip ospf interface has been used to observe the components of an interface data structure.[8]

[8] Depending on the version of IOS you are running, the output of this command might show more information than is discussed here; but this information is essential to every OSPF interface.

### Example 8-2. The OSPF-specific data related to an interface can be observed with the command *show ip ospf interface*. In this example, the interface is attached to a point-to-point network type.

```
Renoir#show ip ospf interface Serial1.738
Serial1.738 is up, line protocol is up
   Internet Address 192.168.21.21/30, Area 7
   Process ID 1, Router ID 192.168.30.70, Network Type POINT_TO_POINT, Cost: 781
   Transmit Delay is 1 sec, State POINT_TO_POINT,
   Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
      Hello due in 00:00:07
   Neighbor Count is 1, Adjacent neighbor count is 1
      Adjacent with neighbor 192.168.30.77
   Message digest authentication enabled
      Youngest key id is 10
```

The components of the interface data structure are as follows:

- IP Address and Mask This component is the configured address and mask of the interface. OSPF packets originated from this interface will have this source address. In Example 8-2, the address/mask pair is 192.168.21.21/30.
- Area ID The area to which the interface, and the network to which it is attached, belong. OSPF packets originated from this interface will have this Area ID. In Example 8-2, the area ID is 7.

- Process ID This Cisco-specific feature is not part of the open standard. Cisco routers are capable of running multiple OSPF processes and use the Process ID to distinguish them. The Process ID has no significance outside the router on which it is configured. In Example 8-2, the Process ID is 1.

- Router ID In Example 8-2, the Router ID is 192.168.30.70.

- Network Type The type of network to which the interface is connected: broadcast, point-to-point, NBMA, point-to-multipoint, or virtual link. In Example 8-2, the network type is point-to-point.[9]

  [9] Depending on the version of IOS you are running, the output of this command might show more information than is discussed here; but this information is essential to every OSPF interface.

- Cost The outgoing cost for packets transmitted from this interface. Cost is the OSPF metric, expressed as an unsigned 16-bit integer in the range of 1 to 65535. Cisco uses a default cost of $10^8$/BW, expressed in whole numbers, where BW is the configured bandwidth of the interface and $10^8$ is the *reference bandwidth*. The interface in Example 8-2 has a configured bandwidth of 128K (not shown in the example), so the cost is $10^8$/128K = 781.

The cost can be changed with the command ip ospf cost. This command is especially important when configuring Cisco routers in a multivendor environment. Another vendor, for example, might use a default cost of 1 on all interfaces (essentially making OSPF cost reflect hop counts). If all routers do not assign costs in the same manner, OSPF can route improperly, suboptimally, or in some other unexpected way.

The reference bandwidth of $10^8$ creates a problem for some modern media with bandwidths higher than 100M (such as OC-3 or above and Gigabit Ethernet). $10^8$/100M = 1, meaning that higher bandwidths calculate to a fraction of 1, which is not allowed. So any cost that is calculated to a fraction of 1 is rounded up to 1. However, this means that if your network consists of high-bandwidth links, all interfaces wind up with a cost of 1 and the calculated shortest paths become based on least router hops. To remedy this, Cisco provides the command auto-cost reference-bandwidth, which allows the default reference bandwidth to be changed.

Other components of the interface data structure are as follows:

- InfTransDelay The seconds by which LSAs exiting the interface will have their ages incremented. In Example 8-2, this is displayed as Transmit Delay and is shown to be the Cisco default, 1 second. InfTransDelay can be changed with the command ip ospf transmit-delay.

- State The functional state of the interface, which is described in the following section, "Interface State Machine."

- Router Priority This 8-bit unsigned integer in the range of 0 to 255 elects the DR and BDR. The priority is not displayed in Example 8-2 because the network type is point-to-point; no DR or BDR is elected on this network type. Example 8-3 shows another OSPF interface in the same router. This interface shows an attached network type of broadcast, so a DR and BDR are elected. The priority shown is 1, the Cisco default. The command ip ospf priority is used to change the Router Priority.

## Example 8-3. This interface is attached to a broadcast network type, and the router is the DR on this network.

```
Renoir#show ip ospf interface Ethernet0
Ethernet0 is up, line protocol is up
   Internet Address 192.168.17.73/29, Area 0
   Process ID 1, Router ID 192.168.30.70, Network Type BROADCAST, Cost: 10
   Transmit Delay is 1 sec, State DR, Priority 1
```

```
Designated Router (ID) 192.168.30.70, Interface address 192.168.17.73
Backup Designated router (ID) 192.168.30.80, Interface address 192.168.17.74
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
   Hello due in 00:00:03
Neighbor Count is 1, Adjacent neighbor count is 1
   Adjacent with neighbor 192.168.30.80 (Backup Designated Router)
Message digest authentication enabled
   Youngest key id is 10
```

- Designated Router The DR for the network to which the interface is attached is recorded both by its Router ID and by the address of the interface attached to the shared network. Note that no DR is displayed in Example 8-2; it will be displayed only for multi-access network types. In Example 8-3, the DR is 192.168.30.70. The address of its attached interface is 192.168.17.73. A look at the Router ID, the interface address, and the interface state shows that Renoir is the DR.

- Backup Designated Router The BDR for the network to which the interface is attached is also recorded both by its Router ID and by the address of the attached interface. In Example 8-3, the BDR is 192.168.30.80, and its interface address is 192.168.17.74.

- HelloInterval The period, in seconds, between transmissions of Hello packets on the interface. This period is advertised in Hello packets that are transmitted from the interface. Cisco uses a default of 10 seconds on broadcast networks and 30 seconds on non-broadcast networks, which can be changed with the command ip ospf hello-interval. Example 8-3 displays HelloInterval as Hello and shows that the default is being used.

- RouterDeadInterval The period, in seconds, that the router will wait to hear a Hello from a neighbor on the network to which the interface is connected before declaring the neighbor down. The RouterDeadInterval is advertised in Hello packets transmitted from the interface. Cisco uses a default of four times the HelloInterval; the default can be changed with the command ip ospf dead-interval. Example 8-3 displays the RouterDeadInterval as Dead and shows that the default is being used.

- Wait Timer The length of time the router will wait for a DR and BDR to be advertised in a neighbor's Hello packet before beginning a DR and BDR selection. The period of the wait timer is the RouterDeadInterval. In Example 8-2, the wait time is irrelevant because the interface is attached to a point-to-point network; no DR or BDR will be used.

- RxmtInterval The period, in seconds, the router will wait between retransmissions of OSPF packets that have not been acknowledged. Example 8-3 displays this period as retransmit and shows that the Cisco default of five seconds is being used. An interface's RxmtInterval can be changed with the command ip ospf retransmit-interval.

- Hello Timer A timer that is set to the HelloInterval. When it expires, a Hello packet is transmitted from the interface. Example 8-3 shows that the Hello timer will expire in three seconds.

- Neighboring Routers A list of all valid neighbors (neighbors whose Hellos have been seen within the past RouterDeadInterval) on the attached network. Example 8-4 shows yet another interface on the same router. Here, five neighbors are known on the network, but only two are adjacent (the Router IDs of only the adjacent neighbors are displayed). As a DRother on this network, the router has established an adjacency only with the DR and the BDR, in keeping with the DR protocol.

## Example 8-4. On this network, the router sees five neighbors but has only formed adjacencies with the DR and the BDR.

```
Renoir#show ip ospf interface Ethernet1
```

```
Ethernet1 is up, line protocol is up
   Internet Address 192.168.32.4/24, Area 78
   Process ID 1, Router ID 192.168.30.70, Network Type BROADCAST, Cost: 10
   Transmit Delay is 1 sec, State DROTHER, Priority 1
   Designated Router (ID) 192.168.30.254, Interface address 192.168.32.2
   Backup Designated router (ID) 192.168.30.80, Interface address 192.168.32.1
   Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
      Hello due in 00:00:01
   Neighbor Count is 5, Adjacent neighbor count is 2
      Adjacent with neighbor 192.168.30.80 (Backup Designated Router)
      Adjacent with neighbor 192.168.30.254 (Designated Router)
   Message digest authentication enabled
      Youngest key id is 10
```

- AuType Describes the type of authentication used on the network. The authentication type may be Null (no authentication), Simple Password, or Cryptographic (Message Digest). Example 8-4 shows that Message Digest authentication is being used. If Null authentication is used, no authentication type or key information will be displayed when show ip ospf interface is invoked.

- Authentication Key A 64-bit password if simple authentication has been enabled for the interface or a message digest key if Cryptographic authentication is used. Example 8-4 shows that the "youngest key ID" is 10. This alludes to the fact that Cryptographic authentication allows the configuration of multiple keys on an interface to ensure smooth and secure key changes.

Example 8-5 shows an interface that is connected to an NBMA network. Notice that the HelloInterval is 30 seconds, the default for NBMA, and that the RouterDeadInterval is at the default of four times the HelloInterval.

### Example 8-5. This interface is attached to a NBMA Frame Relay network and is the BDR for this network.

```
Renoir#show ip ospf interface Serial3
Serial3 is up, line protocol is up
   Internet Address 192.168.16.41/30, Area 0
   Process ID 1, Router ID 192.168.30.105, Network Type NON_BROADCAST, Cost: 64
   Transmit Delay is 1 sec, State BDR, Priority 1
   Designated Router (ID) 192.168.30.210, Interface address 192.168.16.42
   Backup Designated router (ID) 192.168.30.105, Interface address 192.168.16.41
   Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
      Hello due in 00:00:08
   Neighbor Count is 1, Adjacent neighbor count is 1
      Adjacent with neighbor 192.168.30.210 (Designated Router)
```

It is worthwhile to spend some time comparing Example 8-2 through Example 8-5. All four interfaces are on the same router, yet on each network the router performs a different role. In each case, the interface state dictates the role of the OSPF router on a network. The next section describes the various interface states and the interface state machine.

## Interface State Machine

An OSPF-enabled interface will transition through several states before it becomes fully functional. Those states are Down, Point-to-Point, Waiting, DR, Backup, DRother, and Loopback.

- Down This is the initial interface state. The interface is not functional, all interface parameters are set to their initial values, and no protocol traffic is transmitted or received on the interface.
- Point-to-Point This state is applicable only to interfaces connected to point-to-point, point-to-multipoint, and virtual link network types. When an interface transitions to this state, it is fully functional. It will begin sending Hello packets every HelloInterval and will attempt to establish an adjacency with the neighbor at the other end of the link.
- Waiting This state is applicable only to interfaces connected to broadcast and NBMA network types. When an interface transitions to this state, it will begin sending and receiving Hello packets and will set the wait timer. The router will attempt to identify the network's DR and BDR while in this state.
- DR In this state, the router is the DR on the attached network and will establish adjacencies with the other routers on the multi-access network.
- Backup In this state, the router is the BDR on the attached network, and will establish adjacencies with the other routers on the multi-access network.
- DRother In this state, the router is neither the DR nor the BDR on the attached network. It will form adjacencies only with the DR and BDR, although it will track all neighbors on the network.
- Loopback In this state, the interface is looped back via software or hardware. Although packets cannot transit an interface in this state, the interface address is still advertised in Router LSAs (described later) so that test packets can find their way to the interface.

Figure 8-3 shows the OSPF interface states and the input events that will cause a state transition. The input events are described in Table 8-1.

Figure 8-3. The OSPF interface state machine; see Table 8-1 for a description of input events (IEs).

[View full size image]

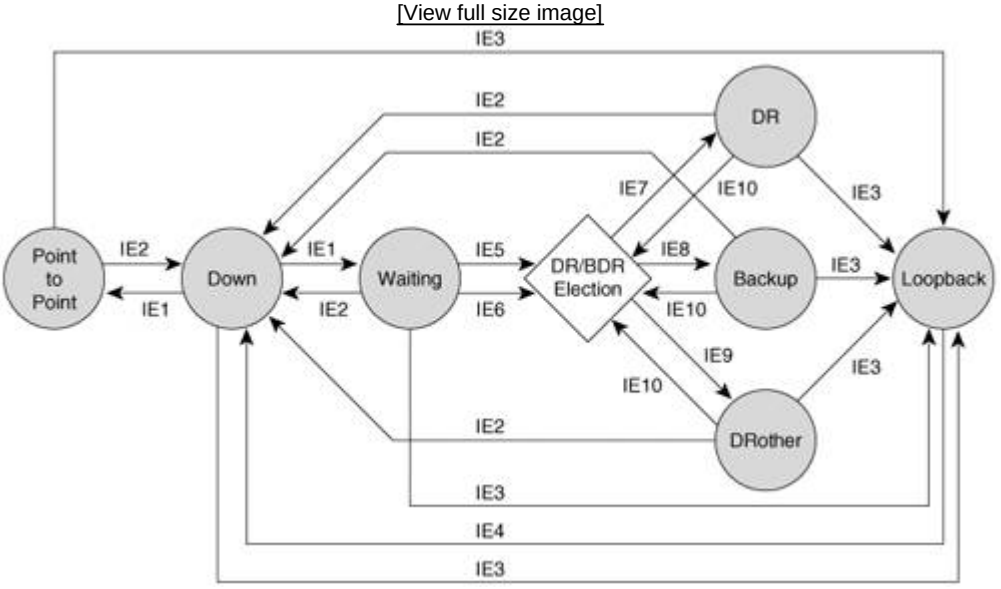



Table 8-1. Input events for the interface state machine.

| Input Event | Description |
|---|---|
| IE1 | Lower-level protocols indicate that the network interface is operational. |
| IE2 | Lower-level protocols indicate that the network interface is not operational. |
| IE3 | Network management or lower-level protocols indicate that the interface is looped up. |
| IE4 | Network management or lower-level protocols indicate that the interface is looped down. |
| IE5 | A Hello packet is received in which either the originating neighbor lists itself as the BDR or the originating neighbor lists itself as the DR and indicates no BDR. |
| IE6 | The wait timer has expired. |
| IE7 | The router is elected as the DR for this network. |
| IE8 | The router is elected as the BDR for this network. |
| IE9 | The router has not been elected as the DR or BDR for this network. |
| IE10 | A change has occurred in the set of valid neighbors on this network. This change may be one of the following:<br>1. The establishment of two-way communication with a neighbor<br>2. The loss of two-way communication with a neighbor<br>3. The receipt of a Hello in which the originating neighbor newly lists itself as the DR or BDR<br>4. The receipt of a Hello from the DR in which that router is no longer listed as the DR<br>5. The receipt of a Hello from the BDR in which that router is no longer listed as the BDR<br>6. The expiration of the RouterDeadInterval without having received a Hello from the DR or the BDR or both |

## OSPF Neighbors

The preceding section discussed a router's relationship with the attached data link. Although a router's interaction with other routers was discussed in the context of electing DRs and BDRs, the purpose of the DR election process is still to establish a relationship with a link. This section discusses a router's relationship with the neighbors on the network. The ultimate purpose of the neighbor relationship is the formation of adjacencies over which to pass routing information.

An adjacency is established in four general phases:

1. *Neighbor discovery.*
2. *Bidirectional communication.* This communication is accomplished when two neighbors list each other's Router IDs in their Hello packets.
3. *Database synchronization.* Database Description, Link State Request, Link State Update,

and Link State Acknowledgement packets (described in a later section) are exchanged to ensure that both neighbors have identical information in their link-state databases. For the purposes of this process, one neighbor will become the master and the other will become the slave. As the name implies, the master will control the exchange of Database Description packets.

4. *Full adjacency.*

As previously discussed, neighbor relationships are established and maintained through the exchange of Hello packets. On broadcast and point-to-point network types, Hellos are multicast to AllSPFRouters (224.0.0.5). On NBMA, point-to-multipoint, and virtual link network types, Hellos are unicast to individual neighbors. The implication of unicasting is that the router must first learn of the existence of its neighbors either through manual configuration or an underlying mechanism such as Inverse ARP. The configuration of neighbors on these network types is covered in the appropriate sections.

Hellos are sent every HelloInterval on every network type, with one exception: On NBMA networks, a router will send Hellos to neighbors whose neighbor state is down every PollInterval. On Cisco routers, the default PollInterval is 120 seconds.

## Neighbor Data Structure

An OSPF router builds the Hello packets for each network using the information stored in the interface data structure of the attached interface. By sending the Hello packets containing this information, the router informs neighbors about itself. Likewise, for each neighbor the router will maintain a neighbor data structure consisting of the information learned from other routers' Hello packets.

In Example 8-6, the command show ip ospf neighbor is used to observe some of the information in the neighbor data structure for a single neighbor.[10]

[10] Compare this usage with Example 8-1.

### Example 8-6. An OSPF router describes each conversation with each neighbor by a neighbor data structure.

```
Seurat#show ip ospf neighbor 10.7.0.1
 Neighbor 10.7.0.1, interface address 10.8.1.2
    In the area 0 via interface Ethernet0/0
    Neighbor priority is 1, State is FULL, 6 state changes
    DR is 10.8.1.1 BDR is 10.8.1.2
    Options is 0x52
    LLS Options is 0x1 (LR)
    Dead timer due in 00:00:30
    Neighbor is up for 09:55:04
    Index 1/3, retransmission queue length 0, number of retransmission 1
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 1, maximum is 1
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Actually, the data structure records more information about each neighbor than is shown in the example.[11] The components of the neighbor data structure are as follows:

[11] And as with **show ip ospf interface**, you might see somewhat different information than what is shown in Example 8-6 depending on what version of IOS you are running.

- Neighbor ID The router ID of the neighbor. In Example 8-6, the neighbor ID is 10.7.0.1.

- Neighbor IP Address The IP address of the neighbor's interface attached to the network. When OSPF packets are unicasted to the neighbor, this address will be the destination address. In Example 8-6, the neighbor's IP address is 10.8.1.2.

- Area ID For two routers to become neighbors, the Area ID carried in a received Hello packet must match the Area ID of the receiving interface. The Area ID of the neighbor in Example 8-6 is 0 (0.0.0.0).

- Interface The interface attached to the network on which the neighbor is located. In Example 8-6, the neighbor is reached via Ethernet0/0.

- Neighbor Priority This component is the Router Priority of the neighbor, as advertised in the neighbor's Hello packets. The priority is used in the DR/BDR election process. The neighbor in Example 8-6 has a priority of 1, the Cisco default.

- State This component is the router's view of the functional state of the neighbor, as described in the following section, "Neighbor State Machine." The state of the neighbor in Example 8-6 is Full.

- Designated Router This address is included in the DR field of the neighbor's Hello packets. The DR in Example 8-6 is 10.8.1.1.

- Backup Designated Router This address is included in the BDR field of the neighbor's Hello packets. The BDR in Example 8-6 is 10.8.1.2.

- PollInterval This value is recorded only for neighbors on NBMA networks. Because neighbors might not be automatically discovered on NBMA networks, if the neighbor state is Down, a Hello will be sent to the neighbor every PollIntervalsome period longer than the HelloInterval. The neighbor in Example 8-6 is on an NBMA network, as indicated by the default Cisco PollInterval of 120 seconds.

- Neighbor Options The optional OSPF capabilities supported by the neighbor. Options are discussed in the section describing the Hello packet format. The value of the Options field in Example 8-6 is 0x52.

- Inactivity Timer A timer whose period is the RouterDeadInterval, as defined in the interface data structure. The timer is reset whenever a Hello is received from the neighbor. If the inactivity timer expires before a Hello is heard from the neighbor, the neighbor is declared Down. In Example 8-6, the inactivity timer is shown as the Dead Timer and will expire in 30 seconds.

Components of the neighbor data structure that are not displayed by the show ip ospf neighbor command are as follows:

- Master/Slave The master/slave relationship, negotiated with neighbors in the ExStart state, establishes which neighbor will control the database synchronization.

- DD Sequence Number The Sequence Number of the Database Description (DD) packet currently being sent to the neighbor.

- Last Received Database Description Packet The Initialize, More, and Master bits; the options; and the sequence number of the last received Database Description packet are recorded. This information is used to determine whether the next DD packet is a duplicate.

- Link State Retransmission List This component is a list of LSAs that have been flooded on the adjacency, but have not yet been acknowledged. The LSAs are retransmitted every RxmtInterval, as defined in the interface data structure, until they are acknowledged or until the adjacency is destroyed. While the display in Example 8-6 does not show the LS Retransmission List, it does show the number of LSAs currently on the list ("retransmission queue length"), which is 0.

- Database Summary List This component is the list of LSAs sent to the neighbor in Database Description packets during database synchronization. These LSAs make up the link-state database when the router goes into exchange state.

- Link State Request List This list records LSAs from the neighbor's Database Description packets that are more recent than the LSAs in the link-state database. Link State Request packets are sent to the neighbor for copies of these LSAs; as the requested LSAs are received in Link State Update packets, the Request List is depleted.

## Neighbor State Machine

An OSPF router transitions a neighbor (as described in the neighbor data structure) through several states before the neighbor is considered fully adjacent:

- Down The initial state of a neighbor conversation indicates that no Hellos have been heard from the neighbor in the last RouterDeadInterval. Hellos are not sent to down neighbors unless those neighbors are on NBMA networks; in this case, Hellos are sent every PollInterval. If a neighbor transitions to the Down state from some higher state, the link state Retransmission, Database Summary, and Link State Request lists are cleared.

- Attempt This state applies only to neighbors on NBMA networks, where neighbors are manually configured. A DR-eligible router transitions a neighbor to the Attempt state when the interface to the neighbor first becomes Active or when the router is the DR or BDR. A router sends packets to a neighbor in Attempt state at the HelloInterval instead of the PollInterval.

- Init This state indicates that a Hello packet has been seen from the neighbor in the last RouterDeadInterval, but two-way communication has not yet been established. A router includes the Router IDs of all neighbors in this state or higher in the Neighbor field of the Hello packets.

- 2-Way This state indicates that the router has seen its own Router ID in the Neighbor field of the neighbor's Hello packets, which means that a bidirectional conversation has been established. On multi-access networks, neighbors must be in this state or higher to be eligible to be elected as the DR or BDR. The reception of a Database Description packet from a neighbor in the init state also causes a transition to 2-Way.

- ExStart In this state, the router and its neighbor establish a master/slave relationship and determine the initial DD sequence number in preparation for the exchange of Database Description packets. The neighbor with the highest Router ID becomes the master.

- Exchange The router sends Database Description packets describing its entire link-state database to neighbors that are in the Exchange state. The router may also send Link State Request packets, requesting more recent LSAs, to neighbors in this state.

- Loading The router sends Link State Request packets to neighbors that are in the Loading state, requesting more recent LSAs that have been discovered in the Exchange state but have not yet been received.

- Full Neighbors in this state are fully adjacent, and the adjacencies appear in Router LSAs and Network LSAs.

Figure 8-4 through Figure 8-6 show the OSPF neighbor states and the input events that cause a state transition. The input events are described in Table 8-2, and the decision points are defined in Table 8-3. Figure 8-4 shows the normal progression from the least functional state to the fully functional state, and Figure 8-5 and Figure 8-6 show the complete OSPF neighbor state machine.

Figure 8-4. The normal series of transitions in the OSPF neighbor state

machine that take a neighbor from Down to Full.

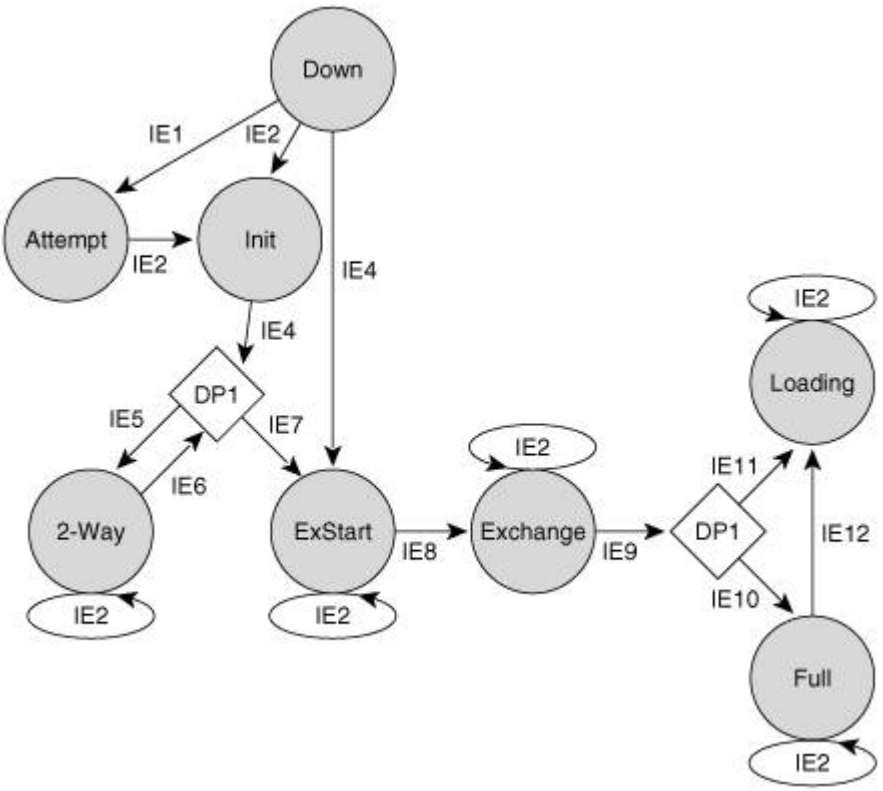



Figure 8-5. The neighbor state machine, from Down to Init.

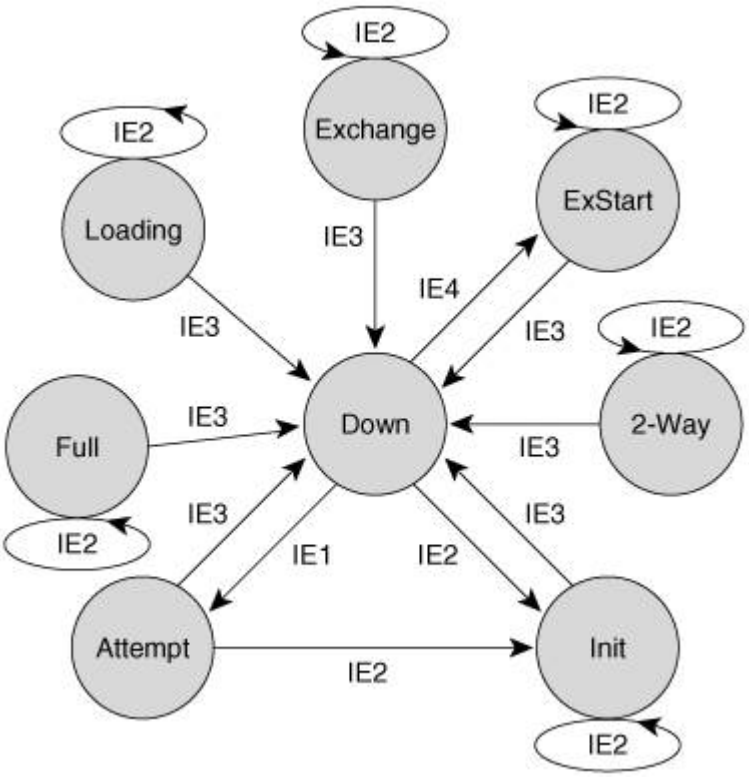


Figure 8-6. The neighbor state machine, from Init to Full.

[View full size image]

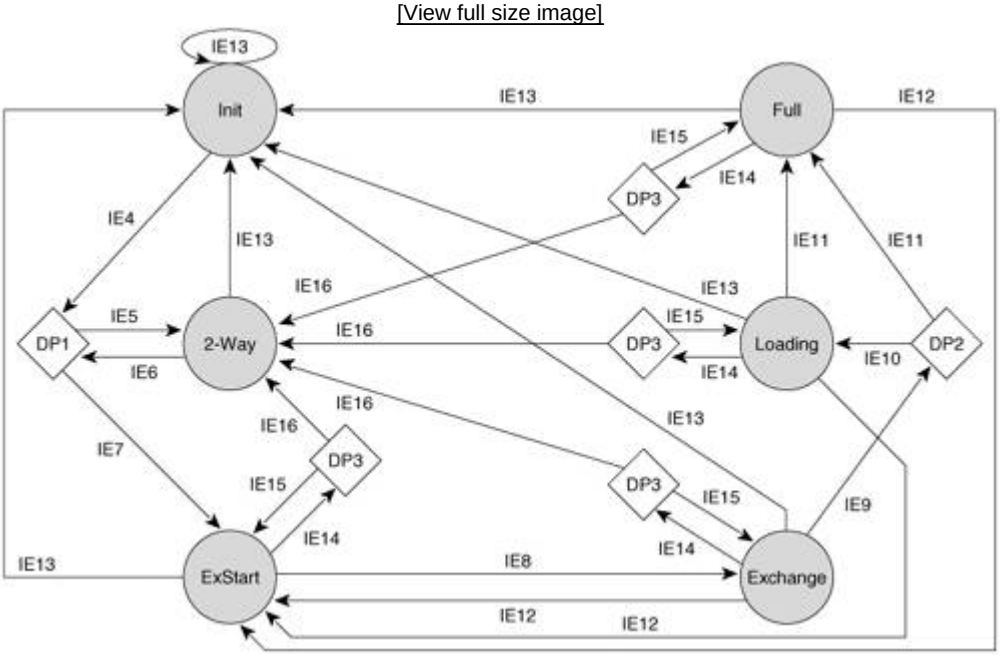



Table 8-2. Input events for Figure 8-4 through Figure 8-6.

| Input Event | Description |
|---|---|
| IE1 | This event occurs only for NBMA-connected neighbors. The input event will be triggered under either of the following conditions:<br><br>1. The interface to the NBMA network first becomes active, and the neighbor is eligible for DR election.<br><br>2. The router becomes either DR or BDR, and the neighbor is not eligible for DR election. |
| IE2 | A valid Hello packet has been received from the neighbor. |
| IE3 | The neighbor is no longer reachable, as determined by the lower level protocols, by an explicit instruction from the OSPF process itself, or by the expiration of the inactivity timer. |
| IE4 | The router first sees its own Router ID listed in the Neighbor field of the neighbor's Hello packet or receives a Database Description packet from the neighbor. |
| IE5 | The neighbor should not become adjacent. |
| IE6 | This input event occurs under either of the following conditions:<br><br>(1) The neighbor state first transitions to 2-Way.<br><br>(2) The interface state changes. |
| IE7 | An adjacency should be formed with this neighbor. |

| Input Event | Description |
|---|---|
| IE8 | The master/slave relationship has been established and DD sequence numbers have been exchanged. |
| IE9 | The exchange of Database Description packets has been completed. |
| IE10 | Entries exist in the Link State Request list. |
| IE11 | The Link State Request list is empty. |
| IE12 | The adjacency should be broken and then restarted. This input event might be triggered by any of the following:<br><br>(1) The reception of a Database Description packet with an unexpected DD sequence number<br><br>(2) The reception of a Database Description packet with the Options field set differently than the Options field of the last DD packet<br><br>(3) The reception of a Database Description packet, other than the first packet, in which the Init bit is set<br><br>(4) The reception of a Link State Request packet for an LSA that is not in the database |
| IE13 | A Hello packet has been received from the neighbor in which the receiving router's Router ID is not listed in the Neighbor field. |
| IE14 | This event occurs when the interface state changes. |
| IE15 | The existing or forming adjacency with this neighbor should continue. |
| IE16 | The existing or forming adjacency with this neighbor should not continue. |

Table 8-3. Decision points for Figure 8-4 and Figure 8-6.

| Decision | Description |
|---|---|
| DP1 | Should an adjacency be established with the neighbor? An adjacency should be formed if one or more of the following conditions is true:<br><br>(1) The network type is point-to-point.<br><br>(2) The network type is point-to-multipoint.<br><br>(3) The network type is virtual link.<br><br>(4) The router is the DR for the network on which the neighbor is located.<br><br>(5) The router is the BDR for the network on which the neighbor is located.<br><br>(6) The neighbor is the DR.<br><br>(7) The neighbor is the BDR. |
| DP2 | Is the Link State Request list for this neighbor empty? |
| DP3 | Should the existing or forming adjacency with the neighbor continue? |

## Building an Adjacency

Neighbors on point-to-point, point-to-multipoint, and virtual link networks always become adjacent unless the parameters of their Hellos don't match. On broadcast and NBMA networks, the DR and BDR become adjacent with all neighbors, but no adjacencies exist between DRothers.

The adjacency building process uses three OSPF packet types:

- Database Description packets (type 2)
- Link State Request packets (type 3)
- Link State Update packets (type 4)

The formats of these packet types are described in detail in a subsequent section, "OSPF Packet Formats."

The Database Description packet is of particular importance to the adjacency-building process. As the name implies, the packets carry a summary description of each LSA in the originating router's link-state database. These descriptions are not the complete LSAs, but merely their headersenough information for the receiving router to decide whether it has the latest copy of the LSA in its own database. In addition, three flags in the DD packet are used to manage the adjacency building process:

- The I-bit, or Initial bit, which when set indicates the first DD packet sent
- The M-bit, or More bit, which when set indicates that this is not the last DD packet to be sent
- The MS-bit, or Master/Slave bit, which is set in the DD packets originated by the master

When the master/slave negotiation begins in the ExStart state, both neighbors will claim to be the master by sending an empty DD packet with the MS-bit set to one. The DD sequence number in these two packets will be set to the originating router's idea of what the sequence number should be. The neighbor with the lower Router ID will become the slave and will reply with a DD packet in which the MS-bit is zero and the DD sequence number is set to the master's sequence number. This DD packet will be the first packet populated with LSA summaries. When the master/slave negotiation is completed, the neighbor state transitions to Exchange.

In the Exchange state, the neighbors synchronize their link-state databases by describing all entries in their respective link-state databases. The Database Summary List is populated with the headers of all LSAs in the router's database; Database Description packets containing the listed LSA headers are sent to the neighbor.

If either router sees that its neighbor has an LSA that is not in its own database, or that the neighbor has a more recent copy of a known LSA, it places the LSA on the Link State Request list. It then sends a Link State Request packet asking for a complete copy of the LSA in question. Link State Update packets convey the requested LSAs. As the requested LSAs are received, they are removed from the Link State Request list.

All LSAs sent in Update packets must be individually acknowledged. Therefore, the transmitted LSAs are entered into the Link State Retransmission list. As they are acknowledged, they are removed from the list. The LSA may be acknowledged by one of two means:

- Explicit Acknowledgment A Link State Acknowledgment packet containing the LSA header is received.
- Implicit Acknowledgment An Update packet that contains the same instance of the LSA (neither LSA is more recent than the other) is received.

The master controls the synchronization process and ensures that only one DD packet is

outstanding at a time. When the slave receives a DD packet from the master, the slave acknowledges the packet by sending a DD packet with the same sequence number. If the master does not receive an acknowledgment of an outstanding DD packet within the RxmtInterval, as specified in the interface data structure, the master sends a new copy of the packet.

The slave sends DD packets only in response to DD packets it receives from the master. If the received DD packet has a new sequence number, the slave sends a DD packet with the same sequence number. If the received sequence number is the same as a previously acknowledged DD packet, the acknowledging packet is re-sent.

When the database synchronization process is complete, one of two state transitions will occur:

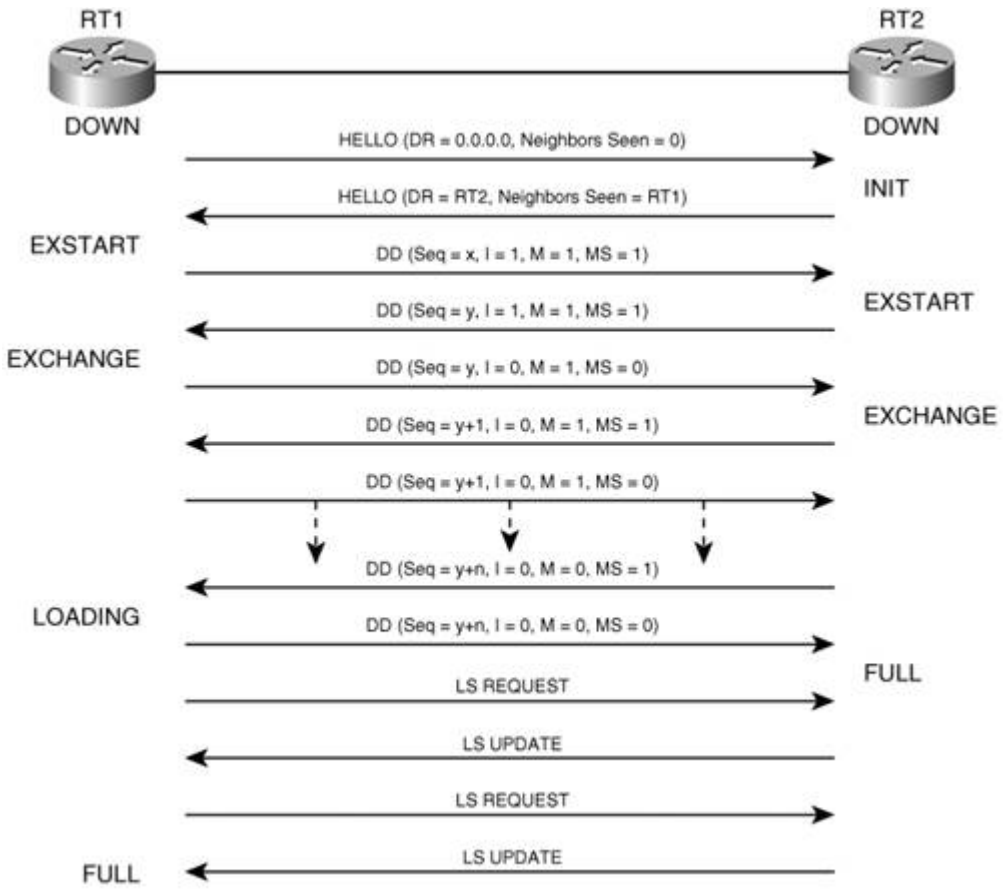
- If there are still entries of the Link State Request list, the router transitions the state of the neighbor to Loading.
- If the Link State Request list is empty, the router transitions the state of the neighbor to Full.

The master knows that the synchronization process is complete when it has sent all the DD packets necessary to fully describe its link-state database and has received a DD packet with the M-bit set to zero. The slave knows that the process is complete when it receives a DD packet with the M-bit set to zero and sends an acknowledging DD packet that also has its M-bit set to zero (that is, the slave has fully described its own database). Because the slave must acknowledge each received DD packet, the slave will always be the first to know that the synchronization process is complete.

Figure 8-7 shows the adjacency-building process. This example is taken directly from RFC 2328.

Figure 8-7. The link-state database synchronization process and associated neighbor states.

[View full size image]




The following steps are illustrated in Figure 8-7:

1. RT1 becomes active on the multi-access network and sends a Hello packet. It has not yet heard from any neighbors, so the Neighbor field of the packet is empty, and the DR and BDR fields are set to 0.0.0.0.

2. Upon reception of the Hello from RT1, RT2 creates a neighbor data structure for RT1 and sets RT1's state to Init. RT2 sends a Hello packet with RT1's Router ID in the Neighbor field; as the DR, RT2 also includes its own interface address in the DR field.

3. Seeing its Router ID in the received Hello packet (IE 4 in Table 8-2), RT1 creates a neighbor data structure for RT2 and sets RT2's state to ExStart for the master/slave negotiation. It then generates an empty (no LSA summaries) Database Description packet; the DD sequence number is set to x, the I-bit is set to indicate that this is RT1's initial DD packet for this exchange, the M-bit is set to indicate that this is not the last DD packet, and the MS-bit is set to indicate that RT1 is asserting itself as the master.

4. RT2 transitions RT1's state to ExStart upon reception of the DD packet. It then sends a responding DD packet with a DD sequence number of y; RT2 has a higher router ID than RT1, so it sets the MS-bit. Like the first DD packet, this one is used for the master/slave negotiation and therefore is empty.

5. Agreeing that RT2 is the master, RT1 transitions RT2's state to Exchange. RT1 will generate a DD packet with RT2's DD sequence number of y and the MS = 0, indicating that RT1 is the slave. This packet will be populated with LSA headers from RT1's Link State Summary list.

6. RT2 transitions its neighbor state to Exchange upon receipt of RT1's DD packet. It will send a DD packet containing LSA headers from its Link State Summary list and will increment the DD sequence number to y + 1.

7. RT1 sends an acknowledging packet containing the same sequence number as in the DD packet that it just received from RT2. The process continues, with RT2 sending a single DD packet and then waiting for an acknowledging packet from RT1 containing the same sequence number before sending the next packet. When RT2 sends the DD packet with the last of its LSA summaries, it sets M = 0.

8. Receiving this packet and knowing that the acknowledging packet it will send contains the last of its own LSA summaries, RT1 knows the Exchange process is done. However, it has entries in its Link State Request list; therefore, it will transition to Loading.

9. When RT2 receives RT1's last DD packet, RT2 transitions RT1's state to Full because it has no entries in its Link State Request list.

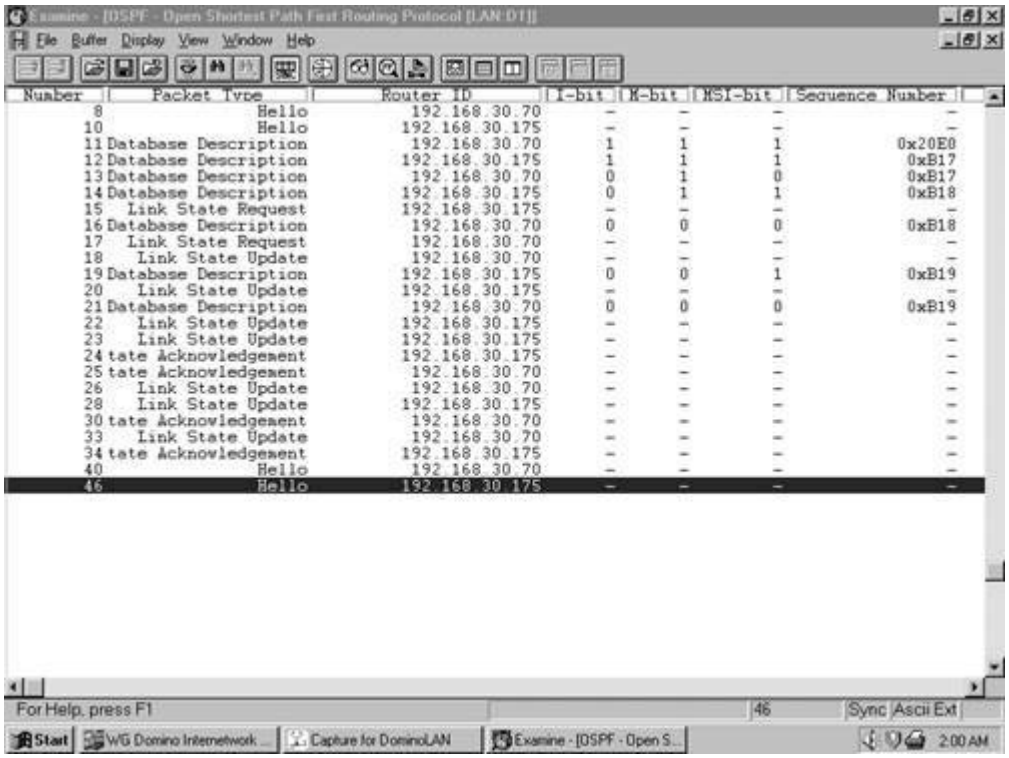
10. RT1 sends Link State Request packets, and RT2 sends the requested LSAs in Link State Update packets, until RT1's Link State Request list is empty. RT1 will then transition RT2's state to Full.

Note that if either router has entries in its Link State Request list, it does not need to wait for the Loading state to send Link State Request packets; it may do so while the neighbor is still in the Exchange state. Consequently, the synchronization process is not as tidy as depicted in Figure 8-7, but it is more efficient.

Figure 8-8 shows an analyzer capture of an adjacency being built between two routers. Although Link State Request and Link State Update packets are being sent while both neighbors are still in the Exchange state, attention to the I-, M-, and MS-bits and the sequence numbers reveals that the real-life process follows the generic procedure of Figure 8-7.

Figure 8-8. This analyzer capture shows an adjacency being built.

[View full size image]



| Number | Packet Type | Router ID | I-bit | M-bit | MSI-bit | Sequence Number |
|---|---|---|---|---|---|---|
| 8 | Hello | 192.168.30.70 | - | - | - | - |
| 10 | Hello | 192.168.30.175 | - | - | - | - |
| 11 | Database Description | 192.168.30.70 | 1 | 1 | 1 | 0x20E0 |
| 12 | Database Description | 192.168.30.175 | 1 | 1 | 1 | 0xB17 |
| 13 | Database Description | 192.168.30.70 | 0 | 1 | 0 | 0xB17 |
| 14 | Database Description | 192.168.30.175 | 0 | 1 | 1 | 0xB18 |
| 15 | Link State Request | 192.168.30.175 | - | - | - | - |
| 16 | Database Description | 192.168.30.70 | 0 | 0 | 0 | 0xB18 |
| 17 | Link State Request | 192.168.30.70 | - | - | - | - |
| 18 | Link State Update | 192.168.30.70 | - | - | - | - |
| 19 | Database Description | 192.168.30.175 | 0 | 0 | 1 | 0xB19 |
| 20 | Link State Update | 192.168.30.175 | - | - | - | - |
| 21 | Database Description | 192.168.30.70 | 0 | 0 | 0 | 0xB19 |
| 22 | Link State Update | 192.168.30.175 | - | - | - | - |
| 23 | Link State Update | 192.168.30.175 | - | - | - | - |
| 24 | tate Acknowledgement | 192.168.30.175 | - | - | - | - |
| 25 | tate Acknowledgement | 192.168.30.70 | - | - | - | - |
| 26 | Link State Update | 192.168.30.70 | - | - | - | - |
| 28 | Link State Update | 192.168.30.175 | - | - | - | - |
| 30 | tate Acknowledgement | 192.168.30.70 | - | - | - | - |
| 33 | Link State Update | 192.168.30.70 | - | - | - | - |
| 34 | tate Acknowledgement | 192.168.30.175 | - | - | - | - |
| 40 | Hello | 192.168.30.70 | - | - | - | - |
| 46 | Hello | 192.168.30.175 | - | - | - | - |



In Example 8-7, the output of the command `debug ip ospf adj` shows the adjacency of Figure 8-8 being built from the perspective of one of the routers (router ID 192.168.30.175).

## Example 8-7. This debug output shows the adjacency events of Figure 8-8 from the perspective of one of the routers.

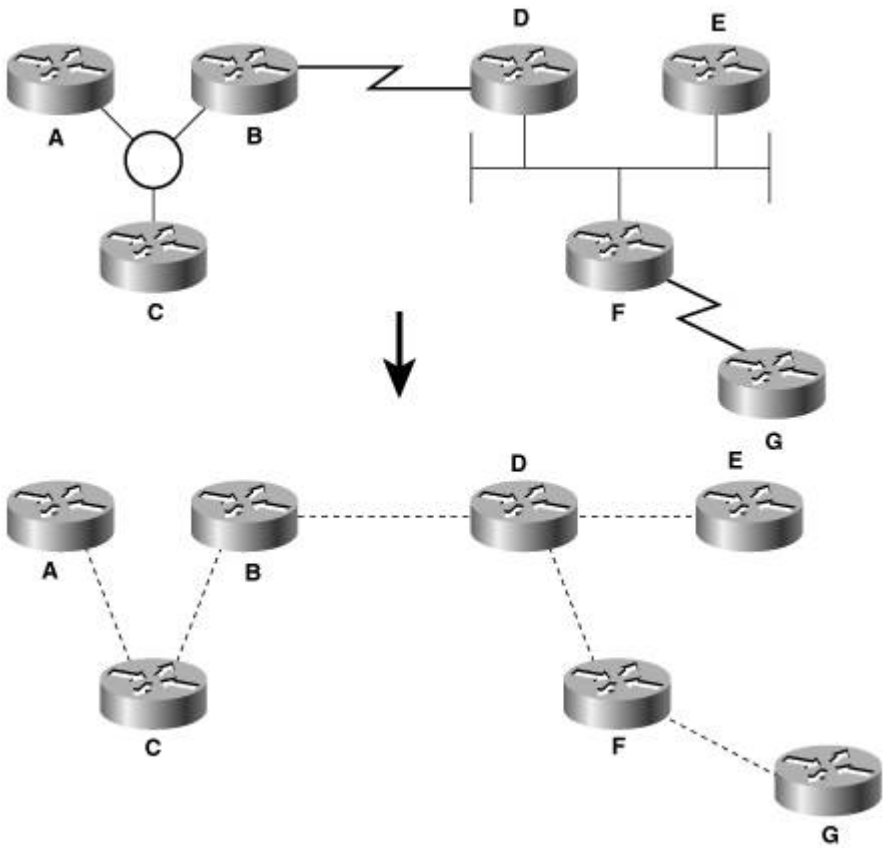
```
Degas#debug ip ospf adj
OSPF adjacency events debugging is on
OSPF: Rcv DBD from 192.168.30.70 on Ethernet0 seq 0x20E0 opt 0x2 flag 0x7 len 32
state INIT
OSPF: 2 Way Communication to 192.168.30.70 on Ethernet0,
  state 2WAY
OSPF: Neighbor change Event on interface Ethernet0
OSPF: DR/BDR election on Ethernet0
OSPF: Elect BDR 192.168.30.70
OSPF: Elect DR 192.168.30.175
   DR: 192.168.30.175 (Id) BDR: 192.168.30.70 (Id)
OSPF: Send DBD to 192.168.30.70 on Ethernet0 seq 0xB17 opt 0x2 flag 0x7 len 32
OSPF: First DBD and we are not SLAVE
OSPF: Rcv DBD from 192.168.30.70 on Ethernet0 seq 0xB17 opt 0x2 flag 0x2 len 92
state EXSTART
OSPF: NBR Negotiation Done. We are the MASTER
OSPF: Send DBD to 192.168.30.70 on Ethernet0 seq 0xB18 opt 0x2 flag 0x3 len 72
OSPF: Database request to 192.168.30.70
OSPF: Rcv DBD from 192.168.30.70 on Ethernet0 seq 0xB18 opt 0x2 flag 0x0 len 32
state EXCHANGE
OSPF: Send DBD to 192.168.30.70 on Ethernet0 seq 0xB19 opt 0x2 flag 0x1 len 32
OSPF: Rcv DBD from 192.168.30.70 on Ethernet0 seq 0xB19 opt 0x2 flag 0x0 len 32
state EXCHANGE
OSPF: Exchange Done with 192.168.30.70 on Ethernet0
OSPF: Synchronized with 192.168.30.70 on Ethernet0,
state FULL
```

At the end of the synchronization process in Figure 8-8, a series of Link State Update and Link State Acknowledgment packets can be observed. These are part of the LSA flooding process, discussed in the next section.

## Flooding

The entire OSPF topology can be depicted as a group of routers, or nodes, interconnected not by physical links but by logical adjacencies (Figure 8-9). For the nodes to route properly over this logical topology, each node must possess an identical map of the topology. This map is the topological database.

Figure 8-9. A group of routers interconnected by data links will be viewed by OSPF as a group of nodes interconnected by adjacencies.




The OSPF topological database is better known as the link-state database. This database consists of all the LSAs the router has received. A change in the topology is represented as a change in one or more of the LSAs. Flooding is the process by which these changed or new LSAs are sent throughout the network, to ensure that the database of every node is updated and remains identical to all other nodes' databases.

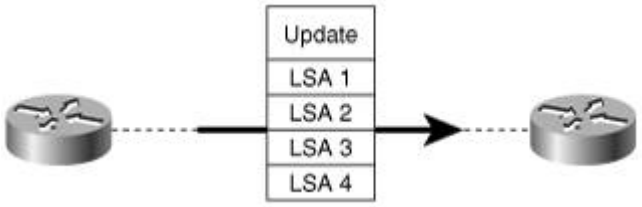
Flooding makes use of the following two OSPF packet types:

- Link State Update packets (type 4)
- Link State Acknowledgment packets (type 5)

As Figure 8-10 shows, each Link State Update and Acknowledgment packet may carry multiple

LSAs. Although the LSAs themselves are flooded throughout the area, the Update and Acknowledgment packets travel only between two nodes across an adjacency.

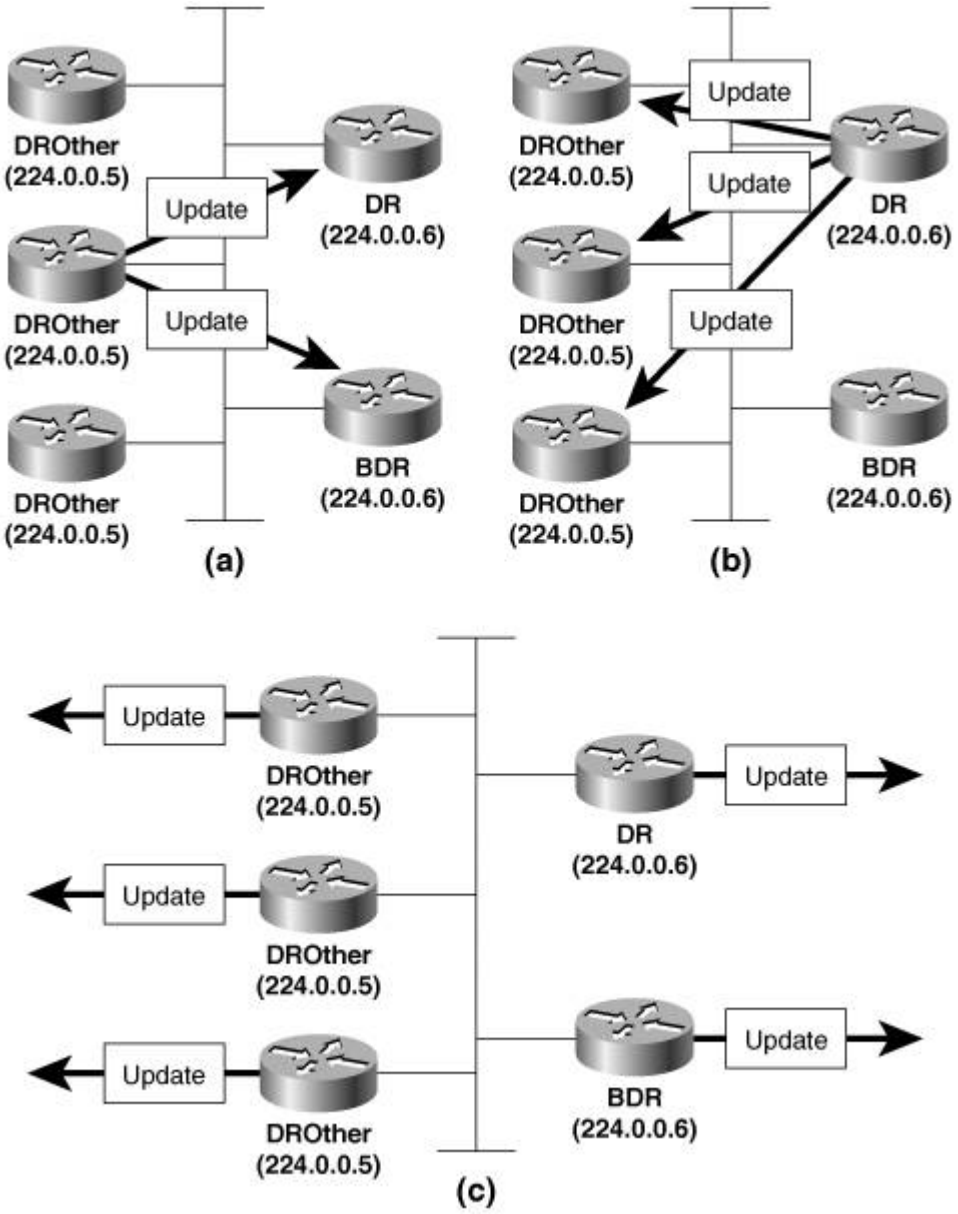
Figure 8-10. LSAs are sent across adjacencies within link-state update packets.




On point-to-point networks, updates are sent to the multicast address AllSPFRouters (224.0.0.5). On point-to-multipoint and virtual link networks, updates are unicasted to the interface addresses of the adjacent neighbors.

On broadcast networks, DRothers form adjacencies only with the DR and BDR. Therefore, updates are sent to the address AllDRouters (224.0.0.6). The DR in turn multicasts an Update packet containing the LSA to all adjacent routers on the network using the address AllSPFRouters. All routers then flood the LSA out all other interfaces (Figure 8-11). Although the BDR hears and records LSAs multicast from DRothers, it does not reflood or acknowledge them unless the DR fails to do so. The same DR/BDR functionality exists on NBMA networks, except that LSAs are unicast from DRothers to the DR and BDR, and the DR unicasts a copy of the LSA to all adjacent neighbors.

Figure 8-11. On a broadcast network, a DRother sends an LSA only to the DR and BDR (a); the DR refloods the LSA to all adjacent neighbors (b); all routers then flood the LSA on all other interfaces (c).




Because identical link-state databases are essential to correct OSPF operation, flooding must be reliable. Transmitting routers must know that their LSAs were received successfully, and receiving routers must know that they are accepting the correct LSAs.

## Reliable Flooding: Acknowledgments

Each individual transmitted LSA must be acknowledged. This may be accomplished by either an *implicit* acknowledgment or an *explicit* acknowledgment.

A neighbor can implicitly acknowledge the receipt of an LSA by including a duplicate of the LSA in an update back to the originator. Implicit acknowledgments are more efficient than explicit acknowledgments in some situations, such as when the neighbor was intending to send an update to the originator anyway.

A neighbor explicitly acknowledges the receipt of an LSA by sending a Link State Acknowledgment packet. A single Link State Acknowledgment packet is capable of acknowledging multiple LSAs. The packet carries only LSA headersenough to completely identify the LSAnot the complete LSA.

When a router first sends an LSA, a copy of the LSA is entered into the Link State Retransmission list of every neighbor to which it was sent. The LSA is retransmitted every RxmtInterval until it is acknowledged or until the adjacency is broken. The Link State Update packets containing retransmissions are always unicast, regardless of the network type.

Acknowledgments might be either *delayed* or *direct*. By delaying an acknowledgment, more LSAs can be acknowledged in a single Link State Acknowledgment packet; on a broadcast network, LSAs from multiple neighbors can be acknowledged in a single multicast Link State Acknowledgment packet. The period by which an acknowledgment is delayed must be less than the RxmtInterval to prevent unnecessary retransmissions. Under normal circumstances, the unicast/multicast addressing conventions used for Link State Update packets on various network types also apply to Link State Acknowledgments.

Direct acknowledgments are always sent immediately and are always unicast. Direct acknowledgments are sent whenever the following conditions occur:

- A duplicate LSA is received from a neighbor, possibly indicating that it has not yet received an acknowledgment.
- The LSA's age is MaxAge (described in the next section), and there is no instance of the LSA in the receiving router's link-state database.

## Reliable Flooding: Sequencing, Checksums, and Aging

Each LSA contains three values that are used to ensure that the most recent copy of the LSA exists in every database. These values are sequence number, checksum, and age.

OSPF uses a 32-bit signed, linear sequence number space (discussed in Chapter 4, "Dynamic Routing Protocols") ranging from InitialSequenceNumber (0x80000001) to MaxSequenceNumber (0x7fffffff). When a router originates an LSA, the router sets the LSA's sequence number to InitialSequenceNumber. Each time the router produces a new instance of the LSA, the router increments the sequence number by one.

If the present sequence number is MaxSequenceNumber and a new instance of the LSA must be created, the router must first flush the old LSA from all databases. This is done by setting the age of the existing LSA to MaxAge (defined later in this section) and reflooding it over all adjacencies. As soon as all adjacent neighbors have acknowledged the prematurely aged LSA, the new instance of the LSA with a sequence number of InitialSequenceNumber may be flooded.

The checksum is a 16-bit integer calculated using a Fletcher algorithm.[12] The checksum is calculated over the entire LSA with the exception of the Age field (which changes as the LSA passes from node to node and would therefore require recalculation of the checksum at each node). The checksum of each LSA is also verified every five minutes as it resides in the link-state database, to ensure that it has not been corrupted in the database.

[12] Alex McKenzie, "ISO Transport Protocol Specification ISO DP 8073," RFC 905, April 1984, Annex B.

The age is an unsigned 16-bit integer that indicates the age of the LSA in seconds. The range is 0 to 3600 (one hour, known as MaxAge). When a router originates an LSA, the router sets the age to 0. As the flooded LSA transits a router, the age is incremented by a number of seconds specified by InfTransDelay. Cisco routers have a default InfTransDelay of one second, which can be changed with the command `ip ospf transmit-delay`. The age is also incremented as it resides in the database.

When an LSA reaches MaxAge, the LSA is reflooded and then flushed from the database. When a router needs to flush an LSA from all databases, it prematurely sets the age to MaxAge and refloods it. Only the router that originated the LSA can prematurely age it.

Example 8-8 shows a portion of a link-state database; the age, sequence number, and checksum of each LSA can be observed. More detailed discussion of the database and the various LSA

types is in "Link-State Database," later in this chapter.

### Example 8-8. The age, sequence number, and checksum for each LSA are recorded in the link-state database. The age is incremented in seconds.

```
Manet#show ip ospf database
       OSPF Router with ID (192.168.30.43) (Process ID 1)
                   Router Link States (Area 3)
Link ID         ADV Router      Age   Seq#        Checksum Link Count
192.168.30.13   192.168.30.13   910   0x80000F29  0xA94E   2
192.168.30.23   192.168.30.23   1334  0x80000F55  0x8D53   3
192.168.30.30   192.168.30.30   327   0x800011CA  0x523    8
192.168.30.33   192.168.30.33   70    0x80000AF4  0x94DD   3
192.168.30.43   192.168.30.43   1697  0x80000F2F  0x1DA1   2
```

When multiple instances of the same LSA are received, a router determines which is the most recent by the following algorithm:

1. Compare the sequence numbers. The LSA with the highest sequence number is more recent.
2. If the sequence numbers are equal, then compare the checksums. The LSA with the highest unsigned checksum is the more recent.
3. If the checksums are equal, then compare the age. If only one of the LSAs has an age of MaxAge (3600 seconds), it is considered the more recent.
4. If the ages of the LSAs differ by more than 15 minutes (known as MaxAgeDiff), the LSA with the lower age is more recent.
5. If none of the preceding conditions are met, the two LSAs are considered identical.

## Areas

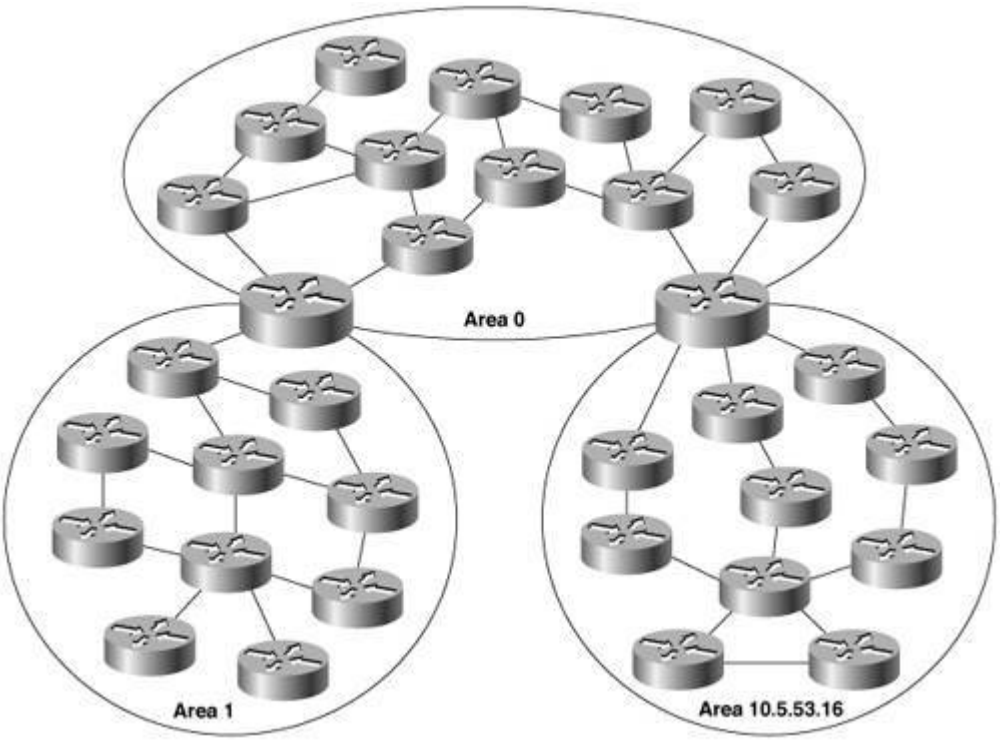
The reader should by now have a good feel for why OSPF, with its multiple databases and complex algorithms, can put greater demands on the memory and processors of a router than the previously examined protocols can. As a network grows, these demands can become significant or even crippling. And although flooding is more efficient than the periodic, full-table updates of RIP, it can still place an unacceptable burden on the data links of a large network. Contrary to popular belief, the SPF algorithm itself is not particularly processor-intensive. Rather, the related processes, such as flooding and database maintenance, burden the CPU.

OSPF uses areas to reduce these adverse effects. In the context of OSPF, an *area* is a logical grouping of OSPF routers and links that effectively divide an OSPF domain into sub-domains (Figure 8-12). Routers within an area will have no detailed knowledge of the topology outside of their area. Because of this condition

- A router must share an identical link-state database only with the other routers in its area, not with the entire OSPF domain. The reduced size of the database reduces the impact on a router's memory.
- The smaller link-state databases mean fewer LSAs to process and therefore less impact on the CPU.

- Because the link-state database must be maintained only within an area, most flooding is also limited to the area.

Figure 8-12. An OSPF area is a logical grouping of OSPF routers. Each area is described by its own link-state database, and each router must maintain a database only for the area to which it belongs.




Areas are identified by a 32-bit *Area ID*. As Figure 8-12 shows, the Area ID may be expressed either as a decimal number or in dotted decimal, and the two formats may be used together on Cisco routers. The choice usually depends on which format is more convenient for identifying the particular Area ID. For example, area 0 and area 0.0.0.0 are equivalent, as are area 16 and area 0.0.0.16, and area 271 and area 0.0.1.15. In each of these cases, the decimal format would probably be preferred. However, given the choice of area 3232243229 and area 192.168.30.29, the latter would probably be chosen.

Three types of traffic may be defined in relation to areas:

- Intra-area traffic consists of packets that are passed between routers within a single area.
- Inter-area traffic consists of packets that are passed between routers in different areas.
- External traffic consists of packets that are passed between a router within the OSPF domain and a router within another routing domain.

Area ID 0 (or 0.0.0.0) is reserved for the backbone. The *backbone* is responsible for summarizing the topologies of each area to every other area. For this reason, all inter-area traffic must pass through the backbone; non-backbone areas cannot exchange packets directly.

Many OSPF designers have a favorite rule of thumb concerning the maximum number of routers that an area can handle. This number might range from 30 to 200. However, the number of routers has little actual bearing on the maximum size of an area. Far more important factors include the number of links in an area, the stability of the topology, the memory and horsepower

of the routers, the use of summarization, and the number of summary LSAs entering the area. Because of these factors, 25 routers might be too many for some areas, and other areas might accommodate well over 500 routers.

It is perfectly reasonable to design a small OSPF network with only a single area. Regardless of number of areas, a potential problem arises when an area is so underpopulated that no redundant links exist within it. If such an area becomes partitioned, service disruptions might occur. Partitioned areas are discussed in more detail in a later section.

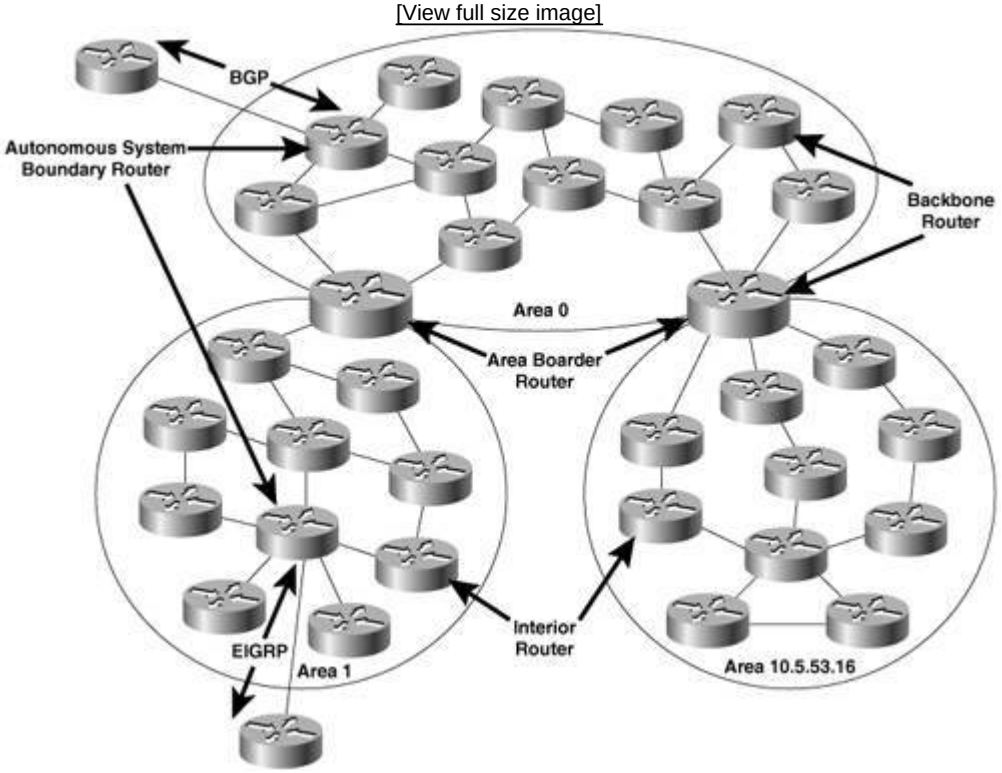
## Router Types

Routers, like traffic, can be categorized in relation to areas. All OSPF routers will be one of four router types, as shown in Figure 8-13:

- Internal Routers are routers whose interfaces all belong to the same area. These routers have a single link-state database.
- Area Border Routers (ABRs) connect one or more areas to the backbone and act as a gateway for inter-area traffic. An ABR always has at least one interface that belongs to the backbone, and must maintain a separate link-state database for each of its connected areas. For this reason, ABRs often have more memory and perhaps more powerful processors than internal routers. An ABR summarizes the topological information of its attached areas into the backbone, which then propagates the summary information to the other areas.

Figure 8-13. All OSPF routers can be classified as an Internal Router, a Backbone Router, an Area Border Router (ABR), or an Autonomous System Boundary Router (ASBR). Note that any of the first three router types might also be an ASBR.

[View full size image]



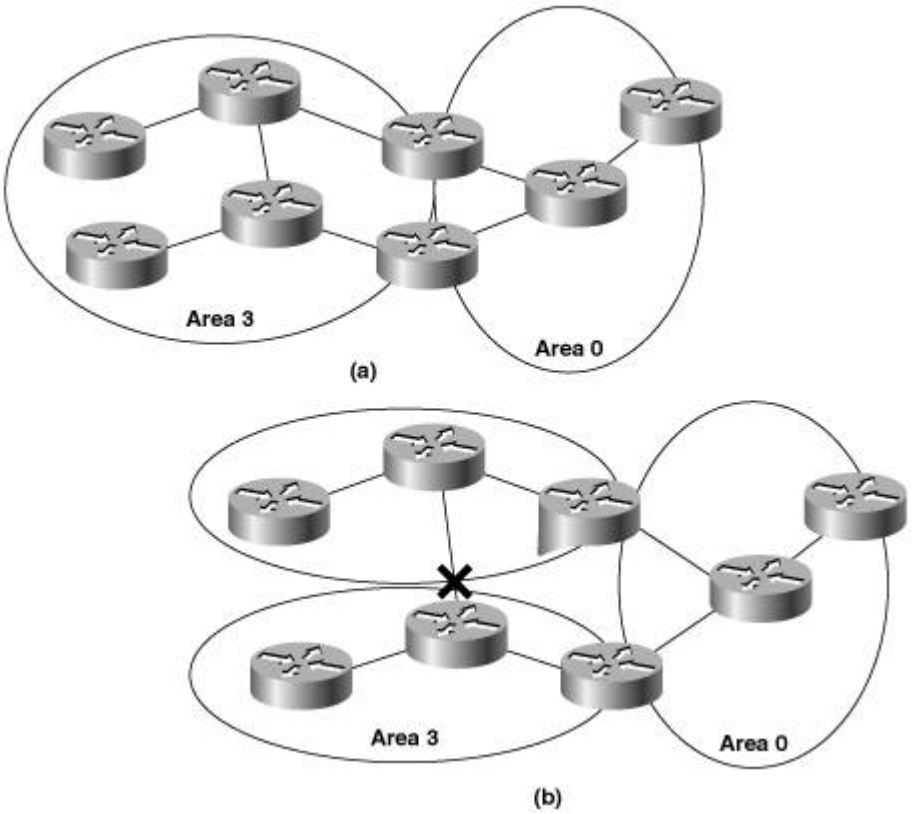
- Backbone Routers are routers with at least one interface attached to the backbone. Although this requirement means that ABRs are also Backbone Routers, Figure 8-13 shows that not all Backbone Routers are ABRs. An Internal Router whose interfaces all belong to area 0 is also a Backbone Router.

- Autonomous System Boundary Routers (ASBRs) are gateways for external traffic, injecting routes into the OSPF domain that were learned (redistributed) from some other protocol, such as the BGP and EIGRP processes shown in Figure 8-13. An ASBR can be located anywhere within the OSPF autonomous system except within stub areas; it may be an Internal, Backbone, or ABR.

## Partitioned Areas

A *partitioned area* is an area in which a link failure causes one part of the area to become isolated from another. If a non-backbone area becomes partitioned and if all routers on either side of the partition can still find an ABR, as in Figure 8-14, no service disruptions will occur. The backbone merely treats the partitioned area as two separate areas. Intra-area traffic from one side of the partition to the other side will become inter-area traffic, passing through the backbone to circumvent the partition. Note that a partitioned area is not the same as an *isolated* area, in which no path exists to the rest of the OSPF domain.

Figure 8-14. (a) Area 3 is connected to the backbone (area 0) by two ABRs. (b) A link failure in area 3 creates a partitioned area, but all routers within area 3 can still reach an ABR. In these circumstances, traffic can still be routed between the two sides of the partitioned area.




A partition of the backbone itself is a more serious matter. As Figure 8-15 shows, a partitioned

backbone area isolates the areas on each side of the partition, creating two separate OSPF domains.

Figure 8-15. If a backbone becomes partitioned, each side of the partition and any connected areas become isolated from the other side.

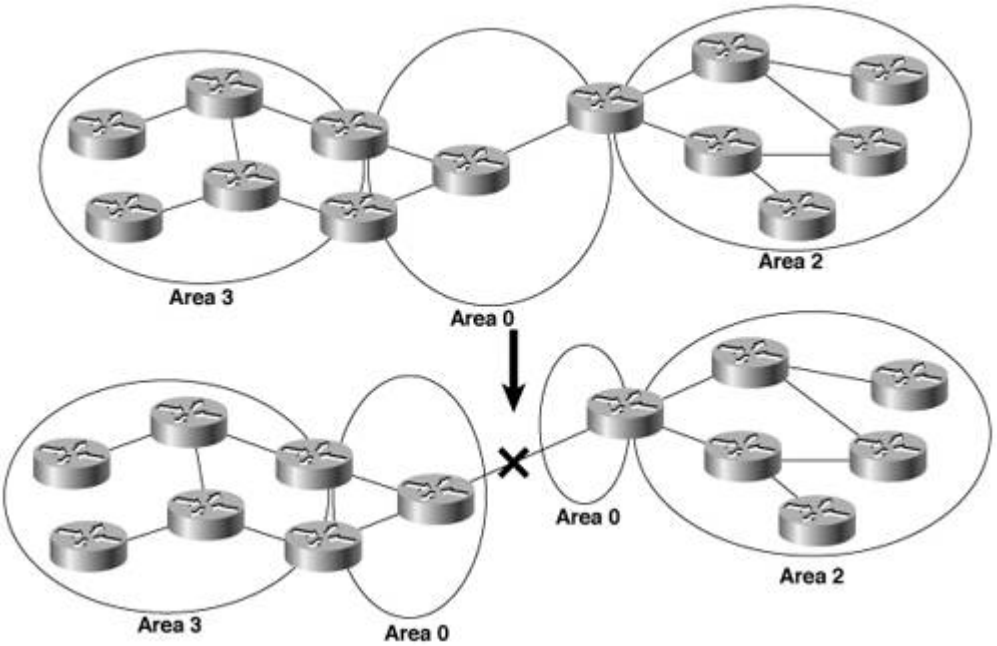
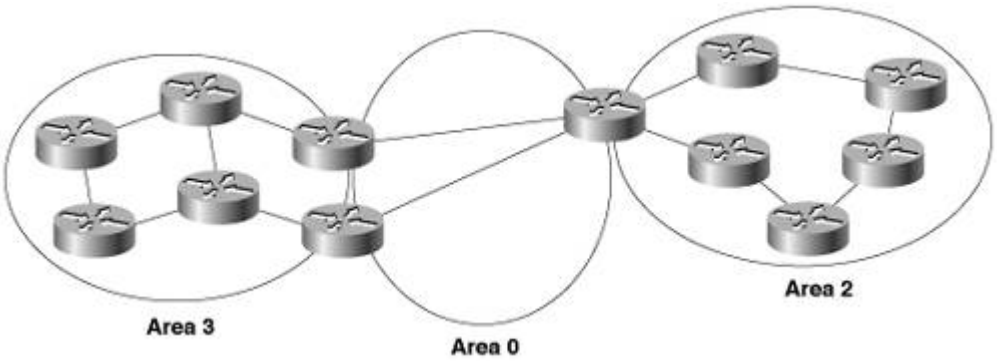



Figure 8-16 shows some better area designs. Both area 0 and area 2 are designed so that neither of them can be partitioned by a single link failure. The vulnerability of area 2, however, is that if the ABR fails, the area will be isolated. Area 3 uses two ABRs; here, neither a single link failure nor a single ABR failure can isolate any part of the area.

Figure 8-16. In areas 0 and 2, no single link failure can partition the area. In area 3, no single ABR or link failure can isolate the area.



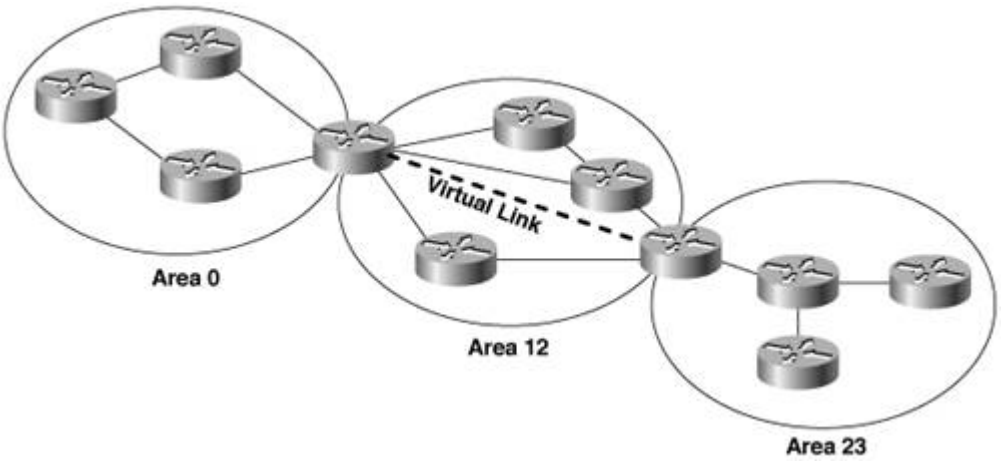

## Virtual Links

A virtual link is a link to the backbone through a non-backbone area. Virtual links are used for

the following purposes:

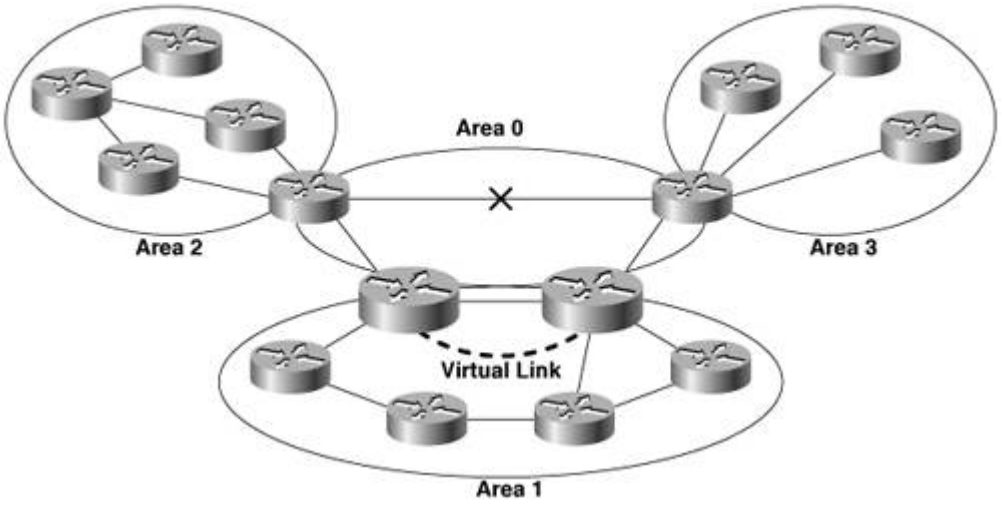
1. To link an area to the backbone through a non-backbone area (Figure 8-17)

Figure 8-17. A virtual link connects area 23 to the backbone through area 12.




2. To connect the two parts of a partitioned backbone through a nonbackbone area (Figure 8-18)

Figure 8-18. A virtual link reconnects a partitioned backbone through a nonbackbone area.




In both examples, the virtual link is not associated with a particular physical link. The virtual link is a tunnel through which packets may be routed on the optimal path from one endpoint to the other.

Several rules are associated with the configuration of virtual links:

- Virtual links must be configured between two ABRs.

- The area through which the virtual link is configured, known as the *transit area*, must have full routing information.
- The transit area cannot be a stub area.

As mentioned previously, OSPF classifies a virtual link as a network type. Specifically, the link is considered an unnumberedthat is, unaddressedlink, belonging to the backbone, between two ABRs. These ABRs are considered neighbors by virtue of the virtual link between them, although they are not linked physically. Within each ABR, the virtual link will transition to the fully functional point-to-point interface state when a route to the neighboring ABR is found in the route table. The cost of the link is the cost of the route to the neighbor. When the interface state becomes point-to-point, an adjacency is established across the virtual link.

Virtual links add a layer of complexity and troubleshooting difficulty to any network. It is best to avoid the need for them by ensuring that areas, particularly backbone areas, are designed with redundant links to prevent partitioning. When two or more networks are merged, sufficient planning should take place beforehand so that no area is left without a direct link to the backbone.

If a virtual link is configured, it should be used only as a temporary fix to an unavoidable topology problem. A virtual link is a flag marking a part of the network that needs to be reengineered. Permanent virtual links are virtually always a sign of a poorly designed network.

## Link-State Database

All valid LSAs received by a router are stored in its link-state database. The collected LSAs will describe a graph of the area topology. Because each router in an area calculates its shortest path tree from this database, it is imperative for accurate routing that all area databases are identical.

A list of the LSAs in a link-state database can be observed with the command `show ip ospf database`, as shown in Example 8-9. This list does not show all of the information stored for each LSA, but shows only the information in the LSA header. Note that this database contains LSAs from multiple areas, indicating that the router is an ABR.

### Example 8-9. The command *show ip ospf database* displays a list of all LSAs in the link-state database.

```
Homer#show ip ospf database

OSPF Router with ID (192.168.30.50) (Process ID 1)

                    Router Link States (Area 0)

Link ID             ADV Router      Age   Seq#        Checksum   Link count
192.168.30.10       192.168.30.10   1010  0x80001416  0xA818     3
192.168.30.20       192.168.30.20   677   0x800013C9  0xDE18     3
192.168.30.70       192.168.30.70   857   0x80001448  0xFD79     3
192.168.30.80       192.168.30.80   1010  0x800014D1  0xEB5C     5

                    Net Link States (Area 0)

Link ID             ADV Router      Age   Seq#        Checksum
192.168.17.18      192.168.30.20    677   0x800001AD  0x849A
192.168.17.34      192.168.30.60    695   0x800003E2  0x4619
192.168.17.58      192.168.30.40    579   0x8000113C  0xF0D
192.168.17.73      192.168.30.70    857   0x8000044F  0xB0E7
```

```
                    Summary Net Link States (Area 0)

Link ID             ADV Router      Age   Seq#        Checksum
172.16.121.0        192.168.30.60   421   0x8000009F  0xD52
172.16.121.0        192.168.30.70   656   0x8000037F  0x86A
10.63.65.0          192.168.30.10   983   0x80000004  0x1EAA
10.63.65.0          192.168.30.80   962   0x80000004  0x780A

                    Summary ASB Link States (Area 0)

Link ID             ADV Router      Age   Seq#        Checksum
192.168.30.12       192.168.30.20   584   0x80000005  0xFC4C
192.168.30.12       192.168.30.30   56    0x80000004  0x45BA
172.20.57.254       192.168.30.70   664   0x800000CE  0xF2CF
172.20.57.254       192.168.30.80   963   0x80000295  0x23CC

                    Router Link States (Area 4)

Link ID             ADV Router      Age   Seq#         Checksum   Link count
192.168.30.14      192.168.30.14   311   0x80000EA5   0x93A0     7
192.168.30.24      192.168.30.24   685   0x80001333   0x6F56     6
192.168.30.50      192.168.30.50   116   0x80001056   0x42BF     2
192.168.30.54      192.168.30.54   1213  0x80000D1F   0x3385     2

                    Summary Net Link States (Area 4)

Link ID             ADV Router      Age   Seq#        Checksum
172.16.121.0        192.168.30.40   1231  0x80000D88  0x73BF
172.16.121.0        192.168.30.50   34    0x800003F4  0xF90D
10.63.65.0          192.168.30.40   1240  0x80000003  0x5110
10.63.65.0          192.168.30.50   42    0x80000005  0x1144

                    Summary ASB Link States (Area 4)

Link ID             ADV Router      Age   Seq#        Checksum
192.168.30.12       192.168.30.40   1240  0x80000006  0x6980
192.168.30.12       192.168.30.50   42    0x80000008  0xC423
172.20.57.254       192.168.30.40   1241  0x8000029B  0xEED8
172.20.57.254       192.168.30.50   43    0x800002A8  0x9818

                    AS External Link States

Link ID             ADV Router      Age   Seq#        Checksum   Tag
10.83.10.0          192.168.30.60   459   0x80000D49  0x9C0B     0
10.1.27.0           192.168.30.62   785   0x800000EB  0xB5CE     0
10.22.85.0          192.168.30.70   902   0x8000037D  0x1EC0     65502
10.22.85.0          192.168.30.80   1056  0x800001F7  0x6B4B     65502
Homer#
```

Most of the entries in Example 8-9 have been deleted for brevity; the actual link-state database contains 1,445 entries and four areas, as shown in Example 8-10.

## Example 8-10. The command *show ip ospf database database-summary* displays the number of LSAs in a link-state database by area and by LSA type.

```
Homer#show ip ospf database database-summary
```

```
                OSPF Router with ID (192.168.30.50) (Process ID 1)

Area ID         Router  Network  Sum-Net  Sum-ASBR  Subtotal  Delete  Maxage
0               8       4        185      27        224       0       0
4               7       0        216      26        249       0       0
5               7       0        107      13        127       0       0
56              2       1        236      26        265       0       0
AS External                                        580       0       0
Total           24      5        744      92        1445
Homer#
```

As mentioned earlier in "Reliable Flooding: Sequencing, Checksums, and Aging," the LSAs are aged as they reside in the link-state database. If they reach MaxAge (1 hour), they are flushed from the OSPF domain. The implication here is that there must be a mechanism for preventing legitimate LSAs from reaching MaxAge and being flushed. This mechanism is the *link-state refresh*. Every 30 minutes, known as the LSRefreshTime, the router that originated the LSA floods a new copy of the LSA with an incremented sequence number and an age of zero. Upon receipt, the other OSPF routers replace the old copy of the LSA and begin aging the new copy.

So the link-state refresh process can be thought of as a keepalive for each LSA. An additional benefit is that any LSAs that might have become corrupted in a router's LS database are replaced with the refreshed copy of the legitimate LSA.

The idea behind associating an individual refresh timer with each LSA is that the LSRefreshTime of the LSAs do not expire all at once, reflooding all LSAs every 30 minutes. Instead, the reflooding is spread out in a semi-random pattern. The problem with this approach is that with each individual LSA being reflooded as its LSRefreshTime expires, bandwidth is used inefficiently. Update packets can be transmitted with only a few, or even a single, LSA.

Prior to IOS 11.3, Cisco chose to have a single LSRefreshTime associated with the entire LS database. Every 30 minutes, each router refreshes all of the LSAs it originated, regardless of their actual age. Although this strategy avoids the problem of inefficiency, it reintroduces the problem the individual refresh timers were meant to solve. If the LS database is large, each router can create spikes in the area traffic and CPU usage every half hour.

Therefore a mechanism known as *LSA group pacing* was introduced to reach a compromise between the problems of individual refresh timers and a single monolithic timer. Each LSA has its own refresh timer, but as the individual refresh timers expire, a delay is introduced before the LSAs are flooded. By delaying the refresh, more LSAs can be grouped together before being flooded, so that Update packets are carrying a larger number of LSAs. By default, the group-pacing interval is 240 seconds (4 minutes). Depending on the IOS version, the default can be changed with either `timers lsa-group-pacing` or `timers pacing lsa-group`.[13] If the database is very large (10,000 or more LSAs), decreasing the group pacing interval is beneficial; if the database is small, increasing the interval might be useful. The range of the group pacing timer is 10 to 1800 seconds.

[13] Group pacing was introduced in IOS 11.3AA with the command **timers lsa-group-pacing**; beginning with IOS 12.2(4)T, the command changed **to timers pacing lsa-group.**

## LSA Types

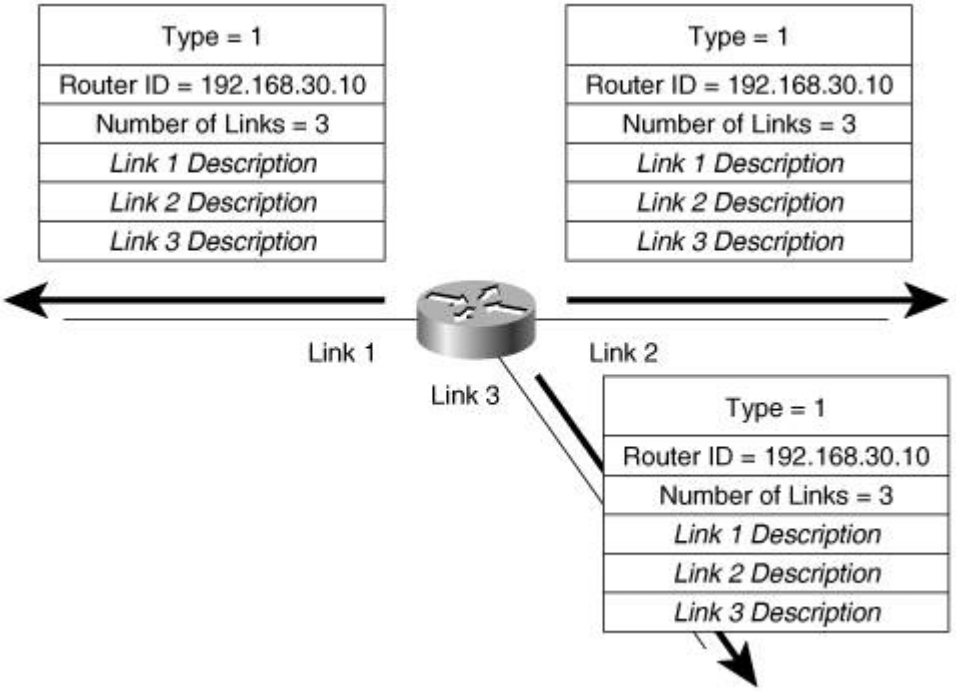
Because of the multiple router types defined by OSPF, multiple types of LSA are also necessary. For example, a DR must advertise the multi-access link and all the routers attached to the link. Other router types would not advertise this type of information. Both Example 8-9 and Example 8-10 show that there are multiple types of LSAs. Each type describes a different aspect of an OSPF network. Table 8-4 lists the LSA types and the type codes that identify them.

Table 8-4. LSA types.

| Type Code | Description |
|---|---|
| 1 | Router LSA |
| 2 | Network LSA |
| 3 | Network Summary LSA |
| 4 | ASBR Summary LSA |
| 5 | AS External LSA |
| 6 | Group Membership LSA |
| 7 | NSSA External LSA |
| 8 | External Attributes LSA |
| 9 | Opaque LSA (link-local scope) |
| 10 | Opaque LSA (area-local scope) |
| 11 | Opaque LSA (AS scope) |

*Router LSAs* are produced by every router (Figure 8-19). This most fundamental LSA lists all of a router's links, or interfaces, the state and outgoing cost of each link, and any known OSPF neighbors on the link. These LSAs are flooded only within the area in which they are originated. The command `show ip ospf database router` will list all of the Router LSAs in a database. Example 8-11 shows a variant of the command, in which a single router LSA is observed by specifying the router's ID. As this and the subsequent illustrations show, the complete LSA is recorded in the link-state database. For a description of all the LSA fields, see the "OSPF Packet Formats" section later in this chapter.

Figure 8-19. The Router LSA describes all of a router's interfaces.



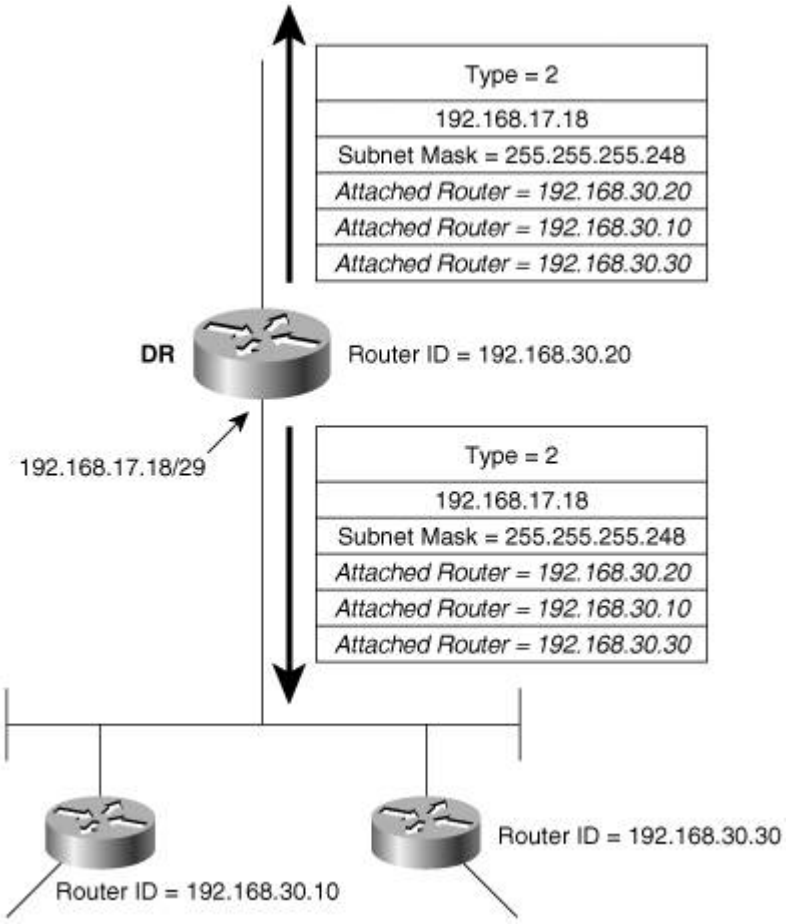
Example 8-11. The command *show ip ospf database router* displays Router LSAs from the link-state database.

```
Homer#show ip ospf database router 192.168.30.10

      OSPF Router with ID (192.168.30.50) (Process ID 1)

                    Router Link States (Area 0)

Routing Bit Set on this LSA
LS age: 680
Options: (No TOS-capability)
LS Type: Router Links
Link State ID: 192.168.30.10
Advertising Router: 192.168.30.10
LS Seq Number: 80001428
Checksum: 0x842A
Length: 60
Area Border Router
 Number of Links: 3

  Link connected to: another Router (point-to-point)
   (Link ID) Neighboring Router ID: 192.168.30.80
   (Link Data) Router Interface address: 192.168.17.9
    Number of TOS metrics: 0
     TOS 0 Metrics: 64

  Link connected to: a Stub Network
   (Link ID) Network/subnet number: 192.168.17.8
   (Link Data) Network Mask: 255.255.255.248
    Number of TOS metrics: 0
     TOS 0 Metrics: 64

  Link connected to: a Transit Network
   (Link ID) Designated Router address: 192.168.17.18
   (Link Data) Router Interface address: 192.168.17.17
    Number of TOS metrics: 0
     TOS 0 Metrics: 10

Homer#
```

One line you will notice in Example 8-11 and in several subsequent LSA displays is the statement "Routing Bit Set on this LSA." The routing bit is not a part of the LSA itself; it is an internal maintenance bit used by IOS indicating that the route to the destination advertised by this LSA is valid. So when you see "Routing Bit Set on this LSA," it means that the route to this destination is in the routing table.

*Network LSAs* are produced by the DR on every multi-access network (Figure 8-20). As discussed earlier, the DR represents the multi-access network and all attached routers as a pseudonode, or a single virtual router. In this sense, a Network LSA represents a pseudonode just as a Router LSA represents a single physical router. The Network LSA lists all attached routers, including the DR itself. Like Router LSAs, Network LSAs are flooded only within the originating area. In Example 8-12, the command show ip ospf database network is used to observe a Network LSA.

Figure 8-20. A DR originates a Network LSA to represent a multi-access network and all attached routers.



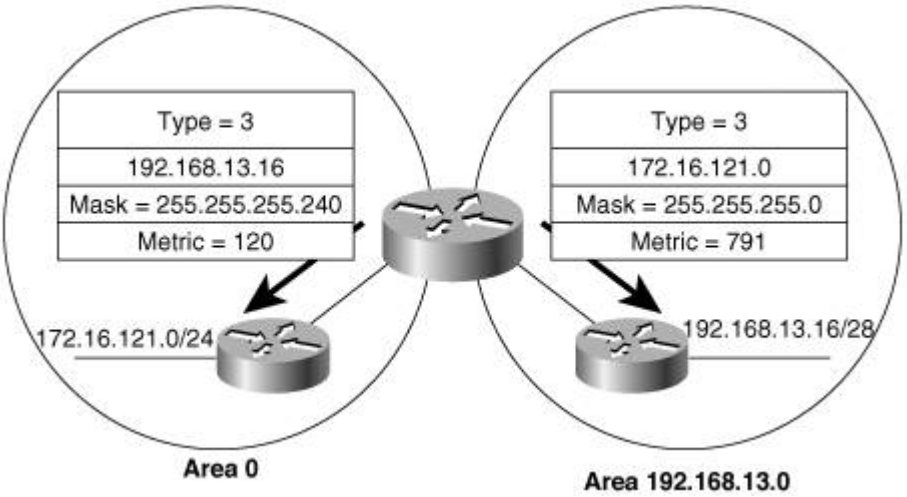

Example 8-12. Network LSAs can be observed with the command *show ip ospf database network*.

```
Homer#show ip ospf database network 192.168.17.18

       OSPF Router with ID (192.168.30.50) (Process ID 1)

                     Net Link States (Area 0)

Routing Bit Set on this LSA
LS age: 244
Options: (No TOS-capability)
LS Type: Network Links
Link State ID: 192.168.17.18 (address of Designated Router)
Advertising Router: 192.168.30.20
LS Seq Number: 800001BF
Checksum: 0x60AC
Length: 32
Network Mask: /29
      Attached Router: 192.168.30.20
      Attached Router: 192.168.30.10
      Attached Router: 192.168.30.30

Homer#
```

Notice that unlike the Router LSA, there is no metric field in the Network LSA. This is because, as explained earlier in this chapter, the cost from the pseudonode represented by the LSA to any attached router is always 0.

*Network Summary LSAs* are originated by ABRs. They are sent into a single area to advertise destinations outside that area (Figure 8-21). In effect, these LSAs are the means by which an ABR tells the internal routers of an attached area what destinations the ABR can reach. An ABR also advertises the destinations within its attached areas into the backbone with Network Summary LSAs. Default routes external to the area, but internal to the OSPF autonomous system, are also advertised by this LSA type. The command show ip ospf database summary is used to display the network summary LSAs in the database, as shown in Example 8-13.

Figure 8-21. An ABR will originate a Network Summary LSA to describe inter-area destinations.




Example 8-13. Network Summary LSAs can be observed with the command *show ip ospf database summary*.

```
Homer#show ip ospf database summary 172.16.121.0

       OSPF Router with ID (192.168.30.50) (Process ID 1)

                     Summary Net Link States (Area 0)

Routing Bit Set on this LSA
LS age: 214
Options: (No TOS-capability)
LS Type: Summary Links(Network)
Link State ID: 172.16.121.0 (summary Network Number)
Advertising Router: 192.168.30.60
LS Seq Number: 800000B1
Checksum: 0xE864
Length: 28
Network Mask: /24
   TOS: 0 Metric: 791
```

When an ABR originates a Network Summary LSA, it includes the cost from itself to the

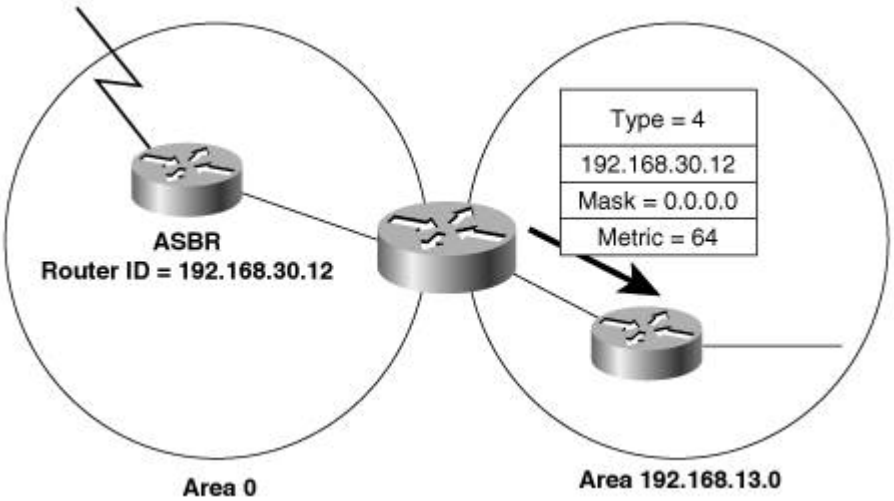
destination the LSA is advertising. The ABR will originate only a single Network Summary LSA for each destination even if it knows of multiple routes to the destination. Therefore, if an ABR knows of multiple routes to a destination within its own attached area, it originates a single Network Summary LSA into the backbone with the lowest cost of the multiple routes. Likewise, if an ABR receives multiple Network Summary LSAs from other ABRs across the backbone, the original ABR will choose the lowest cost advertised in the LSAs and advertise that one cost into its attached non-backbone areas.

When another router receives a Network Summary LSA from an ABR, it does not run the SPF algorithm. Rather, it simply adds the cost of the route to the ABR and the cost included in the LSA. A route to the advertised destination, via the ABR, is entered into the route table along with the calculated cost. This behaviordepending on an intermediate router instead of determining the full route to the destinationis distance vector behavior. So, while OSPF is a link-state protocol within an area, it uses a distance vector algorithm to find inter-area routes.[14]

[14] This distance vector behavior is the reason for requiring a backbone area and requiring that all inter-area traffic pass through the backbone. By forming the areas into what is essentially a hub-and-spoke topology, the route loops to which distance vector protocols are prone are avoided.

*ASBR Summary LSAs* are also originated by ABRs. ASBR Summary LSAs are identical to Network Summary LSAs except that the destination they advertise is an ASBR (Figure 8-22), not a network. The command show ip ospf database asbr-summary is used to display ASBR Summary LSAs (Example 8-14). Note in the illustration that the destination is a host address, and the mask is zero; the destination advertised by an ASBR Summary LSA will always be a host address because it is a route to a router.

Figure 8-22. ASBR Summary LSAs advertise routes to ASBRs.




## Example 8-14. ASBR Summary LSAs can be observed with the command *show ip ospf database asbr-summary*.

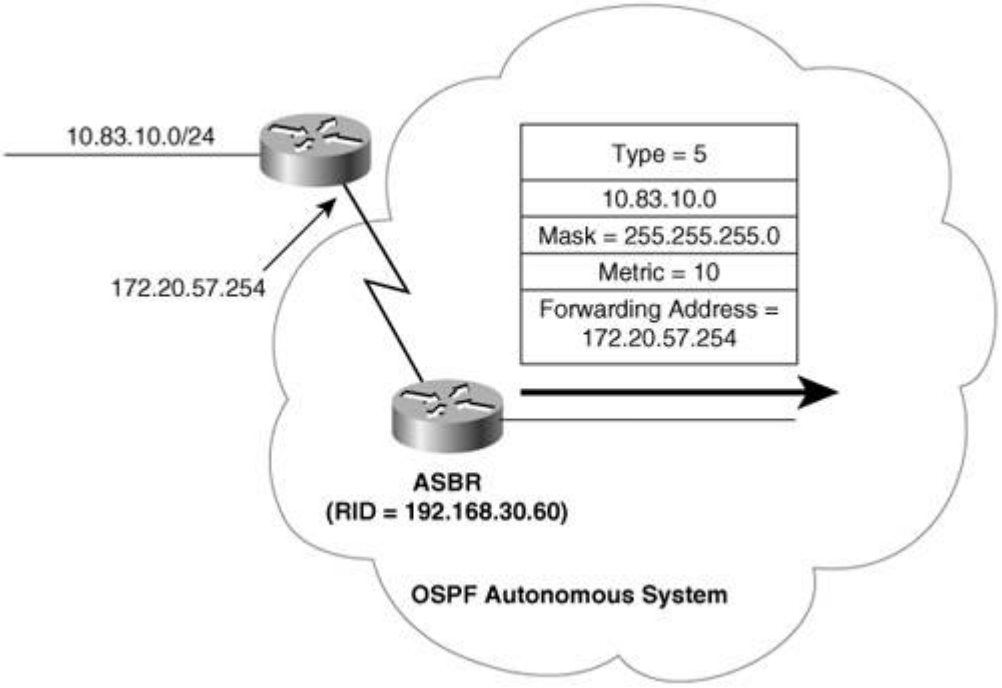
```
Homer#show ip ospf database asbr-summary

       OSPF Router with ID (192.168.30.50) (Process ID 1)

                    Summary ASB Link States (Area 0)

Routing Bit Set on this LSA
LS age: 1640
Options: (No TOS-capability)
```

```
LS Type: Summary Links (AS Boundary Router)
Link State ID: 192.168.30.12 (AS Boundary Router address)
Advertising Router: 192.168.30.20
LS Seq Number: 80000009
Checksum: 0xF450
Length: 28
Network Mask: /0
   TOS: 0 Metric: 64
--More-
```

*Autonomous System External LSAs*, or *External LSAs*, are originated by ASBRs. They advertise either a destination external to the OSPF autonomous system, or a default route[15] external to the OSPF autonomous system (Figure 8-23). Referring back to Example 8-9, you can see that the AS External LSAs are the only LSA types in the database that are not associated with a particular area; external LSAs are flooded throughout the autonomous system. The command show ip ospf database external displays AS External LSAs (Example 8-15).

[15] Default routes are routes that are chosen if no more specific route exists in the route table. OSPF and RIP use an IP address of 0.0.0.0 to identify a default route. See Chapter 12, "Default Routes and On-Demand Routing" for more information.

Figure 8-23. AS External LSAs advertise destinations external to the OSPF autonomous system.




Example 8-15. AS External LSAs can be observed with the command *show ip ospf database external*.

```
Homer#show ip ospf database external 10.83.10.0

       OSPF Router with ID (192.168.30.50) (Process ID 1)

                       AS External Link States
```

```
Routing Bit Set on this LSA
LS age: 1680
Options: (No TOS-capability)
LS Type: AS External Link
Link State ID: 10.83.10.0 (External Network Number)
Advertising Router: 192.168.30.60
LS Seq Number: 80000D5A
Checksum: 0x7A1C
Length: 36
Network Mask: /24
      Metric Type: 1 (Comparable directly to link state metric)
      TOS: 0
      Metric: 10
      Forward Address: 172.20.57.254
      External Route Tag: 0
Homer#
```

*Group Membership LSAs* are used in an enhancement of OSPF known as *Multicast OSPF* (MOSPF).[16] MOSPF routes packets from a single source to multiple destinations, or group members, which share a class D multicast address. Although Cisco supports other multicast routing protocols, MOSPF is not supported as of this writing. For this reason, neither MOSPF nor the Group Membership LSA is covered in this book.

[16] John Moy, "Multicast Extensions to OSPF," RFC 1584, March 1994.

*NSSA External LSAs* are originated by ASBRs within not-so-stubby areas (NSSAs). NSSAs are described in the following section. An NSSA External LSA is almost identical to an AS External LSA, as the section on OSPF packet formats shows. Unlike AS External LSAs, which are flooded throughout an OSPF autonomous system, NSSA External LSAs are flooded only within the not-so-stubby area in which it was originated. The command `show ip ospf database nssa-external` displays NSSA External LSAs (Example 8-16).

## Example 8-16. NSSA External LSAs can be observed with the command *show ip ospf database nssa-external*.

```
Morisot#show ip ospf database nssa-external

         OSPF Router with ID (10.3.0.1) (Process ID 1)

                     Type-7 AS External Link States (Area 15)

LS age: 532
Options: (No TOS-capability, No Type 7/5 translation, DC)
LS Type: AS External Link
Link State ID: 10.0.0.0 (External Network Number)
Advertising Router: 10.3.0.1
LS Seq Number: 80000001
Checksum: 0x9493
Length: 36
Network Mask: /16
      Metric Type: 2 (Larger than any link state path)
      TOS: 0
      Metric: 100
      Forward Address: 10.3.0.1
      External Route Tag: 0

--More-
```

*External Attributes LSAs* were proposed as an alternative to running Internal BGP (iBGP), to transport BGP information across an OSPF domain. This LSA has never been deployed on a wide scale, and is not supported in IOS.

*Opaque LSAs* are a class of LSAs that consist of a standard LSA header followed by application-specific information.[17] The Information field can be used directly by OSPF or indirectly by other applications to distribute information throughout the OSPF domain. Opaque LSAs have been used to add various extensions to OSPF, such as traffic engineering parameters for Multiprotocol Label Switching (MPLS) networks.

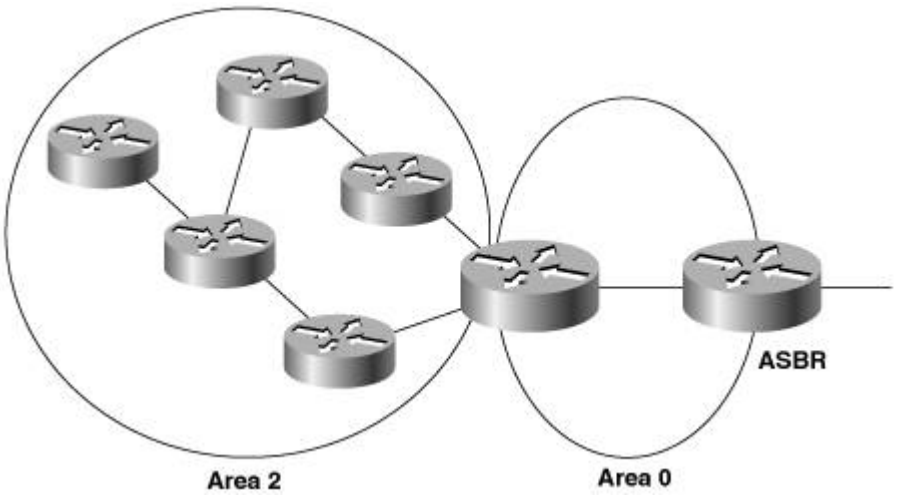
[17] Rob Coltun, "The OSPF Opaque LSA Option," RFC 2370, July 1998.

## Stub Areas

An ASBR learning external destinations will advertise those destinations by flooding AS External LSAs throughout the OSPF autonomous system. In many cases, these External LSAs may make up a large percentage of the LSAs in the databases of every router. For example, Example 8-10 shows that 580 of the LSAs in that database40 percentare external LSAs.

In Figure 8-24, not every router needs to know about all the external destinations. The routers in area 2 must send a packet to an ABR to reach the ASBR, no matter what the external destination might be. For this reason, area 2 can be configured as a *stub area*.

Figure 8-24. Memory can be conserved and performance improved by making area 2 a stub area.




A *stub area* is an area into which AS External LSAs are not flooded. And if type 5 LSAs are not known inside an area, type 4 LSAs are unnecessary; these LSAs are also blocked. ABRs at the edge of a stub area use Network Summary (type 3) LSAs to advertise a single default route (destination 0.0.0.0) into the area. Any destination that the internal routers cannot match to an intra- or inter-area route will match the default route. Because the default route is carried in type 3 LSAs, it will not be advertised outside of the area.

The performance of routers within a stub area can be improved, and memory conserved, by the reduced size of their databases. Of course, the improvement will be more marked in OSPF domains with a large number of type 5 LSAs. There are, however, four restrictions on stub areas:

1. As in any area, all routers in a stub area must have identical link-state databases. To ensure this condition, all stub routers will set a flag (the E-bit) in their Hello packets to zero; they will not accept any Hello from a router in which the E-bit is set to one. As a result, adjacencies will not be established with any router that is not configured as a stub router.

2. Virtual links cannot be configured within, nor transit, a stub area.

3. No router within a stub area can be an ASBR. This restriction is intuitively understandable because ASBRs produce type 5 LSAs, and type 5 LSAs cannot exist within a stub area.

4. A stub area might have more than one ABR, but because of the default route, the internal routers cannot determine which router is the optimal gateway to the ASBR.

## Totally Stubby Areas

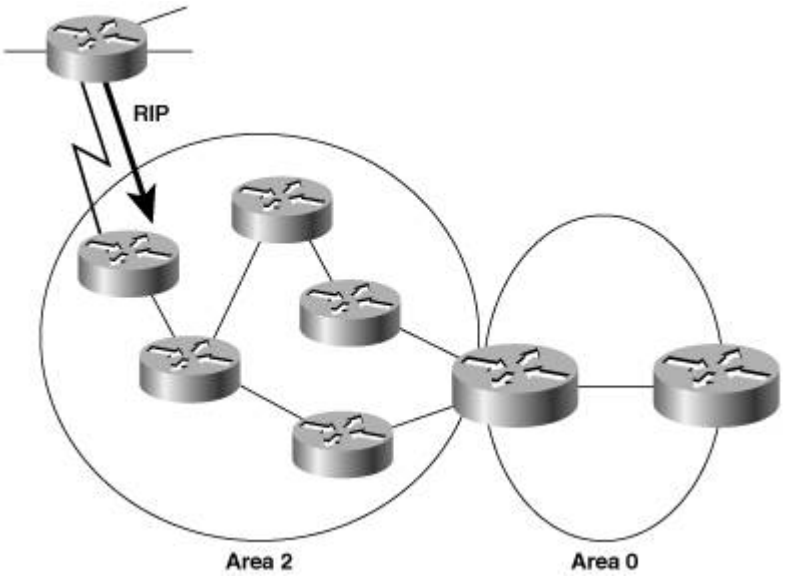
If memory is saved by blocking the propagation of type 5 and type 4 LSAs into an area, wouldn't more memory be saved by blocking type 3 LSAs? In addressing this question, Cisco carries the concept of stub areas to its logical conclusion with a scheme known as *totally stubby areas*.

Totally stubby areas use a default route to reach not only destinations external to the autonomous system but also all destinations external to the area. The ABR of a totally stubby area will block not only AS External LSAs but also all Summary LSAswith the exception of a single type 3 LSA to advertise the default route.

## Not-So-Stubby Areas

In Figure 8-25, a router with a few stub networks must be attached to the OSPF domain via one of the area 2 routers. The router supports only RIP, so the area 2 router will run RIP and redistribute the networks into OSPF. Unfortunately, this configuration makes the area 2 router an ASBR, and therefore area 2 can no longer be a stub area.

Figure 8-25. Because a few external destinations must be redistributed into OSPF at one of the area 2 routers, all of area 2 is ineligible to be a stub area.



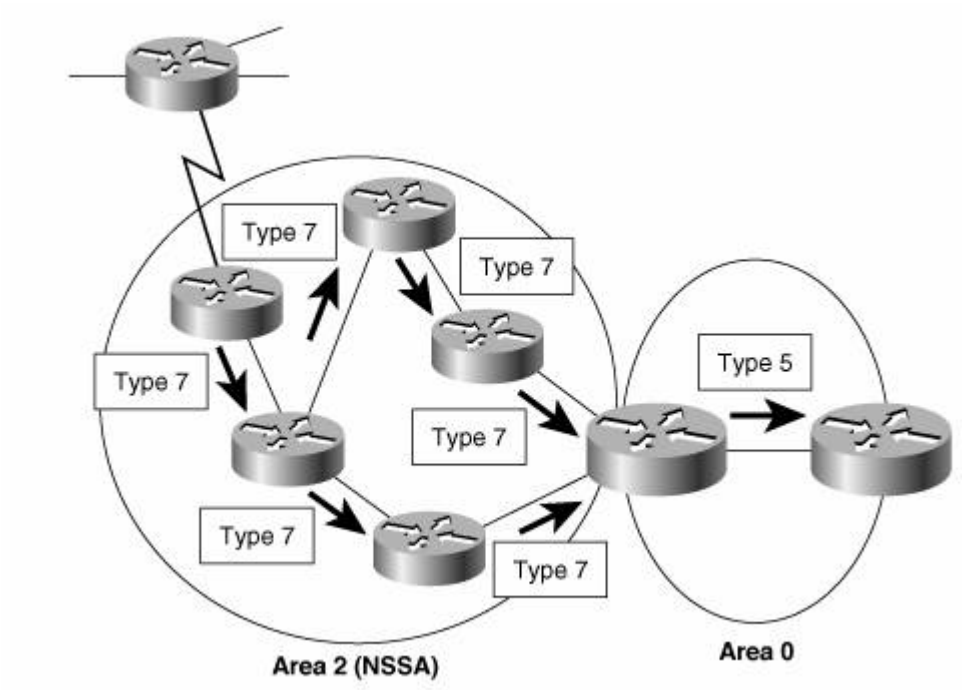
The RIP speaker does not need to learn routes from OSPFa default route pointing to the area 2 router is all it needs. But all OSPF routers must know about the networks attached to the RIP router to route packets to them.

*Not-so-stubby areas* (NSSAs)[18] allow external routes to be advertised into the OSPF autonomous system while retaining the characteristics of a stub area to the rest of the autonomous system. To do this, the ASBR in an NSSA will originate type 7 LSAs to advertise the external destinations. These NSSA External LSAs are flooded throughout the NSSA but are blocked at the ABR.

[18] Rob Coltun and Vince Fuller, "The OSPF NSSA Option," RFC 1587, March 1994.

The NSSA External LSA has a flag in its header known as the P-bit. The NSSA ASBR has the option of setting or clearing the P-bit. If the NSSA's ABR receives a type 7 LSA with the P-bit set to one, it will translate the type 7 LSA into a type 5 LSA and flood it throughout the other areas (see Figure 8-26). If the P-bit is set to zero, no translation will take place and the destination in the type 7 LSA will not be advertised outside of the NSSA. This option allows you to design an NSSA in which the external destinations learned in that area are known only in that area.

Figure 8-26. An ASBR within an NSSA will originate NSSA External LSAs. If the P-bit of an NSSA External LSA is set, the ABR will translate the LSA into an AS External LSA.




NSSAs are supported in IOS 11.2 and later.

Table 8-5 summarizes which LSAs are allowed in which areas.

Table 8-5. LSA types allowed per area type.

| Area Type | 1&2 | 3 | 4 | 5 | 7 |
|---|---|---|---|---|---|
| Backbone (area 0) | Yes | Yes | Yes | Yes | No |
| Non-backbone, non-stub | Yes | Yes | Yes | Yes | No |
| Stub | Yes | Yes | No | No | No |
| Totally stubby | Yes | No[*] | No | No | No |
| Not-so-stubby | Yes | Yes | Yes | No | Yes |

[*] Except for a single type 3 LSA per ABR, advertising the default route.

## Route Table

The Dijkstra algorithm is used to calculate the Shortest Path Tree from the LSAs in the link state database. Chapter 4 has a somewhat detailed discussion of the Dijkstra algorithm; for a full description of the OSPF calculation of the SPF tree, see section 16.1 of RFC 2328.

OSPF determines the shortest path based on an arbitrary metric called cost, which is assigned to each interface. The cost of a route is the sum of the costs of all the outgoing interfaces to a destination. RFC 2328 does not specify any values for cost. Cisco routers calculate a default OSPF cost as $10^8$/BW, where BW is the configured bandwidth of the interface and $10^8$ is the reference bandwidth. As discussed previously, the default reference bandwidth can be changed with the command `auto-cost reference-bandwidth`. Fractional costs are rounded down to the nearest whole number. Table 8-6 shows the default costs calculated by this formula for some typical interfaces.

Table 8-6. Cisco default interface costs.

| Interface Type | Cost ($10^8$/BW) |
|---|---|
| FDDI, Fast Ethernet, any interface > 100M | 1 |
| HSSI (45M) | 2 |
| 16M Token Ring | 6 |
| Ethernet | 10 |
| 4M Token Ring | 25 |
| T1 (1.544M) | 64 |
| DS0 (64K)[*] | 1562 |
| 56K[*] | 1785 |
| Tunnel (9K) | 11111 |

[*] Assumes the default bandwidth of the serial interface has been changed.

The command `ip ospf cost` can be used to override the default automatic cost calculations and assign a fixed cost to an interface. For example, a large network with homogeneous backbone link speeds might assign link costs based on line of sight or wire/fiber distance. LSAs record cost

in a 16-bit field, so the total cost of an interface can range from 1 to 65535.

## Destination Types

Each route entry is classified according to a *destination type*. The destination type will be either *network* or *router*.

*Network entries* are the addresses of networks to which packets can be routed. These are the destinations that are entered into the route table (Example 8-17).

### Example 8-17. The OSPF entries in the route table are network destination types.

```
Homer#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route

Gateway of last resort is 192.168.32.2 to network 0.0.0.0

O E1 192.168.118.0/24 [110/94] via 192.168.17.74, 02:15:01, Ethernet0
O E1 10.0.0.0/8 [110/84] via 192.168.17.41, 02:15:01, Serial0.19
O E1 192.168.119.0/24 [110/94] via 192.168.17.74, 02:15:01, Ethernet0
O E2 172.19.0.0/16 [110/21] via 192.168.32.2, 02:15:01, Ethernet1
     172.21.0.0/16 is variably subnetted, 2 subnets, 2 masks
O E2   172.21.0.0/16 [110/801] via 192.168.21.6, 02:15:01, Serial1.724
O      172.21.121.0/24 [110/791] via 192.168.21.6, 04:18:30, Serial1.724
     172.16.0.0/16 is variably subnetted, 104 subnets, 7 masks
O      172.16.21.48/30 [110/844] via 192.168.21.10, 04:18:48, Serial1.725
O IA   172.16.30.61/32 [110/856] via 192.168.17.74, 02:15:19, Ethernet0
O IA   172.16.35.0/24 [110/865] via 192.168.17.74, 02:15:19, Ethernet0
C      172.16.32.0/24 is directly connected, Ethernet1
O      172.16.17.48/29 [110/74] via 192.168.17.74, 06:19:46, Ethernet0
O E1   172.16.46.0/24 [110/30] via 192.168.32.2, 02:15:19, Ethernet1
O      172.16.45.0/24 [110/20] via 192.168.32.2, 3d10h, Ethernet1
O IA   172.16.30.54/32 [110/1061] via 192.168.17.74, 02:15:21, Ethernet0
O      172.16.17.56/29 [110/84] via 192.168.17.74, 06:19:48, Ethernet0
O      172.16.54.0/24 [110/11] via 192.168.32.2, 3d10h, Ethernet1
O      172.16.55.0/24 [110/11] via 192.168.32.2, 3d10h, Ethernet1
O      172.16.52.0/24 [110/11] via 192.168.32.2, 3d10h, Ethernet1
O      172.16.53.0/24 [110/11] via 192.168.32.2, 3d10h, Ethernet1
C      172.16.25.28/30 is directly connected, Tunnel29
--More-
```

*Router entries* are routes to ABRs and ASBRs. If a router needs to send a packet to an inter-area destination, it must know how to find an ABR; if a packet must go to an external destination, the router must know how to find an ASBR. Router entries contain this information, and are kept in a separate, internal route table. This table can be observed with the command `show ip ospf border-routers` (Example 8-18).

### Example 8-18. Router entries, kept in a separate table from network entries, are routes to ABRs and ASBRs.

```
Homer#show ip ospf border-routers

OSPF Process 1 internal Routing Table

Codes: i - Intra-area route, I - Inter-area route
i 192.168.30.10 [74] via 192.168.17.74, Ethernet0, ABR, Area 0, SPF 391
I 192.168.30.12 [148] via 192.168.17.74, Ethernet0, ASBR, Area 0, SPF 391
I 192.168.30.18 [205] via 192.168.17.74, Ethernet0, ASBR, Area 0, SPF 391
i 192.168.30.20 [84] via 192.168.17.74, Ethernet0, ABR, Area 0, SPF 391
i 192.168.30.27 [781] via 192.168.21.6, Serial1.724, ASBR, Area 7, SPF 631
i 192.168.30.30 [74] via 192.168.17.74, Ethernet0, ABR/ASBR, Area 0, SPF 391
I 192.168.30.38 [269] via 192.168.17.74, Ethernet0, ASBR, Area 0, SPF 391
i 192.168.30.37 [390] via 192.168.21.10, Serial1.725, ASBR, Area 7, SPF 631
i 192.168.30.40 [84] via 192.168.17.74, Ethernet0, ABR/ASBR, Area 0, SPF 391
i 192.168.30.47 [400] via 192.168.21.10, Serial1.725, ASBR, Area 7, SPF 631
i 192.168.30.50 [74] via 192.168.17.41, Serial0.19, ABR/ASBR, Area 0, SPF 391
I 192.168.30.62 [94] via 192.168.17.74, Ethernet0, ASBR, Area 0, SPF 391
i 192.168.30.60 [64] via 192.168.17.41, Serial0.19, ABR/ASBR, Area 0, SPF 391
i 192.168.30.60 [790] via 192.168.21.10, Serial1.725, ABR/ASBR, Area 7, SPF 631
i 192.168.30.80 [10] via 192.168.32.5, Ethernet1, ABR/ASBR, Area 78, SPF 158
i 192.168.30.80 [10] via 192.168.17.74, Ethernet0, ABR/ASBR, Area 0, SPF 391
i 172.20.57.254 [10] via 192.168.32.2, Ethernet1, ASBR, Area 78, SPF 158
Homer#
```

As Example 8-18 shows, the internal route table looks very similar to any other route tablethere are destinations, metrics, next-hop addresses, and exit interfaces. The difference is that all destinations are the Router IDs of ABRs and ASBRs. Each entry is tagged as intra-area (i) or inter-area (I), and the entry indicates whether the destination is an ABR, an ASBR, or both. The area is recorded, as is the iteration of the SPF algorithm that installed the entry.

## Path Types

Each route to a network destination is also classified as one of four *path types*. These path types are intra-area, inter-area, type 1 external, and type 2 external:

- Intra-area paths are to destinations within one of the router's attached areas.
- Inter-area paths are to destinations in another area but within the OSPF autonomous system. An inter-area path, tagged with an IA in Example 8-17, always passes through at least one ABR.
- Type 1 external paths (E1 in Example 8-17) are to destinations outside the OSPF autonomous system.

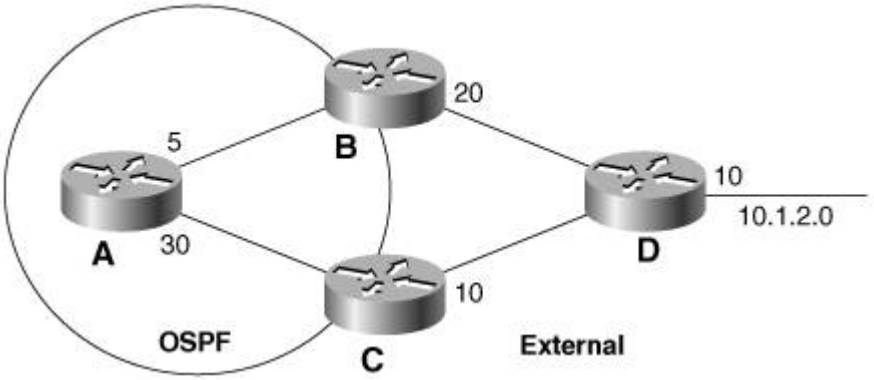
  When an external route is redistributed into any autonomous system, it must be assigned a metric that is meaningful to the routing protocol of the autonomous system. Within OSPF, the ASBR is responsible for assigning a cost to the external routes they advertise. Type 1 external paths have a cost that is the sum of this external cost plus the cost of the path to the ASBR. Configuring an ASBR to advertise an external (redistributed) route with an E1 metric is covered in Chapter 11, "Route Redistribution."
- Type 2 external paths (E2) are also to destinations outside the OSPF autonomous system, but do not take into account the cost of the path to the ASBR.

E1 and E2 routes provide the network administrator with the option of choosing whether the internal cost to the ASBR is important or whether only the external cost of an external route, disregarding the internal cost of reaching the ASBR, is more important. For example, "hot potato" routinggetting packets to external destinations out of the network at the closest exit

pointusually requires E1 metrics, whereas if you want packets to exit your network at the closest point to their external destination, use E2 metrics. OSPF external routes are, by default, E2 paths.

In Figure 8-27, router A has two paths to external destination 10.1.2.0. If the destination is advertised as E1, the A-B-D path will have a cost of 35 (5 + 20 + 10) and will be preferred over the A-C-D path whose cost is 50 (30 + 10 + 10). If the destination is advertised as E2, the cost of the two internal links to the ASBRs will be disregarded. In this case, the A-B-D path has a cost of 30 (20 + 10) and the A-C-D path has a cost of 20 (10 + 10). The latter will be the preferred path.

Figure 8-27. If the route to external network 10.1.2.0 is advertised with an E1 metric, router A will choose B as the "closest" ASBR. If the destination is advertised with an E2 metric, C will be chosen as the ASBR.




## Route Table Lookups

When an OSPF router examines the destination address of a packet, it takes the following steps to select the best route:[19]

[19] The lookup procedure described here adheres to RFC 2328. The earlier OSPF RFCs specify creating a set of matching routes first, then choosing the preferred path type, and choosing the longest match last.

1. Select the route or routes with the most specific match to the destination address. For example, if there are route entries for 172.16.64.0/18, 172.16.64.0/24, and 172.16.64.192/27, and the destination address is 172.16.64.205, the last entry will be chosen. The most specific match should always be the longest matchthe route with the longest address mask. The entries may be host, subnet, network, supernet, or default addresses. If no match can be found, an ICMP Destination Unreachable message will be sent to the source address and the packet will be dropped.

2. Prune the set of selected entries by eliminating less-preferred path types. Path types are prioritized in the following order, with 1 being the most preferred and 4 being the least preferred:

   1. Intra-area paths

   2. Inter-area paths

   3. E1 external paths

4. E2 external paths

If multiple equal-cost, equal-path-type routes exist in the final set, OSPF utilizes them. By default, the Cisco OSPF implementation load balances over a maximum of 16 equal-cost paths (four in older versions of IOS); this number can be changed within the range of one to six with the command maximum-paths.[20]

[20] As this edition is being produced, Cisco is increasing the maximum supported by the **maximum-paths** command from 6 to 16.

## Authentication

OSPF has the capability of authenticating all packets exchanged between neighbors. Authentication may be by simple passwords or by MD5 cryptographic checksums. These authentication methods are discussed in Chapter 6, "RIPv2, RIPng, and Classless Routing," and examples of configuring OSPF authentication are given in the configuration section.

## OSPF over Demand Circuits

OSPF sends Hellos every 10 seconds and refreshes its LSAs every 30 minutes. These functions maintain the neighbor relationships, ensure that the link-state databases are accurate, and use less bandwidth than traditional distance vector protocols such as RIP. However, even this minimal traffic is undesirable on *demand circuits*usage-sensitive connections such as X.25 SVCs, ISDN, and dialup lines. The recurring charges for such links may be determined by connect time or traffic volume or both, thus motivating the network manager to minimize their uptime.

An enhancement that makes OSPF practical over demand circuits is the capability of suppressing the Hello and LSA refresh functions so that a link does not have to be constantly up.[21] Although this enhancement is designed specifically for usage-sensitive circuits, it might be useful on any bandwidth-limited link.[22]

[21] John Moy, "Extending OSPF to Support Demand Circuits," RFC 1793, April 1995.

[22] Although OSPF over demand circuits may be configured on any interface, Hellos are not suppressed on multi-access network types; doing so would prevent the DR processes from functioning properly. As a result, the enhancement is really useful only on point-to-point and point-to-multipoint network types.

OSPF over demand circuits brings up a demand link to perform the initial database synchronization and subsequently brings up the link to flood only LSAs in which certain changes have occurred. These LSA changes are

- A change in the LSA Options field.
- A new instance of an existing LSA is received in which the age is MaxAge.
- A change in the Length field of the LSA header.
- A change in the contents of the LSA, excluding the 20-octet header, the checksum, or the sequence number.

Because no periodic Hellos are exchanged (Hellos are used only to bring up the link), OSPF must make a *presumption of reachability*. That is, it must presume that the demand circuit will be available when needed. In some instances, however, the link might not be immediately accessible. For example, a dialup link might be in use, both B channels of a BRI link might be in use, or the maximum number of allowed X.25 SVCs may already be up. In these situations, where the link is unavailable not because it is down but because of normal operational characteristics, the link is *oversubscribed*.

OSPF will not report an oversubscribed demand link as down, and packets routed to an

oversubscribed link will be dropped rather than being queued. This behavior makes sense because there is no way to predict when the link will again become available; a stream of packets to an unavailable interface could overflow the buffers.

Several changes to the interface and neighbor state machines and to the flooding procedure must be made to support OSPF over demand circuits (see RFC 1793 for more details). Within the LSA format, two changes are made.

First, if LSAs are not periodically refreshed across a demand circuit, no routers on the other side of the link should declare the LSA invalid after MaxAge. The semantics of the LSA's Age field are changed to accomplish this by designating the high bit as the *DoNotAge* bit. When an LSA is flooded over a demand circuit, the transmitting router will set DoNotAge = 1. As the LSA is flooded to all routers on the other side of the link, the Age field will be incremented normally by InfTransDelay seconds.[23] However, after being installed in a database, an LSA will not be aged like the other LSAs.

[23] Note that this means MaxAge will actually be MaxAge + DoNotAge.

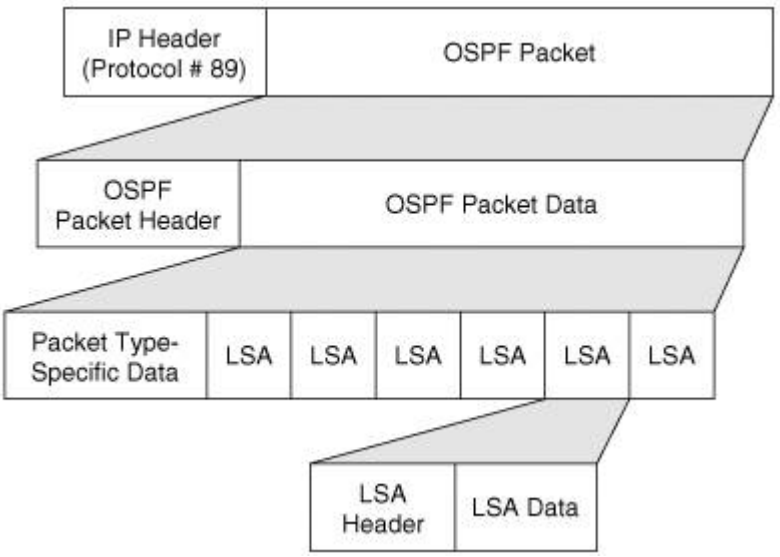
The second change derives from the first change. Because all routers must be capable of correctly interpreting the DoNotAge bit, a new flag known as the *Demand Circuit bit* (DC-bit) is added to all LSAs. By setting this flag in all LSAs it originates, a router signals to the other routers that it is capable of supporting OSPF over demand circuits.

A peripheral benefit of the enhancements made by OSPF over Demand Circuitsnamely, designating the high bit of the Age field as the DoNotAge bitis that you can now reduce flooding in stable topologies. With the command `ip ospf flood-reduction` entered for an interface, the DoNotAge bit is set on LSAs advertised out the interface and the LSAs are not refreshed unless they change.

## OSPF Packet Formats

The OSPF packet consists of multiple encapsulations, and deconstructing one is like peeling an onion. As shown in Figure 8-28, the outside of the onion is the IP header. Encapsulated within the IP header is one of five OSPF packet types. Each packet type begins with an OSPF packet header, whose format is the same for all packet types. The OSPF packet data following the header varies according to the packet type. Each packet type has a number of type-specific fields, followed by more data. The data contained in a Hello packet is a list of neighbors. LS Request packets contain a series of fields describing the requested LSAs. LS Update packets contain a list of LSAs, as shown in Figure 8-28. These LSAs in turn have their own headers and type-specific data fields. Database Description and LS Acknowledgment packets contain a list of LSA headers.

Figure 8-28. An OSPF packet is composed of a series of encapsulations.




Note that OSPF packets are exchanged only between neighbors on a network. They are never routed beyond the network on which they originate.

Figure 8-29 shows an analyzer capture of an IP header for a packet carrying OSPF data, indicated by the protocol number of 89. When OSPF packets are multicast, their TTL is set to 1, as can be seen here. Because an OSPF packet should never be routed past an immediate neighbor, setting the TTL to 1 helps to ensure that the packet never travels more than a single hop. Some routers run processes that prioritize packets according to the Precedence bits (Weighted Fair Queuing and Weighted Random Early Detection, for example). OSPF sets the Precedence bits to Internetwork Control (110b), as shown in Figure 8-29, so that these processes will give a high priority to OSPF packets.

Figure 8-29. OSPF uses a protocol number of 89. It also sets the TTL value in the IP header to 1 and the Precedence bits to Internetwork Control.

[View full size image]

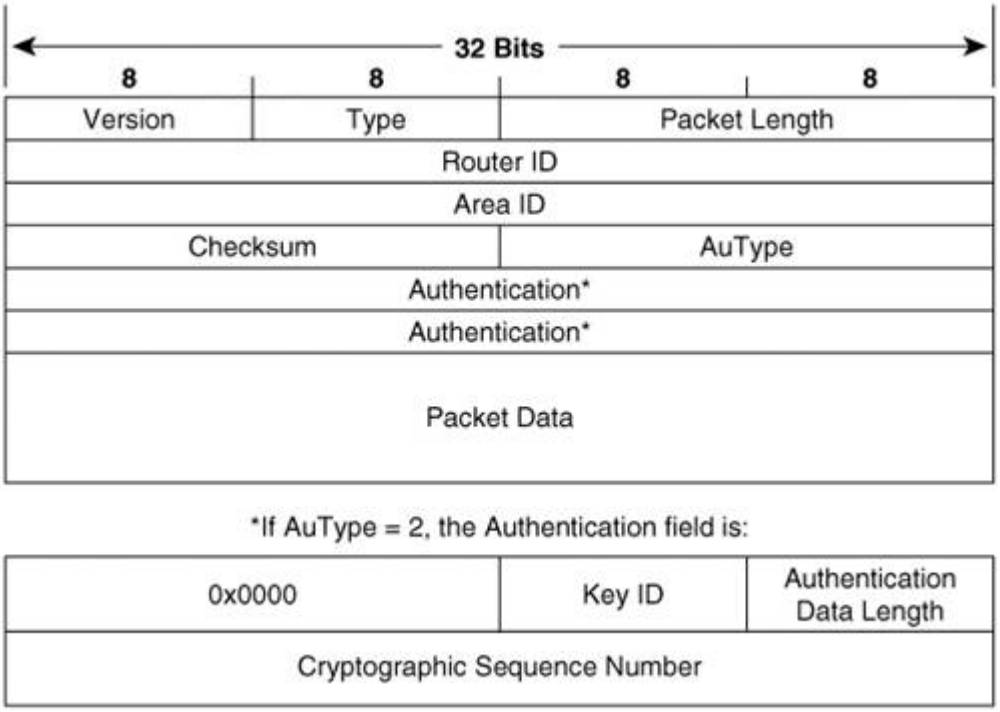
```
IP - Internet Protocol
                 Decode Status : -
                       Version : 4,  Header length : 20
             Type of Service   : 0xC0
                               : 110. .... = Internetwork Control Precedence
                               : ...0 .... = Normal Delay
                               : .... 0... = Normal Throughput
                               : .... .0.. = Normal Reliability
                               : .... ..00 = Reserved
               Total length    : 68 bytes
             Identification    : 10447
           Fragment Control    : 0x00
                               : 0... .... .... .... = Reserved
                               : .0.. .... .... .... = May Fragment
                               : ..0. .... .... .... = Last Fragment
                               : ...0 0000 0000 0000 = Fragment Offset = 0 by
               Time to Live    : 1 seconds/hops
                   Protocol    : 89 (OSPF)
                   Checksum    : 0xA4C4 (Checksum Good)
             Source Address    : 10.8.1.1
        Destination Address    : 224.0.0.5
No IP options
OSPF - Open Shortest Path First Routing Protocol
                 Decode Status : -
```

This section details the five OSPF packet types, beginning with the header. The following section details the LSA types. An Options field is carried in Hello, Database Description packets, and in all LSAs. The format of this field is the same in all cases and is detailed in its own section.

## Packet Header

All OSPF packets begin with a 24-octet header, as shown in Figure 8-30.

Figure 8-30. The OSPF packet header.




- **Version** is the OSPF version number. The OSPF version number is 2. There is an OSPF version 3, created for routing IPv6; OSPFv3 is covered in the next chapter.
- **Type** specifies the packet type following the header. Table 8-7 lists the five packet types by the number appearing in the Type field.

Table 8-7. OSPF packet types.

| Type Code | Description |
|---|---|
| 1 | Hello |
| 2 | Database Description |
| 3 | Link State Request |
| 4 | Link State Update |
| 5 | Link State Acknowledgment |

- **Packet length** is the length of the OSPF packet, in octets, including the header.
- **Router ID** is the ID of the originating router.
- **Area ID** is the area from which the packet originated. If the packet is sent over a virtual link, the Area ID will be 0.0.0.0, the backbone Area ID, because virtual links are considered part of the backbone.
- **Checksum** is a standard IP checksum of the entire packet, including the header.
- **AuType** is the authentication mode being used.

Table 8-8 lists the possible authentication modes.

Table 8-8. OSPF authentication types.

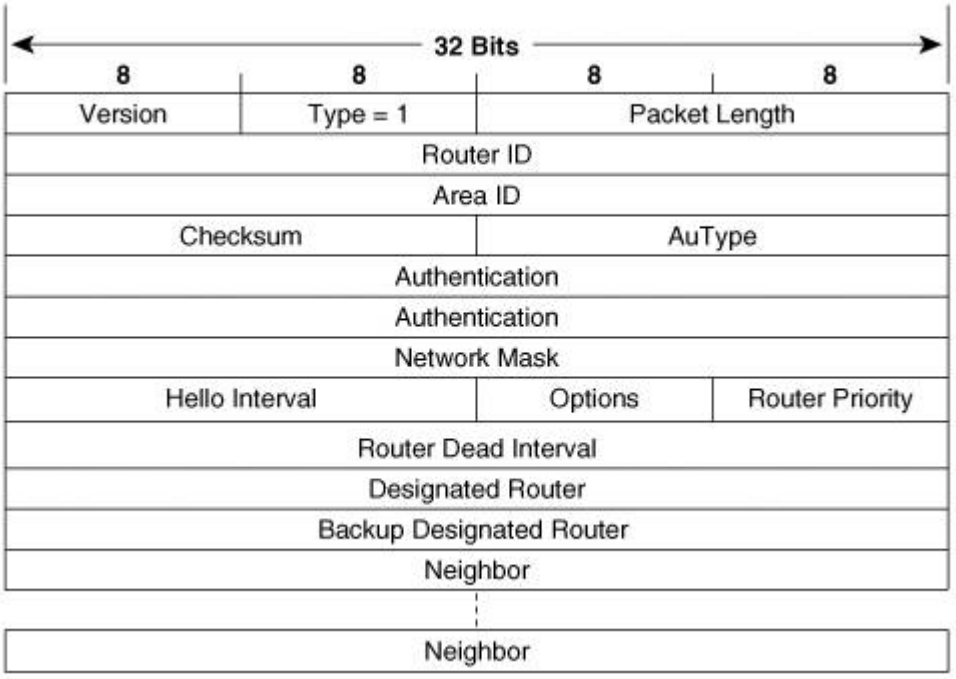
| AuType | Authentication Type |
|---|---|
| 0 | Null (no authentication) |
| 1 | Simple (clear text) Password Authentication |
| 2 | Cryptographic (MD5) Checksum |

- Authentication is the information necessary for the packet to be authenticated by whatever mode is specified in the AuType field. If AuType = 0, the field is not examined and therefore may contain anything. If AuType = 1, the field contains a password of up to 64 bits. If AuType = 2, the Authentication field contains a Key ID, the Authentication Data Length, and a nondecreasing Cryptographic sequence number. The message digest is appended to the end of the OSPF packet, and is not considered part of the packet itself.
- Key ID identifies the authentication algorithm and the secret key used to create the message digest.
- Authentication Data Length specifies the length, in octets, of the message digest appended to the end of the packet.
- Cryptographic Sequence Number is a nondecreasing number used to prevent replay attacks.

## Hello Packet

The Hello packet (Figure 8-31) establishes and maintains adjacencies. The Hello carries parameters on which neighbors must agree in order to form an adjacency.

Figure 8-31. The OSPF Hello packet.

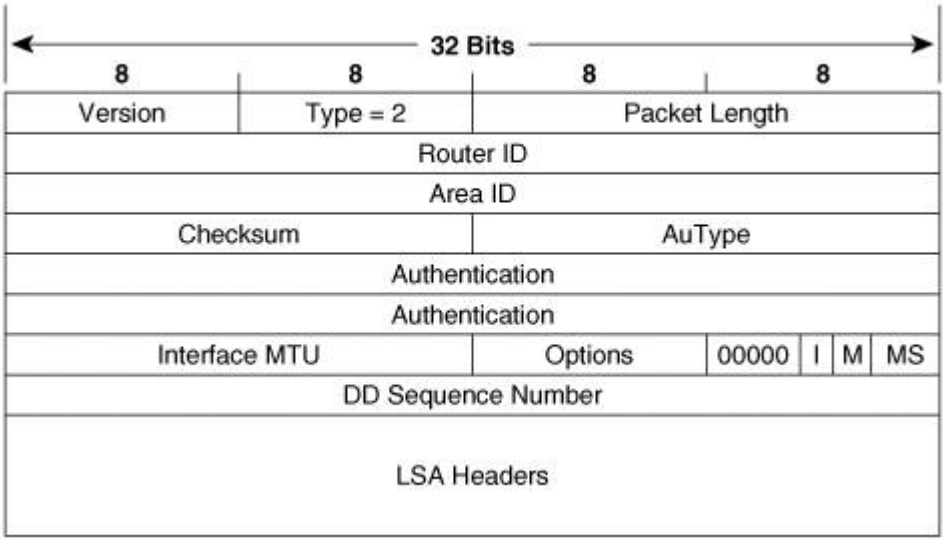



- Network Mask is the address mask of the interface from which the packet was sent. If this mask does not match the mask of the interface on which the packet is received, the packet will be dropped. This technique ensures that routers become neighbors only if they agree on the exact address of their shared network.
- Hello Interval, as discussed earlier, is the period, in seconds, between transmissions of Hello packets on the interface. If the sending and receiving routers don't have the same value for this parameter, they do not establish a neighbor relationship.
- Options are described in "Options Field," later in this chapter. This field is included in the Hello packet to ensure that neighbors have compatible capabilities. A router might reject a neighbor because of a capabilities mismatch.
- Router Priority is used in the election of the DR and BDR. If set to zero, the originating router is ineligible to become the DR or BDR.
- Router Dead Interval is the number of seconds the originating router will wait for a Hello from a neighbor before declaring the neighbor dead. If a Hello is received in which this number does not match the RouterDeadInterval of the receiving interface, the packet is dropped. This technique ensures that neighbors agree on this parameter.
- Designated Router is the IP address of the interface of the DR on the network (not its Router ID). During the DR election process, this may only be the originating router's idea of the DR, not the finally elected DR. If there is no DR (because one has not been elected or because the network type does not require DRs), this field is set to 0.0.0.0.
- Backup DR is the IP address of the interface of the BDR on the network. Again, during the DR election process, this may only be the originating router's idea of the BDR. If there is no BDR, this field is set to 0.0.0.0.
- Neighbor is a recurring field that lists all RIDs of all neighbors on the network from which the originating router has received a valid Hello in the past RouterDeadInterval.

## Database Description Packet

The Database Description packet (Figure 8-32) is used when an adjacency is being established (see "Building an Adjacency," earlier in this chapter). The primary purpose of the DD packet is to describe some or all of the LSAs in the originator's database so that the receiver can determine whether it has a matching LSA in its own database. This is done by listing the headers of the LSAs; the LSA header contains enough information to identify not only a particular LSA but also the most recent instance of that LSA. Because multiple DD packets may be exchanged during the database description process, flags are included for managing the exchange via a master/slave polling relationship.

Figure 8-32. The OSPF Database Description packet.



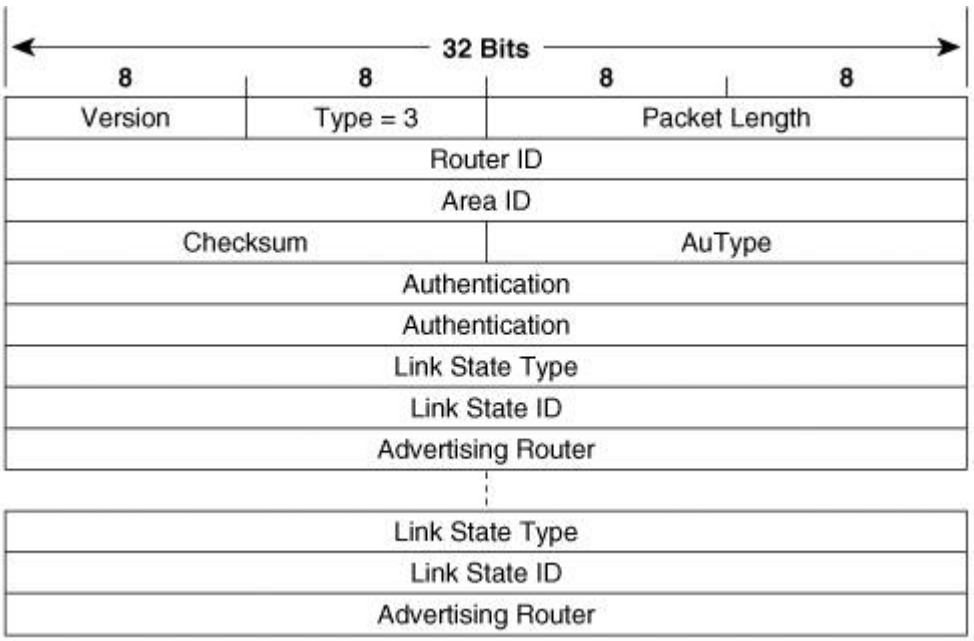

- Interface MTU is the size, in octets, of the largest IP packet that can be sent out the originator's interface without fragmentation. This field is set to 0x0000 when the packet is sent over virtual links.
- Options are described in "Options Field," later in this chapter. The field is included in the Database Description packet so that a router may choose not to forward certain LSAs to a neighbor that doesn't support the necessary capabilities.
- The first five bits of the next octet are unused and are always set to 00000b.
- I-bit, or Initial bit, is set to 1 when the packet is the initial packet in series of DD packets. Subsequent DD packets have I-bit = 0.
- M-bit, or More bit, is set to 1 to indicate that the packet is not the last in a series of DD packets. The last DD packet has M-bit = 0.
- MS-bit, or Master/Slave bit, is set to 1 to indicate that the originator is the master (that is, is in control of the polling process) during a database synchronization. The slave has MS-bit = 0.
- DD Sequence Number ensures that the full sequence of DD packets is received in the database synchronization process. The sequence number is set by the master to some unique value in the first DD packet, and the sequence is incremented in subsequent packets.
- LSA Headers list some or all of the headers of the LSAs in the originator's link-state database. See the section "Link State Header," for a full description of the LSA header; the

header contains enough information to uniquely identify the LSA and the particular instance of the LSA.

## Link State Request Packet

As Database Description packets are received during the database synchronization process, a router takes note of any listed LSAs that are not in its database or are more recent than its own LSA. These LSAs are recorded in the Link State Request list. The router then sends one or more Link State Request packets (Figure 8-33) asking the neighbor for its copy of the LSAs on the Link State Request list. Note that the packet uniquely identifies the LSA by Type, ID, and Advertising Router fields of its header, but it does not request a specific instance of the LSA (identified by the header's sequence number, checksum, and age). Therefore, the request is for the most recent instance of the LSA, whether the requester is aware of that instance or not. This procedure protects against a situation in which the neighbor might acquire or originate a more recent copy of the LSA between the time it last described the LSA and the time a copy of it is requested.

Figure 8-33. The OSPF Link State Request packet.



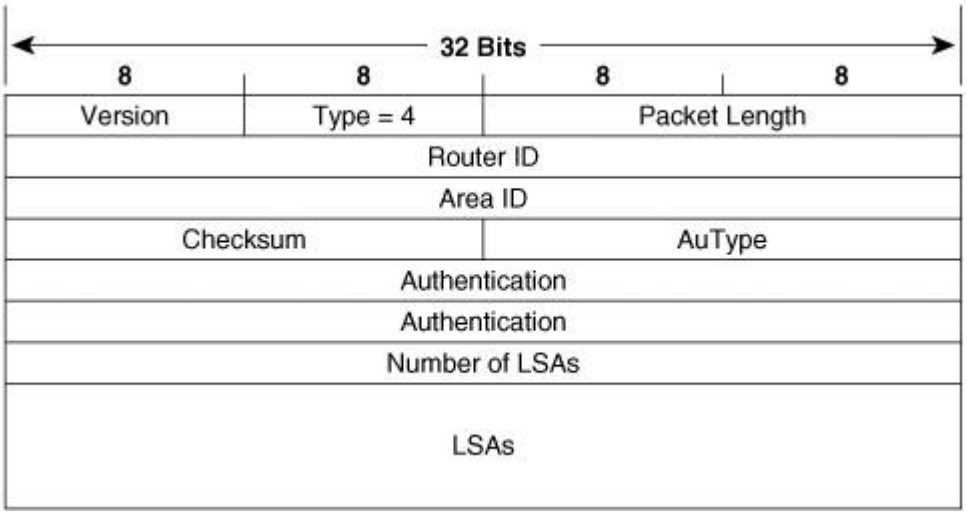

- Link State Type is the LS type number, which identifies the LSA as a Router LSA, Network LSA, and so on. Type numbers are listed in Table 8-4.
- Link State ID is a type-dependent field of the LSA header. See the section "Link State Header" and the LSA-specific sections for a full description of how the various LSAs use this field.
- Advertising Router is the Router ID of the router that originated the LSA.

## Link State Update Packet

The Link State Update packet, shown in Figure 8-34, is used in the flooding of LSAs and to send LSAs in response to Link State Requests. Recall that OSPF packets do not leave the network on

which they were originated. Consequently, a Link State Update packet, carrying one or many LSAs, carries the LSAs only to the originating router's connected neighbors. The receiving neighbor is responsible for re-encapsulating the appropriate LSAs in new LS Update packets for further flooding to its own neighbors.

Figure 8-34. The OSPF Link State Update packet.

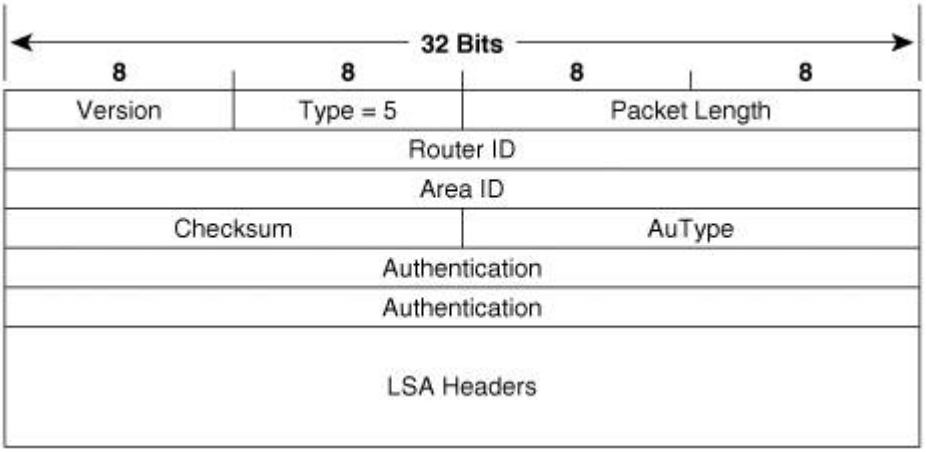



- Number of LSAs specifies the number of LSAs included in this packet.
- LSAs are the full LSAs as described in OSPF LSA formats. Each update can carry multiple LSAs, up to the maximum packet size allowed on the link.

## Link State Acknowledgment Packet

Link State Acknowledgment packets are used to make the flooding of LSAs reliable. Each LSA received by a router from a neighbor must be explicitly acknowledged in a Link State Acknowledgement packet. The LSA being acknowledged is identified by including its header in the LS ACK packet, and multiple LSAs can be acknowledged in a single packet. As Figure 8-35 shows, the LS ACK packet consists of nothing more than an OSPF packet header and a list of LSA headers.

Figure 8-35. The OSPF Link State Acknowledgment packet.



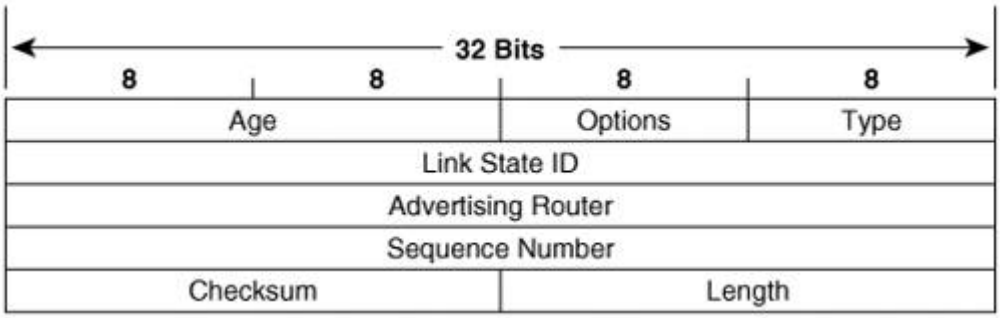

## OSPF LSA Formats

This section details the fields of LSA types 1-5 and 7. The Group Membership LSA (type 6) is not discussed because MOSPF is not covered in this book. Similarly, LSA types 8 through 11 are not covered because they either are not in general deployment (type 8) or because they are used to support capabilities that are outside of the scope of this book.

### LSA Header

The LSA header (Figure 8-36) begins all LSAs and is also used by itself in Database Description and Link State Acknowledgment packets. Three fields in the header uniquely identify every LSA: the Type, Link State ID, and Advertising Router. Additionally, three other fields uniquely identify the most recent instance of an LSA: the Age, Sequence Number, and Checksum.

Figure 8-36. The OSPF LSA header.



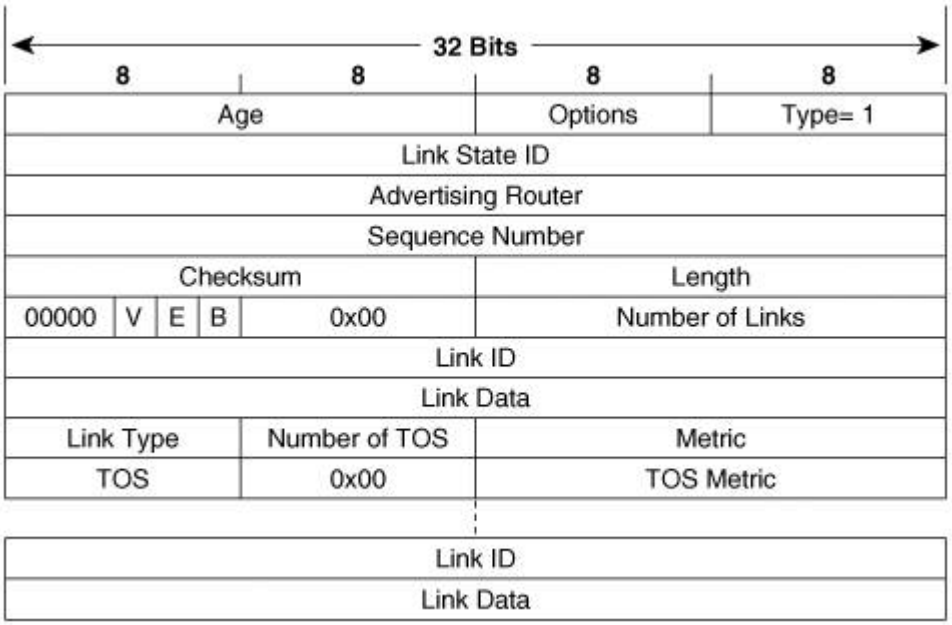

- Age is the time, in seconds, since the LSA was originated. As the LSA is flooded, the age is incremented by InfTransDelay seconds at each router interface it exits. The age is also incremented in seconds as it resides in a link-state database.
- Options is described in the section "Options Field." In the LSA header, the Options field specifies the optional capabilities supported by the portion of the OSPF domain described by the LSA.
- Type is the LSA type. The type codes are shown in Table 8-4.

- Link State ID identifies the portion of the OSPF domain being described by the LSA. The specific usage of this field varies according to the LSA type; the descriptions of each LSA include a description of how the LSA uses this field.
- Advertising Router is the router ID of the router that originated the LSA.
- Sequence Number is incremented each time a new instance of the LSA is originated, so that other routers can identify the most recent instance of the LSA.
- Checksum is the Fletcher checksum of the complete contents of the LSA except for the Age field. If the Age field were included, the checksum would have to be recalculated every time the age was incremented.
- Length is the number of octets of the LSA, including the header.

## Router LSA

A Router LSA (Figure 8-37) is produced by every router. It lists a router's links, or interfaces, along with the state and the outgoing cost of each link and any known neighbors on the link. These LSAs are flooded only within the area in which they are originated. The command show ip ospf database router (refer to Example 8-11) lists the Router LSAs in a database. Note that Router LSAs advertise host routes as stub networks; the Link ID field carries the host IP address, and the Link Data field carries the host address mask of 255.255.255.255.

Figure 8-37. OSPF Router LSA.




- Link State ID for Router LSAs is the originating router's Router ID.
- V, or Virtual Link Endpoint bit, is set to one when the originating router is an endpoint of one or more fully adjacent virtual links having the described area as the transit area.
- E, or External bit, is set to one when the originating router is an ASBR.
- B, or Border bit, is set to one when the originating router is an ABR.

- Number of Links specifies the number of router links the LSA describes. The Router LSA must describe all of the originating router's links, or interfaces, to the area in which the LSA is flooded.

  Subsequent fields in the Router LSA describe each link and appear one or more times, corresponding to the number in the Number of Links field. This discussion covers the Link Type field first, although that field does not appear until after the Link Data field. Understanding link type first is important because the descriptions of the Link ID and Link Data fields vary according to the value of the Link Type field.

- Link Type describes the general type of connection the link provides. Table 8-9 lists the possible values of the field and the associated connection types.

Table 8-9. Link type values.

| Link Type | Connection |
|---|---|
| 1 | Point-to-point connection to another router |
| 2 | Connection to a transit network |
| 3 | Connection to a stub network |
| 4 | Virtual link |

- Link ID identifies the object to which the link connects. This is dependent on the link type, as shown in Table 8-10. Note that when the connected object is another router, the Link ID is the same as the Link State ID in the header of the neighboring router's LSA. During the routing table calculation, this value is used to find the neighbor's LSA in the link-state database.

Table 8-10. Link ID values.

| Link Type | Value of Link ID Field |
|---|---|
| 1 | Neighboring router's Router ID |
| 2 | IP address of the DR's interface |
| 3 | IP network or subnet address |
| 4 | Neighboring router's Router ID |

- Link Data also depends on the value of the Link Type field, as shown in Table 8-11.

Table 8-11. Link data values.

| Link Type | Value of Link Data Field |
|---|---|
| 1 | IP address of the originating router's interface to the network[*] |

| Link Type | Value of Link Data Field |
|---|---|
| 2 | IP address of the originating router's interface to the network |
| 3 | Network's IP address or subnet mask |
| 4 | The MIB-II ifIndex value for the originating router's interface |

[*] If the point-to-point link is unnumbered, this field will instead carry the MIB-II ifIndex value of the interface.

- Number of TOS specifies the number of Type of Service metrics listed for this link. Although TOS is no longer supported in RFC 2328, the TOS fields are still included for backward compatibility with earlier OSPF implementations. If no TOS metrics are associated with a link, this field is set to 0x00.

- Metric is the cost of the link (interface).

  The next two fields are associated with a link corresponding to the number (#) of TOS field. For example, if # of TOS = 3, there will be three 32-bit words containing three instances of these fields. If # of TOS = 0, there will be no instances of these fields.

  Note that Cisco supports only TOS = 0.

- TOS specifies the Type of Service to which the following metric refers.[24] Table 8-12 lists the TOS values (as specified in RFC 1349), the bit values of the corresponding TOS field in the IP header, and the corresponding value used in the OSPF TOS field.

[24] Philip Almquist, "Type of Service in the Internet Protocol Suite," RFC 1349, July 1992.

Table 8-12. OSPF TOS values.

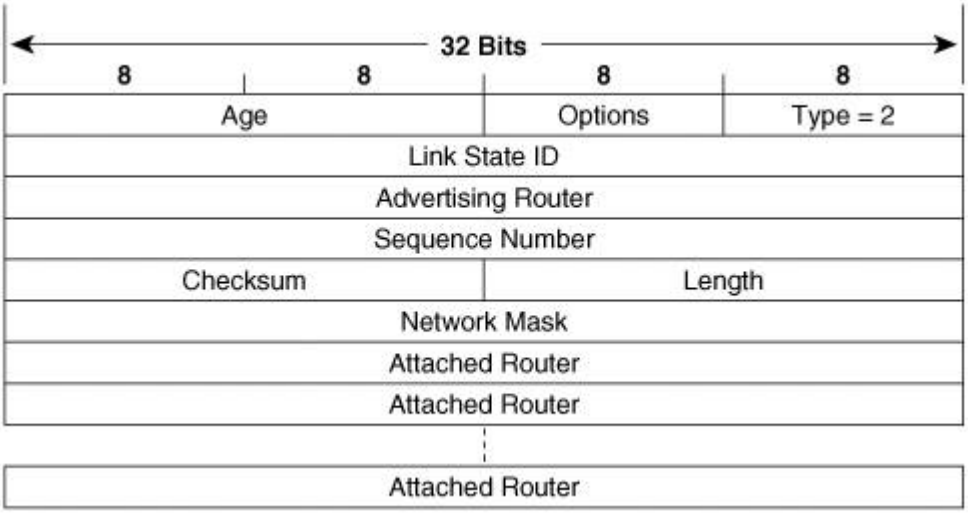
| RFC TOS Value | IP Header TOS Field | OSPF TOS Encoding |
|---|---|---|
| Normal Service | 0000 | 0 |
| Minimize Monetary Cost | 0001 | 2 |
| Maximize Reliability | 0010 | 4 |
| Maximize Throughput | 0100 | 8 |
| Minimize Delay | 1000 | 16 |

- TOS Metric is the metric associated with the specified TOS value.

## Network LSA

Network LSAs (Figure 8-38) are originated by DRs. These LSAs advertise the multi-access network, and all routers (including the DR) attached to the network. Like Router LSAs, Network LSAs are flooded only within the originating area. The command show ip ospf database network (Example 8-12) is used to observe a Network LSA.

Figure 8-38. The OSPF Network LSA.

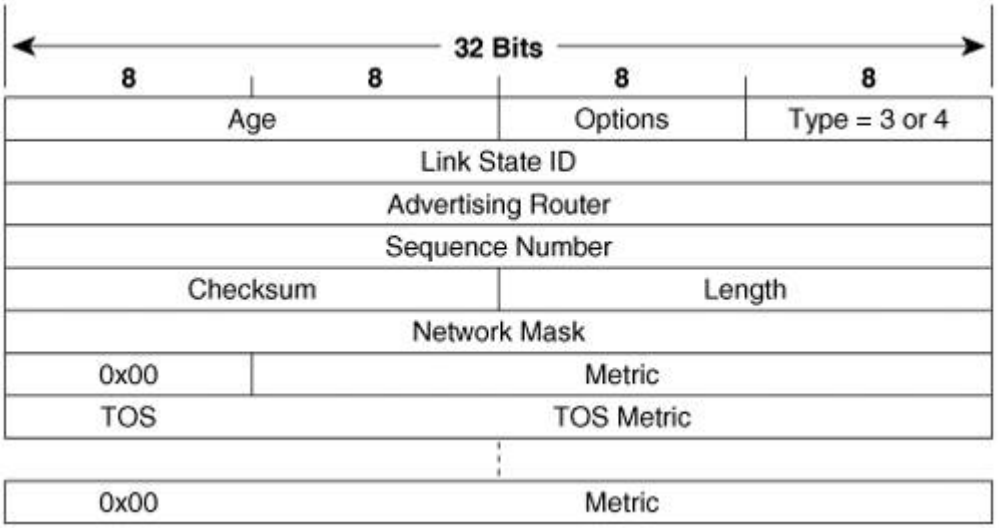



- Link State ID for Network LSAs is the IP address of the DR's interface to the network.
- Network Mask specifies the address or subnet mask used on this network.
- Attached Router lists the Router IDs of all routers on the multi-access network that are fully adjacent with the DR, and the Router ID of the DR itself. The number of instances of this field (and hence the number of routers listed) can be deduced from the LSA header's Length field.

## Network and ASBR Summary LSAs

The Network Summary LSA (type 3) and the ASBR Summary LSA (type 4) have an identical format, shown in Figure 8-39. The only difference in field contents is the Type and the Link State ID. ABRs produce both types of Summary LSA; Network Summary LSAs advertise networks external to an area (including default routes), whereas ASBR Summary LSAs advertise ASBRs external to an area. Both types are flooded only into a single area. The Network Summary LSAs in a router's database can be observed with the command show ip ospf database summary (Example 8-13), and ASBR Summary LSAs can be observed with show ip ospf database asbr-summary (Example 8-14).

Figure 8-39. The OSPF Summary LSA. The format is the same for both type 3 and type 4 Summary LSAs.




- Link State ID, for type 3 LSAs, is the IP address of the network or subnet being advertised. If the LSA is type 4, the Link State ID is the Router ID of the ASBR being advertised.
- Network Mask is the address or subnet mask of the network being advertised in type 3 LSAs. In type 4 LSAs, this field has no meaning and is set to 0.0.0.0.

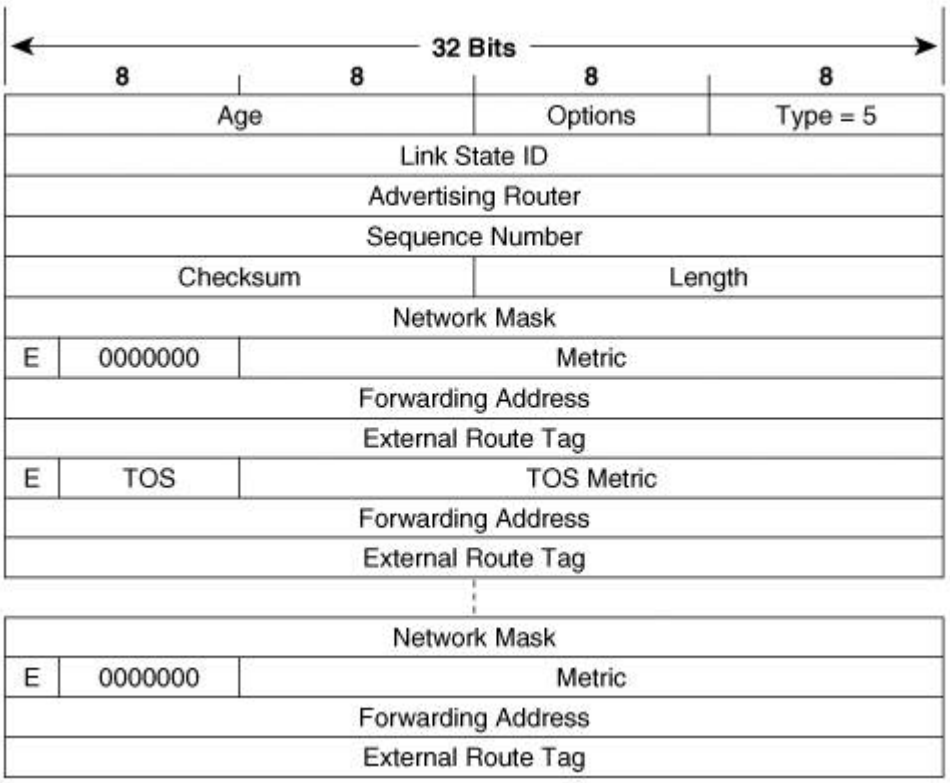
  If a type 3 LSA is advertising a default route, both the Link State ID and the Network Mask fields will be 0.0.0.0.
- Metric is the cost of the route to this destination.

  The TOS and TOS Metric fields are optional and are described in "Router LSA." Again, Cisco supports only TOS = 0.

## Autonomous System External LSA

Autonomous System External LSAs (Figure 8-40) are originated by ASBRs. These LSAs are used to advertise destinations external to the OSPF autonomous system, including default routes to external destinations, and are flooded into all nonstub areas of the OSPF domain. The command show ip ospf database external is used to display AS External LSAs (Example 8-15).

Figure 8-40. The OSPF Autonomous System External LSA.




- Link State ID for AS External LSAs is the IP address of the destination.
- Network Mask is the address or subnet mask for the destination being advertised.

  If the type 5 LSA is advertising a default route, the Link State ID and the Network Mask are both 0.0.0.0.
- E, or External Metric bit, specifies the type of external metric to be used with this route. If the E-bit is set to 1, the metric type is E2. If the E-bit = 0, the metric type is E1. See the section "Path Types," earlier in this chapter, for more information on E1 and E2 external metrics.
- Metric is the cost of the route, as set by the ASBR.
- Forwarding Address is the address to which packets for the advertised destination should be forwarded. If the forwarding address is 0.0.0.0, packets are forwarded to the originating ASBR.
- External Route Tag is an arbitrary tag that may be applied to the external route. This field is not used by the OSPF protocol itself, but is instead provided for external route management. The setting and use of such tags are discussed in Chapter 14, "Route Maps."

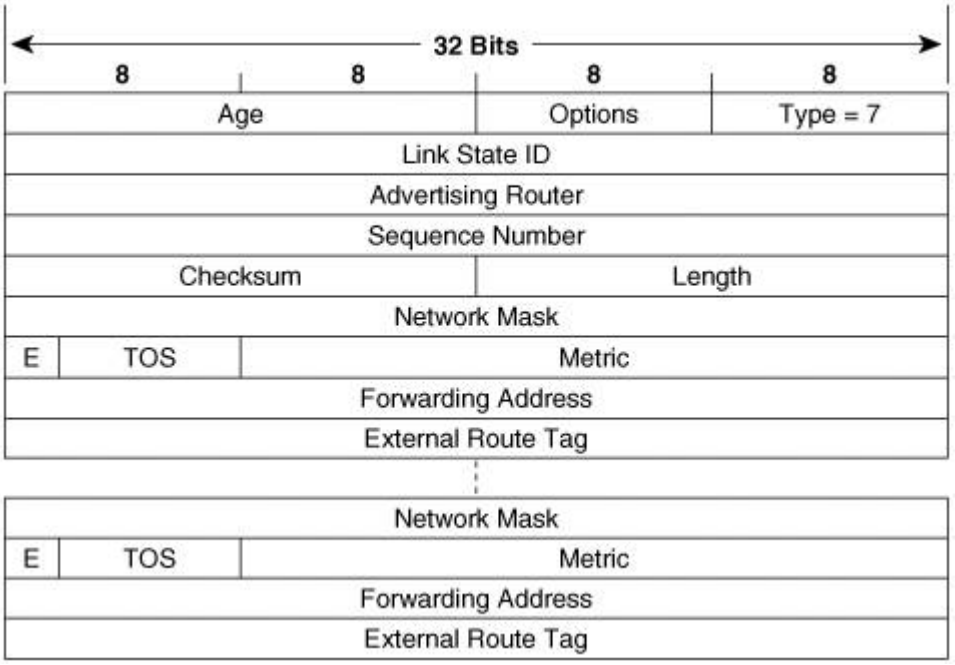
Optionally, TOS fields may be associated with the destination. These fields are the same as discussed previously, except each TOS metric also has its own E-bit, Forwarding Address, and External Route Tag.

## NSSA External LSA

NSSA External LSAs are originated by ASBRs within an NSSA (not-so-stubby area). All fields of the NSSA External LSA (Figure 8-41) are identical to an AS External LSA's fields, with the exception of the Forwarding Address field. Unlike AS External LSAs, which are flooded throughout an OSPF autonomous system, NSSA external LSAs are flooded only within the not-

so-stubby area in which it was originated. The command show ip ospf database nssa-external is used to display NSSA External LSAs (Example 8-16).

Figure 8-41. OSPF NSSA External LSA.




*Forwarding Address*, if the network between the NSSA ASBR and the adjacent autonomous system is advertised as an internal route, is the next hop address on the network. If the network is not advertised as an internal route, the forwarding address will be the NSSA ASBR's Router ID.

## Options Field

The Options field (Figure 8-42) is present in every Hello and Database Description packet and in every LSA. The Options field allows routers to communicate their optional capabilities to other routers.

Figure 8-42. The OSPF Options field.

*DN* is used with MPLS-based layer 3 Virtual Private Networks (VPN), commonly called RFC 2547 VPNs after the RFC that specifies them. When a route is learned from a customer network via OSPF, is advertised across the RFC 2547 VPN using Multiprotocol BGP, and then is advertised back to a customer network via OSPF, a loop can occur in which the OSPF route is redistributed back to the VPN provider network in BGP. The DN bit prevents this looping. When the DN bit is set in a type 3, 5, or 7 LSA, the receiving router cannot use that LSA in its OSPF route calculations.

*O* is set to indicate that the originating router supports Opaque (type 9, 10, and 11) LSAs.

*DC* is set when the originating router is capable of supporting OSPF over demand circuits.

*EA* is set when the originating router is capable of receiving and forwarding External Attributes LSAs. These LSAs are not in general usage and are not covered in this book.

*N* is used only in Hello packets. A router sets N-bit = 1 to indicate support for NSSA External LSAs. If N-bit = 0, the router will not accept or send these LSAs. Neighboring routers with mismatched N-bits will not become adjacent; this restriction ensures that all routers in an area support NSSA capabilities equally. If the N-bit = 1, the E-bit must be 0.

*P* is used only in NSSA External LSA headers. (For this reason, the N- and P-bit can use the same position.) This bit tells the ABR of a not-so-stubby area to translate type 7 LSAs into type 5 LSAs.

*MC* is set when the originating router is capable of forwarding IP multicast packets. This bit is used by MOSPF.

*E* is set when the originating router is capable of accepting AS External LSAs. It will be set to 1 in all AS External LSAs and in all LSAs originated in the backbone and nonstub areas. E-bit = 0 in all LSAs originated within a stub area. Additionally, the bit is used in the Hello packet to indicate an interface's capability of sending and receiving type 5 LSAs. Neighboring routers with mismatched E-bits will not become adjacent; this restriction ensures that all routers in an area support stub capabilities equally.

MT, when set, indicates that the originating router supports Multitopology OSPF (MT-OSPF). MT-OSPF, as of this writing, is only a proposal and has not yet found general adoption.

Older OSPF standards specified that the options bit position now occupied by the MT bit was the *T* bit. T was set when the originating router is capable of supporting TOS. However, because the TOS capability was never deployed, the T bit was also never used.

PREV NEXT

# Configuring OSPF

The many options and configuration variables available to OSPF frequently make it the IGP of choice in large IP networks. However, the opinion is occasionally expressed that OSPF configuration is "too complex" to be a good choice for small internets. This is nonsense. As the first case study shows, getting a basic OSPF configuration up and running involves only a few extra keystrokes in the network command; if the operation of OSPF is reasonably well understood, these extra keystrokes will be intuitive.

## Case Study: A Basic OSPF Configuration

The three steps necessary to begin a basic OSPF process are

Step 1. Determine the area to which each router interface will be attached.

Step 2. Enable OSPF with the command router ospf process-id.

Step 3. Specify the interfaces on which to run OSPF, and their areas, with the network area command.

Unlike the process ID associated with IGRP and EIGRP, the OSPF process ID is not an autonomous system number. The process ID can be any positive integer and has no significance outside the router on which it is configured. Cisco IOS allows multiple OSPF processes to run on the same router;[25] the process ID merely distinguishes one process from another within the device.

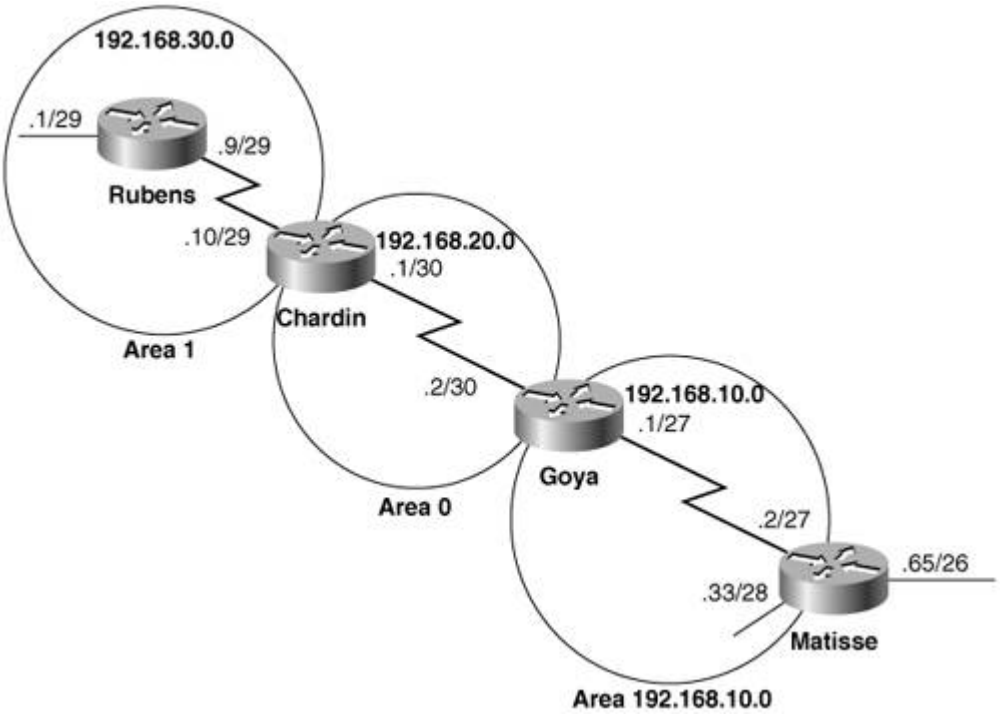
[25] Although the use of multiple processes on one router is possible, it is highly discouraged because of the demands that the multiple databases will place on router resources.

The network command used with RIP allows only the specification of a major network address. If some interfaces within the network should not run the routing protocol, the passive-interface command has to be used with those protocols. As is the network command used with the wildcard mask for EIGRP, the network area command is much more flexible then the network command used for RIP and EIGRP before the wildcard option was introduced, reflecting the fully classless nature of OSPF. Any address range can be specified with an (address, inverse mask) pair. The inverse mask is the same as the inverse mask used with access lists.[26] The area can be specified in either decimal or dotted decimal.

[26] See Appendix B, "Tutorial: Access Lists," for a tutorial on the use of inverse masks.

Figure 8-43 shows an OSPF network. Note that each area has an assigned IP address from which its subnets are derived. Limiting an area to a single address or subnet is not necessary, but doing so has significant advantages, as will be seen in a later case study on address summarization. Note also that this example is designed to demonstrate the configuration of multiple areas. In "real life," it would be much wiser to put such a small network within a single area. Further, that single area does not have to be area 0. The rule is that all areas must connect to the backbone; therefore, a backbone area is needed only if there is more than one area.

Figure 8-43. Chardin and Goya are ABRs; Rubens and Matisse are Internal Routers.




Each of the four routers in Figure 8-43 is configured differently to demonstrate the flexibility of the network area command. The configurations are displayed in Example 8-19 through Example 8-22:

### Example 8-19. Rubens's OSPF network area configuration.

```
router ospf 10
 network 0.0.0.0 255.255.255.255 area 1
```

### Example 8-20. Chardin's OSPF network area configuration.

```
router ospf 20
 network 192.168.30.0 0.0.0.255 area 1
 network 192.168.20.0 0.0.0.255 area 0
```

### Example 8-21. Goya's OSPF network area configuration.

```
router ospf 30
 network 192.168.20.0 0.0.0.3 area 0.0.0.0
 network 192.168.10.0 0.0.0.31 area 192.168.10.0
```

### Example 8-22. Matisse's OSPF network area configuration.

```
router ospf 40
 network 192.168.10.2 0.0.0.0 area 192.168.10.0
 network 192.168.10.33 0.0.0.0 area 192.168.10.0
```

The first thing to note is that the process IDs are different for each router. Usually these numbers

are the same across an internet for consistency of configuration. Here the process IDs are configured differently merely to demonstrate that they have no meaning outside of the router. These four differently numbered processes are able to communicate.

The next thing to notice is the format of the network area command. Following the network portion is an IP address and an inverse mask. When the OSPF process first becomes active, it will "run" the IP addresses of all active interfaces against the (address, inverse mask) pair of the first network statement. All interfaces that match will be assigned to the area specified by the area portion of the command. The process will then run the addresses of any interfaces that did not match the first network statement against the second network statement. The process of running IP addresses against network statements continues until all interfaces have been matched or until all network statements have been used. It is important to note that this process is consecutive, beginning with the first network statement. As a result, the order of the statements can be important, as is shown in the troubleshooting section.

Rubens's network statement will match all interfaces on the router. The address 0.0.0.0 is really just a placeholder; the inverse mask of 255.255.255.255 is the element that does all of the work here. With "don't care" bits placed across the entire four octets, the mask will find a match with any address and place the corresponding interface into area 1. This method provides the least precision in controlling which interfaces will run OSPF.

Chardin is an ABR between area 1 and area 0. This fact is reflected in its network statements. Here the (address, inverse mask) pairs will place any interface that is connected to any subnet of major network 192.168.30.0 in area 1 and any interface that is connected to any subnet of major network 192.168.20.0 in the backbone area.

Goya is also an ABR. Here the (address, inverse mask) pairs will match only the specific subnets configured on the two interfaces. Notice also that the backbone area is specified in dotted decimal Both this format and the decimal format used at Chardin will cause the associated area fields of the OSPF packets to be 0x00000000, so they are compatible.

Matisse has one interface, 192.168.10.65/26, which is not running OSPF. The network statements for this router are configured to the individual interface addresses, and the inverse mask indicates that all 32 bits must match exactly. This method provides the most precise control over which interfaces will run OSPF.

Finally, note that although Matisse's interface 192.168.10.65/26 is not running OSPF, that address is numerically the highest on the router. As a result, Matisse's Router ID is 192.168.10.65 (Example 8-23).

## Example 8-23. The command *show ip ospf process-id* displays process-specific information.

```
Matisse#show ip ospf 40
 Routing Process "ospf 40" with ID 192.168.10.65
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Supports Link-local Signaling (LLS)
 Initial SPF schedule delay 5000 msecs
 Minimum hold time between two consecutive SPFs 10000 msecs
 Maximum wait time between two consecutive SPFs 10000 msecs
 Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
 LSA group pacing timer 240 secs
 Interface flood pacing timer 33 msecs
 Retransmission pacing timer 66 msecs
 Number of external LSA 0. Checksum Sum 0x000000
 Number of opaque AS LSA 0. Checksum Sum 0x000000
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 1. 1 normal 0 stub 0 nssa
```

```
 External flood list length 0
    Area 192.168.10.0
        Number of interfaces in this area is 2
        Area has no authentication
        SPF algorithm last executed 00:47:42.792 ago
        SPF algorithm executed 4 times
        Area ranges are
        Number of LSA 5. Checksum Sum 0x02A444
        Number of opaque link LSA 0. Checksum Sum 0x000000
        Number of DCbitless LSA 0
        Number of indication LSA 0
        Number of DoNotAge LSA 0
        Flood list length 0
```

## Case Study: Setting Router IDs with Loopback Interfaces

Suppose router Matisse from Figure 8-43 has been configured in a staging center and then sent to the field to be installed. During the bootup, the router reports that it cannot allocate a Router ID, and it seems to report the `network area` commands as configuration errors (Example 8-24). Worse, the OSPF commands are no longer in the running configuration.

### Example 8-24. OSPF will not boot if it cannot find an active IP address for its Router ID.

```
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-J1S3-M), Version 12.3(6), RELEASE SOFTWARE (fc3)
Copyright (c) 1986-2004 by Cisco Systems, Inc.
Compiled Wed 11-Feb-04 19:24 by kellythw
Image text-base: 0x80008098, data-base: 0x8199F778

cisco 2621 (MPC860) processor (revision 0x200) with 61440K/4096K bytes of memory


Processor board ID JAD05090PW2 (1141326406)
M860 processor: part number 0, mask 49
Bridging software.
X.25 software, Version 3.0.0.
TN3270 Emulation software.
2 FastEthernet/IEEE 802.3 interface(s)
1 Serial network interface(s)
32K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)


 network 192.168.10.2 0.0.0.0 area 192.168.10.0
    ^
% Invalid input detected at '^' marker.
 network 192.168.10.33 0.0.0.0 area 192.168.10.0
    ^
% Invalid input detected at '^' marker.



Press RETURN to get started!


%OSPF-4-NORTRID: OSPF process 40 cannot start. There must
```

```
be at least one "up" IP interface, for OSPF to use as router ID
```

The problem here is that during bootup all the interfaces on the router were administratively shut down. If OSPF cannot find an active IP address for its Router ID, it cannot start. And if the OSPF process isn't active, the subsequent network area commands will be invalid.

The solution to this problem (assuming you have a valid reason for having all physical interfaces in shutdown) is to use a loopback interface. The loopback interface, which is a virtual, software-only interface, is always up. Therefore, its IP address is always available.

The more common reason for using loopback interfaces on OSPF routers is that the interfaces allow the network administrator to control the Router IDs. When the OSPF process looks for a Router ID, OSPF will prefer the address of a loopback interface over the addresses of all physical interfaces, regardless of the numerical order. If there are multiple loopback interfaces with IP addresses, OSPF will choose the numerically highest loopback address.

Controlling the Router IDs so that individual OSPF routers are more easily identified facilitates management and troubleshooting. The Router IDs are usually managed by one of two methods:

- Set aside a legitimate network or subnet address to be used strictly for Router IDs.
- Use a "bogus" IP address range.

The first method has the disadvantage of using up the assigned network address space. The second method will preserve the legitimate addresses, but one must remember that what is bogus in one internet is legitimate in another. Using easily recognized addresses such as 1.1.1.1, 2.2.1.1, and so on is fine as long as you remember that these are not public addresses. Care must be taken that the bogus addresses do not leak out to the public Internet.

The configurations of the last section are modified to use loopback addresses as displayed in Example 8-25 through Example 8-28.

### Example 8-25. A Loopback interface is added to Rubens's configuration.

```
interface Loopback0
 ip address 192.168.50.1 255.255.255.255
!
router ospf 10
 network 192.168.30.0 0.0.0.255 area 1
```

### Example 8-26. A Loopback interface is added to Chardin's configuration

```
interface Loopback0
 ip address 192.168.50.2 255.255.255.255
!
router ospf 20
 network 192.168.30.0 0.0.0.255 area 1
 network 192.168.20.0 0.0.0.255 area 0
```

### Example 8-27. A Loopback interface is added to Goya's configuration.

```
interface Loopback0
 ip address 192.168.50.3 255.255.255.255
!
router ospf 30
```

```
 network 192.168.20.0 0.0.0.3 area 0.0.0.0
 network 192.168.10.0 0.0.0.31 area 192.168.10.0
```

Example 8-28. A Loopback interface is added to Matisse's configuration.

```
interface Loopback0
 ip address 192.168.50.4 255.255.255.255
!
router ospf 40
 network 192.168.10.2 0.0.0.0 area 192.168.10.0
 network 192.168.10.33 0.0.0.0 area 192.168.10.0
```

For this example, the network address 192.168.50.0 has been set aside for exclusive use as Router IDs. Router IDs are thus easily distinguished from other IP addresses in this internet.

The first thing to note about this configuration is the address masks used with the loopback addresses: Each mask is configured as a host address. This step is not really necessary, because OSPF treats a loopback interface as a stub host; whatever (address, mask) pair is configured, the address of the loopback interface will be advertised as a host route. The host mask is used merely to keep things neat, and to reflect the way in which the address is advertised.

However, the second point of interest makes the first somewhat irrelevant. Remember that OSPF does not have to be running on an interface for its IP address to be used as the Router ID. In fact, having OSPF advertise the loopback addresses just creates unnecessary LSAs. In the example shown, notice that the `network area` statements do not refer to the loopback addresses. In fact, the configuration at Rubens had to be changed. Rubens's previous command, `network 0.0.0.0 255.255.255.255 area 1`, would have picked up the loopback address.

In addition to aiding management and troubleshooting, using loopback interfaces will also make an OSPF internet more stable. In some early versions of IOS, if a physical interface from which the Router ID was taken experiences a hardware failure,[27] if the interface is administratively shut down, or if the IP address is inadvertently deleted, the OSPF process must acquire a new Router ID. Therefore, the router must prematurely age and flood its old LSAs and then flood LSAs containing the new ID. A loopback interface has no hardware components to fail. The current IOS behavior is to obtain the Router ID; if the interface associated with the Router ID fails or the IP address is deleted, the router retains the Router ID until the router is reloaded or the OSPF process is reset.

[27] Merely disconnecting the interface will not cause the Router ID to change.

Another option is to manually assign a Router ID to the router with the OSPF command `router-id` As with the Router ID address assignment using loopback interfaces, you can arbitrarily choose an IP address to use as the value of the `router-id` command. If a Router ID is configured using this command, this command's value will become the router ID when the OSPF process is reset or the router is reloaded. When the router is reloaded, however, there still has to be an interface with an IP address that comes up before the OSPF process can start. After the OSPF process starts, the address assigned with the `router-id` command will become the Router ID.

## Case Study: Domain Name Service Lookups

Loopback interfaces simplify the management and troubleshooting of OSPF internets by providing predictable Router IDs. This simplification can be taken even further by recording the Router IDs in a Domain Name Service (DNS) database. The router can then be configured to consult the server address-to-name mappings, or Reverse DNS lookups, and then display the routers by name instead of by Router ID (Example 8-29).

### Example 8-29. OSPF can be configured to use DNS to map Router IDs to names for use in some show commands.

```
Goya#show ip ospf neighbor

Neighbor ID     Pri   State      Dead Time    Address       Interface
chardin           0   FULL/  -   00:00:36     192.168.20.1  Serial0/0.2
matisse           0   FULL/  -   00:00:34     192.168.10.2  Serial0/0.1
Goya#show ip ospf database

            OSPF Router with ID (192.168.50.3) (Process ID 30)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
192.168.50.2    chardin         78          0x80000007 0x005A70 2
192.168.50.3    goya            78          0x80000008 0x004C7B 2

                Summary Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age         Seq#       Checksum
192.168.10.0    goya            74          0x80000001 0x00B356
192.168.10.32   goya            54          0x80000001 0x00DCFB
192.168.30.0    chardin         85          0x80000001 0x007766
192.168.30.8    chardin         100         0x80000001 0x001DB9

                Router Link States (Area 192.168.10.0)


Link ID         ADV Router      Age         Seq#       Checksum Link count
192.168.50.3    goya            68          0x80000007 0x0024D3 2
192.168.50.4    matisse         67          0x80000007 0x00E080 3
--More-
```

Goya was configured to perform DNS lookups as shown in Example 8-30.

### Example 8-30. Goya is configured to perform DNS lookups.

```
ip name-server 172.19.35.2
!
ip ospf name-lookup
```

The first command specifies the address of the DNS server, and the second enables the OSPF process to perform DNS lookups. In some cases, a router is identified by an interface address instead of a Router ID. Adding entries to the DNS database for the router interfaces, such as *rubens-e0*, allows the interfaces to also be identified by name while differentiating them from the Router IDs.

The address of the name server used in this example does not belong to one of the subnets shown in Figure 8-43. The method by which this network is reached is the subject of the next case study.

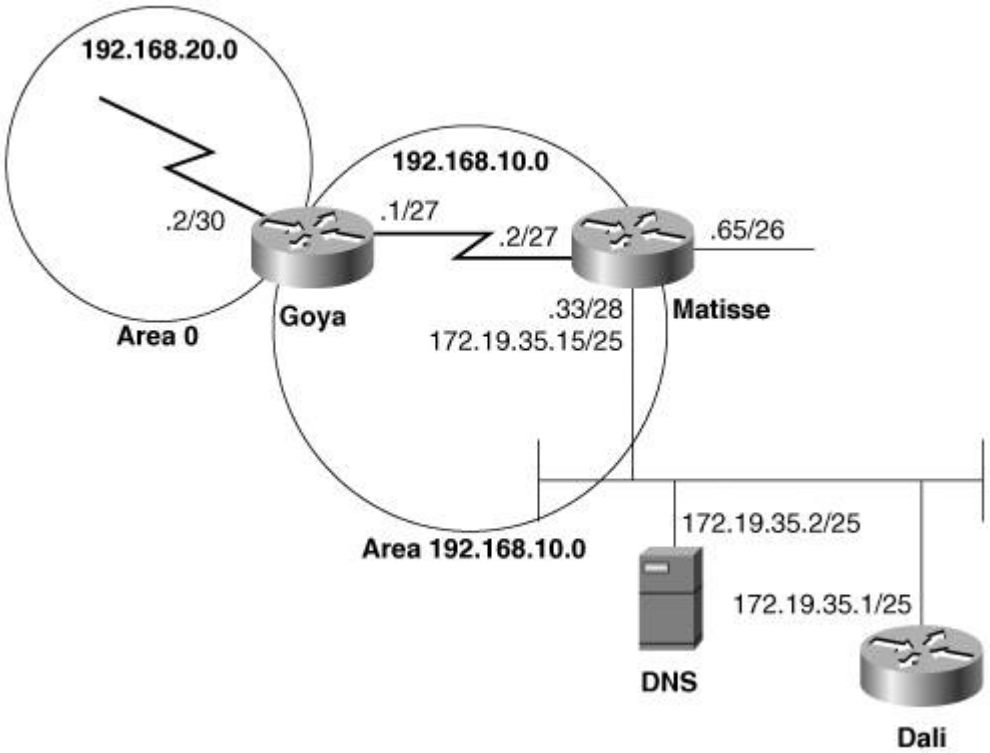
## Case Study: OSPF and Secondary Addresses

Two rules are related to the use of secondary addresses in an OSPF environment:

- OSPF will advertise a secondary network or subnet only if it is also running on the primary network or subnet.
- OSPF sees secondary networks as stub networks (networks on which there are no OSPF neighbors) and therefore will not send Hellos on them. Consequently, no adjacencies can be established on secondary networks.

Figure 8-44 shows the DNS server and an additional router attached to the FA0/0 interface of Matisse. The server and the new router have addresses in subnet 172.19.35.0/25, so Matisse's FA0/0 has been given a secondary address of 172.19.35.15/25 (see Example 8-31).

Figure 8-44. Router Dali and the DNS server are not part of the OSPF domain and are attached to Matisse via a secondary network address.




Example 8-31. Matisse is configured with a secondary address.

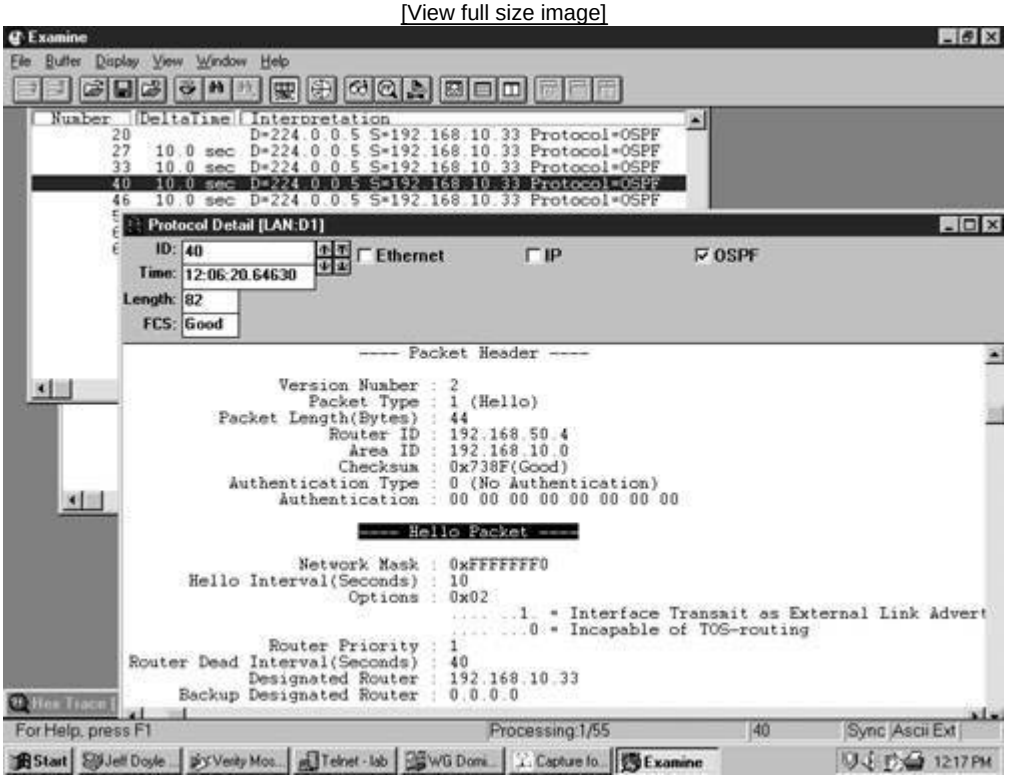
```
interface FastEthernet0/0
 ip address 172.19.35.15 255.255.255.128 secondary
 ip address 192.168.10.33 255.255.255.240
!
router ospf 40
 network 192.168.10.2 0.0.0.0 area 192.168.10.0
 network 192.168.10.33 0.0.0.0 area 192.168.10.0
 network 172.19.35.15 0.0.0.0 area 192.168.10.0
```

With this configuration, Matisse will advertise subnet 172.19.35.0/25 to its neighbors. However, if the network area statement for 192.168.10.33 should be deleted, subnet 172.19.35.0/25 will nc longer be advertised. Because Matisse is attached to subnet 172.19.35.0/25 via a secondary address, it cannot establish an adjacency with any routers on that subnet (Figure 8-45). However, the DNS server uses Dali as its default gateway. Therefore Matisse and Dali must be able to route

packets to each other.

Figure 8-45. This analyzer capture is from the network to which Matisse, Dali, and the DNS server are attached. The smaller window shows that Hellos are only being sourced from Matisse's primary address of 192.168.10.33. The larger window shows a decode of one of the Hellos.

[View full size image]




An assessment of the internet as described so far shows that

- Subnet 172.19.35.0/25 is being advertised into the OSPF domain; a packet with a destination address of 172.19.35.2 will be routed to Matisse's FA0/0 interface and from there directly to the DNS server (Example 8-32).
- Because the DNS server must send replies to network addresses different than its own, it wil send the replies to Dali for routing.
- Dali is not exchanging routing information with Matisse, so it does not know how to reach the networks within the OSPF autonomous system.

Example 8-32. The MAC identifier of the DNS server is recorded in Matisse's ARP cache, indicating that the server can be reached directly. If packets destined for the server had to be routed through Dali, the MAC identifier for both the server and for Dali would be 0000.0c0a.2aa9 in this cache.

```
Matisse#show arp
Protocol  Address       Age (min)  Hardware Addr   Type  Interface
```

```
Internet  192.168.10.65       -   0005.5e6b.50a1  ARPA  FastEthernet0/1
Internet  192.168.10.33       -   0005.5e6b.50a0  ARPA  FastEthernet0/0
Internet  172.19.35.15        -   0005.5e6b.50a0  ARPA  FastEthernet0/0
Internet  172.19.35.1         1   0010.7b3c.6bd3  ARPA  FastEthernet0/0
Internet  172.19.35.2        22   0002.6779.0f4c  ARPA  FastEthernet0/0
```

So the one step needed to "close the circuit" is to tell Dali how to reach the OSPF networks. This is easily done with a static route:

```
Dali(config)#ip route 192.168.0.0 255.255.0.0 172.19.35.15
```

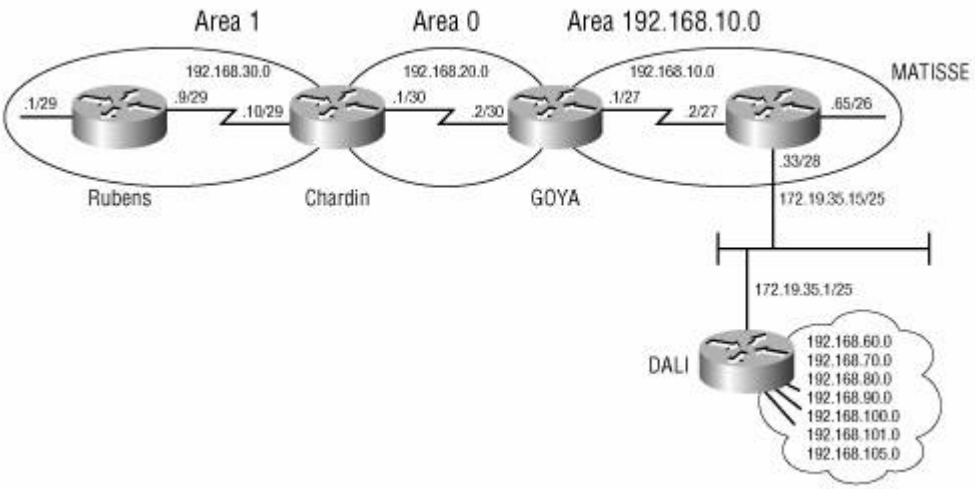
Note that static routes are classless, so the one supernet entry can be used to match all addresses within the OSPF autonomous system.

In this example, Matisse is not an ASBR. Although it sends packets to destinations outside of the autonomous system, it is not accepting any information about exterior destinations and therefore is not originating any type 5 LSAs.

Figure 8-46 shows a new set of destinations reachable via Dali. Matisse must now become an ASBR and advertise the routes into the OSPF domain. However, it must first learn the routes. This task can be done by configuring static routes or by running a routing protocol that will communicate over the secondary network. In either case, the routes must then be redistributed into OSPF.

Figure 8-46. The OSPF autonomous system must learn about the destinations reachable via Dali, but Matisse's secondary address to Dali prevents the two routers from sharing information via OSPF.

[View full size image]




RIP, which has no difficulties with secondary addresses, is chosen to communicate with Dali. Matisse's configuration is as shown in Example 8-33.

Example 8-33. RIP is added to Matisse's configuration.

```
interface FastEthernet0/0
 ip address 172.19.35.15 255.255.255.128 secondary
```

```
 ip address 192.168.10.33 255.255.255.240
!
router ospf 40
 redistribute rip metric 10 subnets
 network 192.168.10.2 0.0.0.0 area 192.168.10.0
 network 192.168.10.33 0.0.0.0 area 192.168.10.0
!
router rip
 network 172.19.0.0
```

This configuration enables RIP on the secondary network of FA0/0, allowing Matisse to learn routes from Dali (Example 8-34). The routes are redistributed into OSPF (which is no longer running on the secondary address) and assigned an OSPF cost of 10, with the command `redistribute rip metric 10 subnets`. See Chapter 11, "Route Redistribution," for more details on redistribution. Example 8-35 shows that the routes are advertised into the OSPF domain with default external type 2 (E2) metrics; notice that at Rubens the cost of these routes is still 10. Matisse advertises these external destinations with type 5 LSAs, making it an ASBR (Example 8-36).

## Example 8-34. Dali has passed its routing information to Matisse via RIP.

```
Matisse#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

R    192.168.90.0/24 [120/1] via 172.19.35.1, 00:00:09, FastEthernet0/0
R    192.168.105.0/24 [120/1] via 172.19.35.1, 00:00:09, FastEthernet0/0
     192.168.30.0/29 is subnetted, 2 subnets
O IA    192.168.30.0 [110/193] via 192.168.10.1, 02:12:53, Serial0/0.1
O IA    192.168.30.8 [110/192] via 192.168.10.1, 02:12:53, Serial0/0.1
R    192.168.60.0/24 [120/1] via 172.19.35.1, 00:00:09, FastEthernet0/0
     192.168.10.0/24 is variably subnetted, 3 subnets, 3 masks
C       192.168.10.64/26 is directly connected, FastEthernet0/1
C       192.168.10.32/28 is directly connected, FastEthernet0/0
C       192.168.10.0/27 is directly connected, Serial0/0.1
     172.19.0.0/25 is subnetted, 1 subnets
C       172.19.35.0 is directly connected, FastEthernet0/0
R    192.168.80.0/24 [120/1] via 172.19.35.1, 00:00:10, FastEthernet0/0

     192.168.20.0/30 is subnetted, 1 subnets
O IA    192.168.20.0 [110/128] via 192.168.10.1, 02:13:06, Serial0/0.1
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.4 is directly connected, Loopback0
R    192.168.70.0/24 [120/1] via 172.19.35.1, 00:00:13, FastEthernet0/0
R    192.168.100.0/24 [120/1] via 172.19.35.1, 00:00:13, FastEthernet0/0
R    192.168.101.0/24 [120/1] via 172.19.35.1, 00:00:13, FastEthernet0/0
```

## Example 8-35. The RIP-learned routes are redistributed into the OSPF autonomous system as path type E2.

```
Rubens#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

O E2 192.168.90.0/24 [110/10] via 192.168.30.10, 02:25:59, Serial0/0.1
O E2 192.168.105.0/24 [110/10] via 192.168.30.10, 02:25:59, Serial0/0.1
     192.168.30.0/29 is subnetted, 2 subnets
C       192.168.30.0 is directly connected, FastEthernet0/0
C       192.168.30.8 is directly connected, Serial0/0.1
O E2 192.168.60.0/24 [110/10] via 192.168.30.10, 02:25:59, Serial0/0.1
     192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
O IA    192.168.10.32/28 [110/193] via 192.168.30.10, 02:26:04, Serial0/0.1
O IA    192.168.10.0/27 [110/192] via 192.168.30.10, 02:26:11, Serial0/0.1
     172.19.0.0/25 is subnetted, 1 subnets
O E2    172.19.35.0 [110/10] via 192.168.30.10, 00:00:27, Serial0/0.1

O E2 192.168.80.0/24 [110/10] via 192.168.30.10, 02:26:00, Serial0/0.1
     192.168.20.0/30 is subnetted, 1 subnets
O IA    192.168.20.0 [110/128] via 192.168.30.10, 02:26:17, Serial0/0.1
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.1 is directly connected, Loopback0O E2 192.168.70.0/24 [110/10]
via 192.168.30.10, 02:26:03, Serial0/0.1
O E2 192.168.100.0/24 [110/10] via 192.168.30.10, 02:26:03, Serial0/0.1
O E2 192.168.101.0/24 [110/10] via 192.168.30.10, 02:26:03, Serial0/0.
```

Example 8-36. Matisse (RID = 192.168.50.4) is now an ASBR because it is originating autonomous system external LSAs to advertise the external routes.

```
Rubens#show ip ospf border-routers

OSPF Process 10 internal Routing Table

Codes: i - Intra-area route, I - Inter-area route

i 192.168.50.2 [64] via 192.168.30.10, Serial0/0.1, ABR, Area 1, SPF 7
I 192.168.50.4 [192] via 192.168.30.10, Serial0/0.1, ASBR, Area 1, SPF 7
```

## Case Study: Stub Areas

Because no type 5 LSAs are being originated within area 1 of Figure 8-46, it can be configured as a stub area. Note that when an attached area is configured as a stub area, the Hellos originated by the router into that area will have E = 0 in the Options field. Any router receiving these Hellos, which is not similarly configured, will drop the packets, and an adjacency will not be established. I there is an existing adjacency, it will be broken. Consequently, if an operational area is going to be reconfigured as a stub area, downtime should be scheduled; routing will be disrupted until all routers are reconfigured.

A stub area is configured by adding the area stub command to the OSPF process as displayed in

Example 8-37 and Example 8-38.

## Example 8-37. Rubens is configured to make area 1 a stub area.

```
router ospf 10
 network 0.0.0.0 255.255.255.255 area 1
 area 1 stub
```

## Example 8-38. Chardin is configured to make area 1 a stub area.

```
router ospf 20
 network 192.168.30.0 0.0.0.255 area 1
 network 192.168.20.0 0.0.0.255 area 0
 area 1 stub
```

Comparing the link-state database of Rubens before (Example 8-39) and after (Example 8-40) the configured stub area shows that all autonomous system external LSAs and ASBR summary LSAs have been eliminated from the database. In this case, the size of the database has been reduced by more than 50 percent.

## Example 8-39. Rubens's database has a total of 14 LSAs before area 1 is configured as a stub area.

```
Rubens#show ip ospf database database-summary

            OSPF Router with ID (192.168.50.1) (Process ID 10)

Area 1 database summary
  LSA Type      Count    Delete   Maxage
  Router        2        0        0
  Network       0        0        0
  Summary Net   3        0        0
  Summary ASBR  1        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   0        0        0
  Subtotal      6        0        0

Process 10 database summary
  LSA Type      Count    Delete   Maxage
  Router        2        0        0
  Network       0        0        0
  Summary Net   3        0        0
  Summary ASBR  1        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   0        0        0
  Type-5 Ext    8        0        0

  Opaque AS     0        0        0
  Total         14       0        0
```

## Example 8-40. The stub area configuration eliminates the eight type 5

LSAs and the single type 4 LSA from Rubens's database. One type 3 LSA, which advertises the default route, has been added.

```
Rubens#show ip ospf database database-summary

            OSPF Router with ID (192.168.50.1) (Process ID 10)

Area 1 database summary
  LSA Type      Count    Delete   Maxage
  Router        2        0        0
  Network       0        0        0
  Summary Net   4        0        0
  Summary ASBR  0        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   0        0        0
  Subtotal      6        0        0

Process 10 database summary
  LSA Type      Count    Delete   Maxage
  Router        2        0        0
  Network       0        0        0
  Summary Net   4        0        0
  Summary ASBR  0        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   0        0        0

  Type-5 Ext    0        0        0
  Opaque AS     0        0        0
  Total         6        0        0
```

When a stub area is attached to an ABR, the router will automatically advertise a default route (destination 0.0.0.0) into the area via a Network Summary LSA. The database summary in Example 8-40 indicates this additional type 3 LSA. The last entry in Rubens's route table (Example 8-41) shows the default route advertised by Chardin.

Example 8-41. Rubens's route table shows that all external routes have been eliminated (compare this to Example 8-35) and that a default route has been added.

```
Rubens#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.30.10 to network 0.0.0.0

     192.168.30.0/29 is subnetted, 2 subnets
C       192.168.30.0 is directly connected, FastEthernet0/0
C       192.168.30.8 is directly connected, Serial0/0.1
     192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
O IA    192.168.10.32/28 [110/193] via 192.168.30.10, 00:05:24, Serial0/0.1
O IA    192.168.10.0/27 [110/192] via 192.168.30.10, 00:05:24, Serial0/0.1
```

```
     192.168.20.0/30 is subnetted, 1 subnets
O IA    192.168.20.0 [110/128] via 192.168.30.10, 00:05:24, Serial0/0.1
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.1 is directly connected, Loopback0
O*IA 0.0.0.0/0 [110/65] via 192.168.30.10, 00:05:25, Serial0/0.1
```

The ABR will advertise a default route with a cost of 1. The cost of the serial link between Rubens and Chardin is 64; Example 8-41 shows the total cost of the default route to be 64 + 1 = 65. This default cost can be changed with the `area default-cost` command. For example, Chardin can be configured to advertise the default route with a cost of 20 as shown in Example 8-42.

### Example 8-42. Chardin's configuration sets the cost of the advertised default route.

```
router ospf 20
 network 192.168.30.0 0.0.0.255 area 1
 network 192.168.20.0 0.0.0.255 area 0
 area 1 stub
 area 1 default-cost 20
```

The resulting cost increase, 64 + 20 = 84, can be seen in Example 8-43. Changing the cost of the default route has no real benefit here, but might be useful in stub areas with more than one ABR. Normally, each Internal Router would merely choose the default route with the lowest cost. By manipulating the advertised cost, the network administrator could cause all Internal Routers to use the same ABR. The second ABR, advertising a higher cost, would be used only if the first were to fail.

### Example 8-43. Rubens's route table reflects the results of changing the cost of the default route.

```
Rubens#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.30.10 to network 0.0.0.0

     192.168.30.0/29 is subnetted, 2 subnets
C       192.168.30.0 is directly connected, FastEthernet0/0
C       192.168.30.8 is directly connected, Serial0/0.1
     192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
O IA    192.168.10.32/28 [110/193] via 192.168.30.10, 00:18:49, Serial0/0.1
O IA    192.168.10.0/27 [110/192] via 192.168.30.10, 00:18:49, Serial0/0.1
     192.168.20.0/30 is subnetted, 1 subnets
O IA    192.168.20.0 [110/128] via 192.168.30.10, 00:18:49, Serial0/0.1
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.1 is directly connected, Loopback0
O*IA 0.0.0.0/0 [110/84] via 192.168.30.10, 00:10:36, Serial0/0.1
```

## Case Study: Totally Stubby Areas

Totally stubby areas are configured by placing the keyword no-summary at the end of the area stub command. This step is necessary only at the ABR; the Internal Routers use the standard stub area configuration. To make area 1 in Figure 8-46 a totally stubby area, Chardin's configuration would be as shown in Example 8-44.

## Example 8-44. Chardin is configured to make area 1 a totally stubby area.

```
router ospf 20
 network 192.168.30.0 0.0.0.255 area 1
 network 192.168.20.0 0.0.0.255 area 0
 area 1 stub no-summary
```

Example 8-45 shows that the LSAs in Rubens's database have been reduced to three; Example 8-46 shows the route table.

## Example 8-45. Changing area 1 to a totally stubby area eliminates all but one of the type 3 LSAs (the default route).

```
Rubens#show ip ospf database database-summary

            OSPF Router with ID (192.168.50.1) (Process ID 10)

Area 1 database summary
  LSA Type      Count    Delete   Maxage
  Router        2        0        0
  Network       0        0        0
  Summary Net   1        0        0
  Summary ASBR  0        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   0        0        0
  Subtotal      3        0        0

Process 10 database summary
  LSA Type      Count    Delete   Maxage
  Router        2        0        0
  Network       0        0        0
  Summary Net   1        0        0
  Summary ASBR  0        0        0
  Type-7 Ext    0        0        0
  Opaque Link   0        0        0
  Opaque Area   0        0        0

  Type-5 Ext    0        0        0
  Opaque AS     0        0        0
  Total         3        0        0
```

## Example 8-46. A route table in a totally stubby area will contain only intra-area routes and the default route.

```
Rubens#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 192.168.30.10 to network 0.0.0.0

     192.168.30.0/29 is subnetted, 2 subnets
C       192.168.30.0 is directly connected, FastEthernet0/0
C       192.168.30.8 is directly connected, Serial0/0.1
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.1 is directly connected, Loopback0
O*IA 0.0.0.0/0 [110/84] via 192.168.30.10, 00:15:52, Serial0/0.1
```

## Case Study: Not-So-Stubby Areas

The earlier case study, "OSPF and Secondary Addresses," left off with Matisse accepting routes from Dali via RIP and redistributing them into the OSPF domain (see Figure 8-46). This step makes Matisse an ASBR, and by extension makes area 192.168.10.0 ineligible to become a stub or totally stubby area. However, there is no need for AS External LSAs to enter the area from the backbone area; therefore area 192.168.10.0 can be configured as an NSSA. Example 8-47 shows the configuration at Matisse.

### Example 8-47. Matisse is configured to make area 192.168.10.0 a not-so-stubby area.

```
router ospf 40
 redistribute rip metric 10
 network 192.168.10.2 0.0.0.0 area 192.168.10.0
 network 192.168.10.33 0.0.0.0 area 192.168.10.0
 area 192.168.10.0 nssa
!
router rip
 network 172.19.0.0
```

The same area nssa statement shown here is configured at Goya. Because Goya is an ABR, it will translate type 7 LSAs received on the NSSA-attached interface into type 5 LSAs. These translated LSAs will be flooded into the backbone and hence to the other areas. Comparing the route tables of Goya and Chardin shows that Goya has tagged the external routes as NSSA[28] (Example 8-48). Chardin has tagged the routes as E2 (Example 8-49), indicating that they have been learned from type 5 LSAs.

[28] N2 indicates the same metric calculation as E2that is, only the external cost is used. A subsequent example will demonstrate E1 and N1 metric types.

### Example 8-48. The external routes learned from Matisse are tagged as NSSA routes at Goya.

```
Goya#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
```

```
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

O N2 192.168.90.0/24 [110/10] via 192.168.10.2, 00:00:41, Serial0/0.1
O N2 192.168.105.0/24 [110/10] via 192.168.10.2, 00:00:41, Serial0/0.1
     192.168.30.0/29 is subnetted, 2 subnets
O IA    192.168.30.0 [110/129] via 192.168.20.1, 00:00:41, Serial0/0.2
O IA    192.168.30.8 [110/128] via 192.168.20.1, 00:00:41, Serial0/0.2
O N2 192.168.60.0/24 [110/10] via 192.168.10.2, 00:00:41, Serial0/0.1
     192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
O       192.168.10.32/28 [110/65] via 192.168.10.2, 00:00:43, Serial0/0.1
C       192.168.10.0/27 is directly connected, Serial0/0.1
     172.19.0.0/25 is subnetted, 1 subnets
O N2    172.19.35.0 [110/10] via 192.168.10.2, 00:00:43, Serial0/0.1
O N2 192.168.80.0/24 [110/10] via 192.168.10.2, 00:00:43, Serial0/0.1
     192.168.20.0/30 is subnetted, 1 subnets
C       192.168.20.0 is directly connected, Serial0/0.2
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.3 is directly connected, Loopback0
O N2 192.168.70.0/24 [110/10] via 192.168.10.2, 00:00:55, Serial0/0.1
O N2 192.168.100.0/24 [110/10] via 192.168.10.2, 00:00:56, Serial0/0.1
O N2 192.168.101.0/24 [110/10] via 192.168.10.2, 00:00:56, Serial0/0.1
```

Example 8-49. Chardin has tagged the same routes as E2, indicating they have been learned from autonomous system external LSAs.

```
Chardin#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

O E2 192.168.90.0/24 [110/10] via 192.168.20.2, 00:02:49, Serial0/0.2
O E2 192.168.105.0/24 [110/10] via 192.168.20.2, 00:02:49, Serial0/0.2
     192.168.30.0/29 is subnetted, 2 subnets
O       192.168.30.0 [110/65] via 192.168.30.9, 00:08:41, Serial0/0.1
C       192.168.30.8 is directly connected, Serial0/0.1
O E2 192.168.60.0/24 [110/10] via 192.168.20.2, 00:02:49, Serial0/0.2
     192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
O IA    192.168.10.32/28 [110/129] via 192.168.20.2, 00:02:55, Serial0/0.2
O IA    192.168.10.0/27 [110/128] via 192.168.20.2, 00:03:16, Serial0/0.2
     172.19.0.0/25 is subnetted, 1 subnets
O E2    172.19.35.0 [110/10] via 192.168.20.2, 00:02:50, Serial0/0.2
O E2 192.168.80.0/24 [110/10] via 192.168.20.2, 00:02:50, Serial0/0.2
     192.168.20.0/30 is subnetted, 1 subnets
C       192.168.20.0 is directly connected, Serial0/0.2
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.2 is directly connected, Loopback0
O E2 192.168.70.0/24 [110/10] via 192.168.20.2, 00:02:52, Serial0/0.2
O E2 192.168.100.0/24 [110/10] via 192.168.20.2, 00:02:52, Serial0/0.2
O E2 192.168.101.0/24 [110/10] via 192.168.20.2, 00:02:52, Serial0/0.2
```

This translation can also be observed by examining Goya's database. Example 8-50 shows that the database contains both type 7 and type 5 LSAs to the same external routes. The originating router for the type 7 LSAs is Matisse, whereas the originating router for the type 5 LSAs is Goya.

## Example 8-50. Goya's link-state database indicates that the type 7 LSAs from Matisse (192.168.50.4) have been translated into type 5 LSAs by Goya (192.168.50.3).

```
                Type-7 AS External Link States (Area 192.168.10.0)

Link ID         ADV Router      Age         Seq#       Checksum Tag
172.19.35.0     192.168.50.4    371         0x80000001 0x00C444 0
192.168.60.0    192.168.50.4    371         0x80000001 0x00A521 0
192.168.70.0    192.168.50.4    371         0x80000001 0x003785 0
192.168.80.0    192.168.50.4    371         0x80000001 0x00C8E9 0
192.168.90.0    192.168.50.4    371         0x80000001 0x005A4E 0
192.168.100.0   192.168.50.4    371         0x80000001 0x00EBB2 0
192.168.101.0   192.168.50.4    371         0x80000001 0x00E0BC 0
192.168.105.0   192.168.50.4    371         0x80000001 0x00B4E4 0

                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
172.19.35.0     192.168.50.3    305         0x80000001 0x005FB4 0
192.168.60.0    192.168.50.3    306         0x80000001 0x004091 0
192.168.70.0    192.168.50.3    306         0x80000001 0x00D1F5 0
192.168.80.0    192.168.50.3    306         0x80000001 0x00635A 0
192.168.90.0    192.168.50.3    390         0x80000001 0x00F4BE 0
192.168.100.0   192.168.50.3    390         0x80000001 0x008623 0
192.168.101.0   192.168.50.3    390         0x80000001 0x007B2D 0
192.168.105.0   192.168.50.3    390         0x80000001 0x004F55 0
```

Several configuration options are available for the ABR. First, the no-summary option can be used with the area nssa command to block the flooding of type 3 and type 4 LSAs into the NSSA. To turn area 192.168.10.0 into a somewhat schizophrenically named "totally stubby not-so-stubby" area, Goya's configuration would be as shown in Example 8-51.

## Example 8-51. Goya is configured to make area 192.168.10.0 a totally stubby not-so-stubby area.

```
router ospf 30
 network 192.168.20.0 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.31 area 192.168.10.0
 area 192.168.10.0 nssa no-summary
```

Matisse's route table (Example 8-52) shows the elimination of all inter-area routes and the addition of a default route advertised by Goya.

## Example 8-52. All inter-area routes have been replaced with a default route to the ABR.

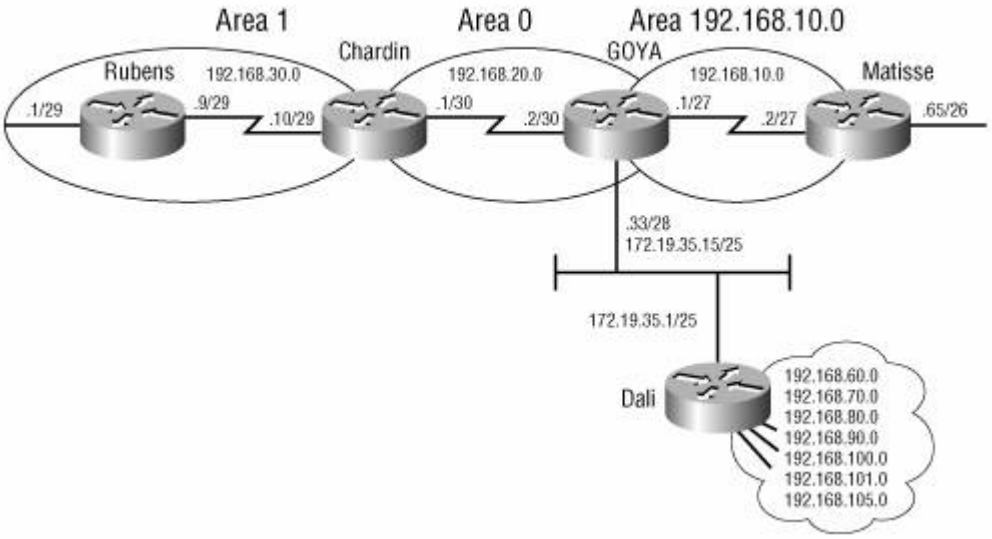
```
Matisse#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
        ia - IS-IS inter area, * - candidate default, U - per-user static route
        o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.10.1 to network 0.0.0.0

R    192.168.90.0/24 [120/1] via 172.19.35.1, 00:00:20, FastEthernet0/0
R    192.168.105.0/24 [120/1] via 172.19.35.1, 00:00:20, FastEthernet0/0
R    192.168.60.0/24 [120/1] via 172.19.35.1, 00:00:20, FastEthernet0/0
     192.168.10.0/24 is variably subnetted, 3 subnets, 3 masks
C       192.168.10.64/26 is directly connected, FastEthernet0/1
C       192.168.10.32/28 is directly connected, FastEthernet0/0
C       192.168.10.0/27 is directly connected, Serial0/0.1
     172.19.0.0/25 is subnetted, 1 subnets
C       172.19.35.0 is directly connected, FastEthernet0/0
R    192.168.80.0/24 [120/1] via 172.19.35.1, 00:00:21, FastEthernet0/0
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.4 is directly connected, Loopback0
R    192.168.70.0/24 [120/1] via 172.19.35.1, 00:00:21, FastEthernet0/0

R    192.168.100.0/24 [120/1] via 172.19.35.1, 00:00:22, FastEthernet0/0
R    192.168.101.0/24 [120/1] via 172.19.35.1, 00:00:22, FastEthernet0/0
O*IA 0.0.0.0/0 [110/65] via 192.168.10.1, 00:11:40, Serial0/0.1
```

In Figure 8-47, the link to Dali has been moved from Matisse to Goya; the IP address has also moved. Goya is now an ASBR redistributing RIP-learned routes into OSPF.

Figure 8-47. The link to Dali has moved to Goya, which now speaks RIP to Dali and redistributes the learned routes into OSPF.

[View full size image]




When an ABR is also an ASBR and is connected to a not-so-stubby area, the default behavior is to advertise the redistributed routes into the NSSA as shown in Example 8-53.

## Example 8-53. An ABR that is also an ASBR will advertise the external routes into an NSSA with type 7 LSAs. In this example, Goya is advertising the external routes with an N1 metric type.

```
Matisse#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

O N1 192.168.90.0/24 [110/74] via 192.168.10.1, 00:00:07, Serial0/0.1
O N1 192.168.105.0/24 [110/74] via 192.168.10.1, 00:00:07, Serial0/0.1
     192.168.30.0/29 is subnetted, 2 subnets
O IA    192.168.30.0 [110/193] via 192.168.10.1, 00:03:11, Serial0/0.1
O IA    192.168.30.8 [110/192] via 192.168.10.1, 00:03:11, Serial0/0.1
O N1 192.168.60.0/24 [110/74] via 192.168.10.1, 00:00:07, Serial0/0.1
     192.168.10.0/24 is variably subnetted, 3 subnets, 3 masks
C       192.168.10.64/26 is directly connected, FastEthernet0/1
C       192.168.10.32/28 is directly connected, FastEthernet0/0
C       192.168.10.0/27 is directly connected, Serial0/0.1
     172.19.0.0/25 is subnetted, 1 subnets
O N1    172.19.35.0 [110/74] via 192.168.10.1, 00:03:07, Serial0/0.1
O N1 192.168.80.0/24 [110/74] via 192.168.10.1, 00:00:08, Serial0/0.1
     192.168.20.0/30 is subnetted, 1 subnets

O IA    192.168.20.0 [110/128] via 192.168.10.1, 00:03:14, Serial0/0.1
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.4 is directly connected, Loopback0
O N1 192.168.70.0/24 [110/74] via 192.168.10.1, 00:00:10, Serial0/0.1
O N1 192.168.100.0/24 [110/74] via 192.168.10.1, 00:00:10, Serial0/0.1
O N1 192.168.101.0/24 [110/74] via 192.168.10.1, 00:00:10, Serial0/0.1
```

The default redistribution may be turned off on an ABR/ASBR by adding the statement no-redistribution to the area nssa command. In the sample internet, no types 3, 4, 5, or 7 LSAs should be sent into area 192.168.10.0 from the ABR. The desired redistribution is accomplished by the Example 8-54 configuration at Goya:[29]

[29] Note the metric-type 1 statement in the redistribution command. This statement causes the external destinations to be advertised with an E1 metric; within the NSSA, the metric type becomes N1, as shown in Example 8.53.

## Example 8-54. Goya is configured to not redistribute RIP routes into the NSSA 192.168.10.0.

```
interface Ethernet0/0
 ip address 172.19.35.15 255.255.255.128
!
router ospf 30
 redistribute rip metric 10 metric-type 1 subnets
 network 192.168.20.0 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.31 area 192.168.10.0
 area 192.168.10.0 nssa no-redistribution no-summary
!
router rip
```

```
 network 172.19.0.0
```

Here① the area nssa command blocks type 5 LSAs from entering the area through Goya;② no-redistribution blocks type 7 LSAs, and③ no-summary blocks types 3 and 4. As before, the no-summary command also causes Goya to send a single type 3 LSA into the area to advertise a default route. Example 8-55 shows Matisse's route table after type 7 redistribution is disabled at Goya. Note that the external networks are still reachable, even though they are not in the table, because of the default route.

## Example 8-55. After no-redistribution is added to Goya's area nssa command, the route table of Example 8-53 no longer contains routes learned from type 7 LSAs.

```
Matisse#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.10.1 to network 0.0.0.0

     192.168.10.0/24 is variably subnetted, 3 subnets, 3 masks
C       192.168.10.64/26 is directly connected, FastEthernet0/1
C       192.168.10.32/28 is directly connected, FastEthernet0/0
C       192.168.10.0/27 is directly connected, Serial0/0.1
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.4 is directly connected, Loopback0
O*IA 0.0.0.0/0 [110/65] via 192.168.10.1, 00:00:51, Serial0/0.1
```

In the final example, Goya is to allow types 3 and 4 LSAs into the NSSA, but not types 5 and 7. The problem is that when the no-summary statement is removed, the ABR will no longer originate a type 3 LSA advertising a default route. Without a default route, the exterior destinations will not be reachable from within the NSSA. The statement default-information-originate, added to the area nssa command, will cause the ABR to advertise a default route into the NSSAthis time, with a type 7 LSA. Using this statement, Goya's OSPF configuration is displayed in Example 8-56.

## Example 8-56. Default route is originated at Goya with the default-information-originate command.

```
router ospf 30
 redistribute rip metric 10 metric-type 1 subnets
 network 192.168.20.0 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.31 area 192.168.10.0
 area 192.168.10.0 nssa no-redistribution default-information-originate
```

Example 8-57 shows Matisse's route table after the reconfiguration. The table contains inter-area routes and a default route with an N2 tag indicating that the route was learned from a type 7 LSA.

## Example 8-57. Adding the default-information-originate statement to

the area nssa command causes the ABR to advertise a default route intc an NSSA.

```
Matisse#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.10.1 to network 0.0.0.0

     192.168.30.0/29 is subnetted, 2 subnets
O IA    192.168.30.0 [110/193] via 192.168.10.1, 00:00:26, Serial0/0.1
O IA    192.168.30.8 [110/192] via 192.168.10.1, 00:00:26, Serial0/0.1
     192.168.10.0/24 is variably subnetted, 3 subnets, 3 masks
C       192.168.10.64/26 is directly connected, FastEthernet0/1
C       192.168.10.32/28 is directly connected, FastEthernet0/0
C       192.168.10.0/27 is directly connected, Serial0/0.1
     192.168.20.0/30 is subnetted, 1 subnets
O IA    192.168.20.0 [110/128] via 192.168.10.1, 00:00:27, Serial0/0.1
     192.168.50.0/32 is subnetted, 1 subnets
C       192.168.50.4 is directly connected, Loopback0
O*N2 0.0.0.0/0 [110/1] via 192.168.10.1, 00:00:22, Serial0/0.1
```

## Case Study: Address Summarization

Although stub areas conserve resources within non-backbone areas by preventing certain LSAs from entering, these areas do nothing to conserve resources on the backbone. All addresses within an area are still advertised out to the backbone. This situation is where address summarization can help. Like stub areas, address summarization conserves resources by reducing the number of LSAs flooded. It also conserves resources by hiding instabilities. For example, a "flapping" subnet will cause LSAs to be flooded throughout the network at each state transition. However, if that subnet address is subsumed by a summary address, the individual subnet and its instabilities are no longer advertised.

The Cisco OSPF can perform two types of address summarization: inter-area summarization and external route summarization. *Inter-area summarization* is, as the name implies, the summarization of addresses between areas; this type of summarization is always configured on ABRs. *External route summarization* allows a set of external addresses to be redistributed into an OSPF domain as a summary address and is configured on ASBRs. Inter-area summarization is covered in this section, and external route summarization is covered in Chapter 11.

In Figure 8-48, area 15 contains eight subnets: 10.0.0.0/16 through 10.7.0.0/16. Figure 8-49 shows that these addresses can be represented with the single summary address 10.0.0.0/13.

Figure 8-48. The addresses in areas 15 and 25 can be summarized into the backbone area.

[View full size image]

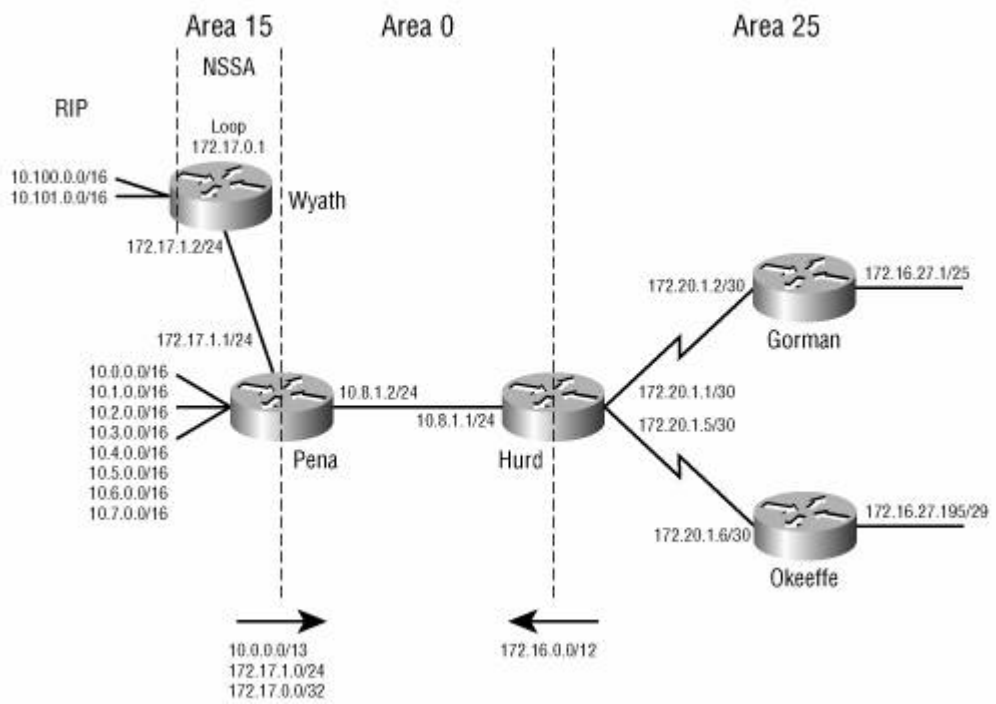



Figure 8-49. The summary address 10.0.0.0/13 represents the range of addresses from 10.0.0.0/16 to 10.7.0.0/16.

```
11111111111111110000000000000000  =  16-bit mask
00001010000000000000000000000000  =  10.0.0.0/16
00001010000000010000000000000000  =  10.1.0.0/16
00001010000000100000000000000000  =  10.2.0.0/16
00001010000000110000000000000000  =  10.3.0.0/16
00001010000001000000000000000000  =  10.4.0.0/16
00001010000001010000000000000000  =  10.5.0.0/16
00001010000001100000000000000000  =  10.6.0.0/16
00001010000001110000000000000000 = 10.7.0.0/16
00001010000000000000000000000000  =  10.0.0.0/13
```

An ABR can be configured to advertise a summary address either into the backbone area or into a non-backbone area. Best practice dictates that a non-backbone area's addresses should be summarized into the backbone by its own ABR, as opposed to having all other ABRs summarize the area into their areas. Then, from the backbone area, the summary will be advertised across the backbone and into the other areas. This both simplifies the router configurations and reduces the size of the LS database in the backbone.

The area range command specifies the area to which the summary address belongs, the summary address, and the address mask. Recall from Chapter 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)," that when a summary route is configured for EIGRP, a route to the null interface is automatically entered into the route table to prevent black holes and route loops.[30] OSPF also enters this route automatically. In the IOS mainline versions earlier then 12.1, however, the route to the null interface will not be entered automatically. In addition, in IOS releases that have the capability of creating this route, the router can be configured to not install it in the route table using the command no discard-route. Therefore, whenever you are configuring summary routes within an OSPF domain, be sure to verify that the route for the summary address, pointing to the null interface, is created. If it is not automatically created, be sure to add it or to issue the discard-route command.

[30] The reasons for this route are discussed in more detail, with examples, in Chapters 11, "Route Redistribution," and 12, "Default Routes and On-Demand Routing."

Example 8-58 shows Pena's OSPF configuration.

## Example 8-58. Pena's OSPF configuration.

```
router ospf 1
 network 10.0.0.0 0.7.255.255 area 15
 network 10.8.0.0 0.7.255.255 area 0
 area 15 range 10.0.0.0 255.248.0.0
!
ip route 10.0.0.0 255.248.0.0 Null0
```

Figure 8-49 shows that the range of addresses represented by 10.0.0.0/13 is contiguousthat is, the three bits that are summarized constitute every combination from 000 to 111. The addresses in area 25 are different. These do not form a contiguous range. They may, however, still be summarized with the Example 8-59 configuration at Hurd.

## Example 8-59. Hurd's OSPF configuration.

```
router ospf 1
 network 10.8.0.0 0.0.255.255 area 0
 network 172.20.0.0 0.0.255.255 area 25
 area 25 range 172.16.0.0 255.240.0.0
!
ip route 172.16.0.0 255.240.0.0 Null0
```

This summary will work, even if some of the addresses in the range appear elsewhere in the network. In Figure 8-48, network 172.17.0.0/16 is in area 15, although it belongs to the set of addresses being summarized from area 25. Pena advertises this address into the backbone area, where Hurd learns it and advertises it into area 25. The accompanying mask is more specific (that is, longer) than the mask of the summary address 172.16.0.0/12; because OSPF is classless, it will route destination addresses belonging to 172.17.0.0 to the correct destination.

Although the address configuration of Figure 8-48 will work, it is an undesirable design practice. The point of summarization is to conserve resources; network 172.17.1.0/24 must be advertised across the backbone independently of 172.16.0.0/12. Such a design also creates the potential for route loops if default addresses are used. This problem is discussed in Chapter 12.

Notice also in Figure 8-48 that subnets 172.16.27.0/25 (at Gorman) and 172.16.27.192/29 (at Okeeffe) are discontiguous. Again because OSPF is a classless routing protocol, Gorman and Okeeffe do not act as network border routers. The subnets and their masks are advertised into network 172.20.0.0, and no routing ambiguities will occur.

The default behavior of the `area range` command is to advertise the range specified. It might be desirable to suppress the advertisement of a range of addresses. The `area range` command with the keyword `not-advertise` causes prefixes in the specified range to be suppressed. They are not advertised in LSAs.

Pena is configured to suppress the range 172.17.0.0/16. Pena's configuration becomes as displayed in Example 8-60.

## Example 8-60. Pena is configured to suppress the advertisement of a range of addresses.

```
router ospf 1
 area 15 range 10.0.0.0 255.248.0.0
```

```
 area 15 range 172.17.0.0 255.255.0.0 not-advertise
 network 10.0.0.0 0.7.255.255 area 15
 network 10.8.0.0 0.7.255.255 area 0
 network 172.17.0.0 0.0.255.255 area 15
```

This command has the desired affect of suppressing the 172.17.0.0/16 range, but it has an undesirable consequence. Look at the route table on Hurd before (Example 8-61) and after (Example 8-62) entering the command.

## Example 8-61. Hurd's route table before Pena suppressed an address range shows two external routes.

```
Hurd#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.17.0.0/16 is variably subnetted, 2 subnets, 2 masks
O IA    172.17.1.0/24 [110/11] via 10.8.1.2, 00:03:29, Ethernet0/0
O IA    172.17.0.1/32 [110/12] via 10.8.1.2, 00:03:29, Ethernet0/0
     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O       172.16.27.192/29 [110/65] via 172.20.1.6, 00:03:29, Serial0/0.2
O       172.16.27.0/25 [110/65] via 172.20.1.2, 00:03:29, Serial0/0.1
     172.20.0.0/30 is subnetted, 2 subnets
C       172.20.1.0 is directly connected, Serial0/0.1
C       172.20.1.4 is directly connected, Serial0/0.2
     10.0.0.0/8 is variably subnetted, 4 subnets, 3 masks
C       10.8.1.0/24 is directly connected, Ethernet0/0
O IA    10.0.0.0/13 [110/11] via 10.8.1.2, 00:03:30, Ethernet0/0
O E2    10.100.0.0/16 [110/10] via 10.8.1.2, 00:03:30, Ethernet0/0
O E2    10.101.0.0/16 [110/10] via 10.8.1.2, 00:03:32, Ethernet0/0
S    172.16.0.0/12 is directly connected, Null0
```

## Example 8-62. Hurd's route table after Pena suppresses a range shows the external routes are no longer in the route table.

```
Hurd#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O       172.16.27.192/29 [110/65] via 172.20.1.6, 00:06:11, Serial0/0.2
O       172.16.27.0/25 [110/65] via 172.20.1.2, 00:06:11, Serial0/0.1
```

```
     172.20.0.0/30 is subnetted, 2 subnets
C       172.20.1.0 is directly connected, Serial0/0.1
C       172.20.1.4 is directly connected, Serial0/0.2
     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.8.1.0/24 is directly connected, Ethernet0/0
O IA    10.0.0.0/13 [110/11] via 10.8.1.2, 00:06:12, Ethernet0/0
S    172.16.0.0/12 is directly connected, Null0
```

The external routes learned via RIP on Wyeth, 10.100.0.0 and 10.101.0.0, are no longer in Hurd's route table. Example 8-63 displays the OSPF database for 10.100.0.0 on Hurd. Notice the specified forwarding address.

## Example 8-63. The type 5 external link state shows the forwarding address to be the address of the router that originated the external route.

```
Hurd#show ip ospf data external

            OSPF Router with ID (172.20.1.5) (Process ID 1)

                Type-5 AS External Link States

  Routing Bit Set on this LSA
  LS age: 421
  Options: (No TOS-capability, DC)
  LS Type: AS External Link
  Link State ID: 10.100.0.0 (External Network Number )
  Advertising Router: 172.17.0.1
  LS Seq Number: 80000001
  Checksum: 0xF1CC
  Length: 36
  Network Mask: /16
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 10
        Forward Address: 172.17.0.1
        External Route Tag: 0
```

The forwarding address is the loopback interface of Wyeth. If the forwarding address in an external LSA is specified, and this address is not reachable, the address contained in the LSA is not inserted into the route table. When Pena translates type 7 NSSA LSAs into type 5 LSAs, by default the forwarding address is transferred from the type 7 to the type 5. The ABR can be configured to suppress the forwarding address during the translation, replacing the specified address with the address 0.0.0.0. When another router receives the type 5 external LSA with the forwarding address suppressed, instead of trying to direct traffic for the external address to the forwarding address, the receiving router will attempt to direct the traffic to the type 7 to type 5 translating ABR router.

Example 8-64 has Pena's new configuration.

## Example 8-64. Pena's configuration suppresses the inclusion of a forwarding address in translated type 5 LSAs.

```
router ospf 1
 area 15 range 10.0.0.0 255.248.0.0
 area 15 range 172.17.0.0 255.255.0.0 not-advertise
```

```
 network 10.0.0.0 0.7.255.255 area 15
 network 10.8.0.0 0.7.255.255 area 0
 network 172.17.0.0 0.0.255.255 area 15
 area 15 nssa translate type7 suppress-fa
```

Example 8-65 shows Hurd's external database entry for 10.100.0.0, and Example 8-66 shows Hurd's new route table.

## Example 8-65. The external OSPF database entry shows a suppressed forwarding address.

```
Hurd#show ip ospf data external

            OSPF Router with ID (172.20.1.5) (Process ID 1)

                Type-5 AS External Link States

  Routing Bit Set on this LSA
  LS age: 13
  Options: (No TOS-capability, DC)
  LS Type: AS External Link
  Link State ID: 10.100.0.0 (External Network Number )
  Advertising Router: 172.17.0.1
  LS Seq Number: 80000002
  Checksum: 0xA9D2
  Length: 36
  Network Mask: /16
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 10
        Forward Address: 0.0.0.0
        External Route Tag: 0
```

## Example 8-66. The external routes for 10.100.0.0 and 10.101.0.0 are inserted into the route table after suppressing the forwarding address.

```
Hurd#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O       172.16.27.192/29 [110/65] via 172.20.1.6, 00:09:32, Serial0/0.2
O       172.16.27.0/25 [110/65] via 172.20.1.2, 00:09:32, Serial0/0.1
     172.20.0.0/30 is subnetted, 2 subnets
C       172.20.1.0 is directly connected, Serial0/0.1
C       172.20.1.4 is directly connected, Serial0/0.2
     10.0.0.0/8 is variably subnetted, 4 subnets, 3 masks
C       10.8.1.0/24 is directly connected, Ethernet0/0
O IA    10.0.0.0/13 [110/11] via 10.8.1.2, 00:09:33, Ethernet0/0
```

```
O E2    10.100.0.0/16 [110/10] via 10.8.1.2, 00:00:46, Ethernet0/0
O E2    10.101.0.0/16 [110/10] via 10.8.1.2, 00:00:46, Ethernet0/0
S    172.16.0.0/12 is directly connected, Null0
```

The forwarding address displayed in Hurd's database is now 0.0.0.0 and the two external routes are back in the route table.

## Case Study: Filtering Between Areas

Another LAN is added to Okeeffe. Devices on the LAN are to be accessed by other devices within area 25, but not from the rest of the network. Another way to limit and control the addresses that are exchanged between areas is to configure LSA type 3 filtering on ABRs. The ABRs can filter network addresses being advertised by type 3 LSAs either into or out of an area.

A LAN with address 192.168.1.0/24 is added to Okeeffe in area 25. This address is advertised in LSAs throughout the network even though it is only to be accessed by other devices in area 25. To prevent the address from being advertised outside area 25, type 3 LSA filtering is configured on Hurd as shown in Example 8-67.

### Example 8-67. Hurd uses type 3 LSA filtering.

```
router ospf 1
 area 25 filter-list prefix area25outbound out
!
ip prefix-list area25outbound seq 10 deny 192.168.1.0/24
ip prefix-list area25outbound seq 20 permit 0.0.0.0/0 le 32
```

The OSPF command `area` *PID* `filter-list prefix` specifies the name of a filter list to apply to outbound or inbound LSAs. Outbound lists filter LSAs sent into areas other then the one specified by the command. In our example, the list filters LSAs with addresses originating in area 25 and being sent into non-area 25 areas, such as area 0. Inbound lists filter LSAs as they are sent into area 25.

The first line of the prefix list is clear. The statement denies address 192.168.1.0/24 from being advertised in a type 3 LSA. The second line permits everything else: every address with a mask from length 0 to length 32 bits. This second line is required because there is a implicit "deny all" statement at the end of the prefix list. 192.168.1.0/24 is prevented from being sent in type 3 LSAs outside of area 25. Every other address is permitted.

## Case Study: Authentication

OSPF packets can be authenticated to prevent inadvertent or intentional introduction of bad routing information. Table 8-8 lists the types of authentication available. Null authentication (type 0), which means no authentication information is included in the packet header, is the default. Authentication using simple clear-text passwords (type 1) or MD5 cryptographic checksums (type 2) can be configured. When authentication is configured, it must be configured for an entire area.

If increased network security is the objective, type 1 authentication should be used only when devices within an area cannot support the more secure type 2 authentication. Clear-text authentication leaves the network vulnerable to a "sniffer attack," in which packets are captured by a protocol analyzer and the passwords are read (see Chapter 6 and especially Figure 6.8). However, type 1 authentication can be useful when performing OSPF reconfigurations. For example, separate passwords can be used on "old" OSPF routers and "new" OSPF routers sharing a common broadcast network to prevent them from talking to each other.

To configure type 1 authentication for an area, the command ip ospf authentication-key is used to assign a password of up to eight octets to each interface attached to the area. The passwords do not have to be the same throughout the area, but must be the same between neighbors. Type 1 authentication is then enabled by entering the area authentication command to the OSPF configuration.

Referring to Figure 8-48, type 1 authentication is enabled for areas 0 and 25. Example 8-68 displays Hurd's configuration.

## Example 8-68. Hurd is configured to use clear text authentication.

```
interface Ethernet 0/0
 ip address 10.8.1.1 255.255.255.0
 ip ospf authentication-key santafe

interface Serial 0/0.1
 ip address 172.20.1.1 255.255.255.252
 ip ospf authentication-key taos

interface Serial 0/0.2
 ip address 172.20.1.5 255.255.255.252
 ip ospf authentication-key abiquiu

router ospf 1
 network 10.8.0.0 0.0.255.255 area 0
 network 172.20.0.0 0.0.255.255 area 25
 area 25 range 172.16.0.0 255.240.0.0
 area 0 authentication
 area 25 authentication

ip route 172.16.0.0 255.240.0.0 null0
```

The password "santafe" is used between Hurd and Pena; "taos" is used between Hurd and Gorman, and "abiquiu" is used between Hurd and Okeeffe.

The MD5 algorithm, used by type 2 authentication, computes a hash value from the OSPF packet contents and a password (or key). This hash value is transmitted in the packet, along with a key ID and a non-decreasing sequence number. The receiver, knowing the same password, will calculate its own hash value. If nothing in the message has changed, the receiver's hash value should match the sender's value transmitted with the message. The key ID allows routers to reference multiple passwords, making password changes easier and more secure. An example of password migration is included in this case study. The sequence number prevents "replay attacks," in which OSPF packets are captured, modified, and retransmitted to a router.

To configure type 2 authentication for an area, the command ip ospf message-digest-key md5 assigns a password of up to 16 bytes and key ID between 1 and 255 to each interface attached to the area. Like type 1, the passwords do not have to be the same throughout the area, but both the key ID and the password must be the same between neighbors. Type 2 authentication is then enabled by entering the area authentication message-digest command to the OSPF configuration.

Hurd is configured to use type 2 authentication as shown in Example 8-69.

## Example 8-69. Hurd is configured to use MD5 authentication.

```
interface Ethernet 0/0
 ip address 10.8.1.1 255.255.255.0
```

```
 ip ospf message-digest-key 5 md5 santafe

interface Serial 0/0.1
 ip address 172.20.1.1 255.255.255.252
 ip ospf message-digest-key 10 md5 taos

interface Serial 0/0.2
 ip address 172.20.1.5 255.255.255.252
 ip ospf message-digest-key 15 md5 abiquiu

router ospf 1
 network 10.8.0.0 0.0.255.255 area 0
 network 172.20.0.0 0.0.255.255 area 25
 area 25 range 172.16.0.0 255.240.0.0
 area 0 authentication message-digest
 area 25 authentication message-digest

ip route 172.16.0.0 255.240.0.0 null0
```

The key allows the password to be changed without having to disable authentication. For example to change the password between Hurd and Okeeffe, the new password would be configured with a different key. Example 8-70 shows Hurd's configuration.

## Example 8-70. Hurd is configured with a new MD5 key.

```
interface Serial0/0.2
 ip address 172.20.1.5 255.255.255.252
 ip ospf message-digest-key 15 md5 abiquiu
 ip ospf message-digest-key 20 md5 steiglitz
```

Hurd will now send duplicate copies of each OSPF packet out S0/0.2; one will be authenticated with key 15, the other with key 20. When Hurd begins receiving OSPF packets from Okeeffe authenticated with key 20, it will stop sending packets with key 15. Once the new key is in use, the old key can be removed from both routers with the command no ip ospf message-digest-key 15 md5 abiquiu.

The passwords in an operational network should never be as predictable as the ones used in these examples. Adding the command service password-encryption to the configuration file of all routers using authentication is also wise. This change will cause the router to encrypt the passwords in any display of the configuration file, thereby guarding against the password being learned by simply observing a text copy of the router's configuration. With encryption, Example 8-71 is a display of what Hurd's configuration would include.

## Example 8-71. Password encryption is configured on Hurd to encrypt the password in the display of the configuration file.

```
service password-encryption
!
interface Ethernet0/0
 ip address 10.8.1.1 255.255.255.0
 ip ospf message-digest-key 5 md5 7 001712008105A0D03
!
interface Serial0/0.1
 ip address 172.20.1.1 255.255.255.252
 ip ospf message-digest-key 10 md5 7 03105A0415
!
```

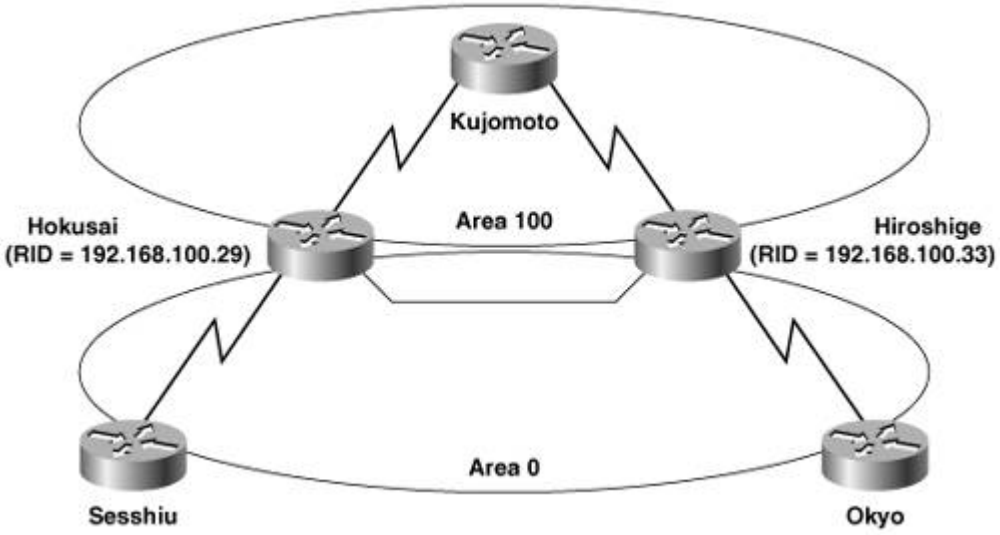
```
interface Serial0/0.2
 ip address 172.20.1.5 255.255.255.252
 ip ospf message-digest-key 20 md5 7 070E23455F1C1010
```

In the authentication configuration of the other protocols discussed in this book, key-chains were used to configure the passwords, or key-strings as they are called. OSPF does not support key-chain configuration at the time of this writing.

## Case Study: Virtual Links

Figure 8-50 shows an network with a poorly designed backbone area. If the link between routers Hokusai and Hiroshige fails, the backbone will be partitioned. As a result, routers Sesshiu and Okyo will be unable to communicate with each other. If these two routers are ABRs to separate areas, inter-area traffic between those areas will also be blocked.

Figure 8-50. A failure of the link between Hokusai and Hiroshige will partition the backbone area.




The best solution to this vulnerability is to add another link to the backbone areabetween Sesshiu and Okyo, for instance. An interim solution, until the backbone can be improved, is to create a virtual link between Hokusai and Hiroshige through area 100.

Virtual links are always created between ABRs, at least one of which must be connected to area 0.[31] At each ABR, the `area virtual-link` command, added to the OSPF configuration, specifies the area through which the virtual link will transit and the Router ID of the ABR at the far end of the link. A virtual link is configured between Hokusai and Hiroshige as shown in Example 8-72 and Example 8-73.

[31] When a virtual link is used to connect an area to the backbone through a non-backbone area, one of the ABRs will be between the two non-backbone areas.

Example 8-72. Hokusai's virtual link configuration.

```
router ospf 10
 network 192.168.100.1 0.0.0.0 area 0
```

```
 network 192.168.100.29 0.0.0.0 area 0
 network 192.168.100.21 0.0.0.0 area 100
 area 100 virtual-link 192.168.100.33
```

Example 8-73. Hiroshige's virtual link configuration.

```
router ospf 10
 network 192.168.100.2 0.0.0.0 area 0
 network 192.168.100.33 0.0.0.0 area 0
 network 192.168.100.25 0.0.0.0 area 100
 area 100 virtual-link 192.168.100.29
```

Packets will normally travel between Sesshiu and Okyo on the backbone link between Hokusai and Hiroshige. If that link fails, the virtual link will be used. Although each router sees the link as an unnumbered point-to-point network (Example 8-74), in reality the packets are being routed via Kujomoto.[32]

[32] The output of the **show ip ospf virtual-link** command might be slightly different, depending upon which IOS version you are using.

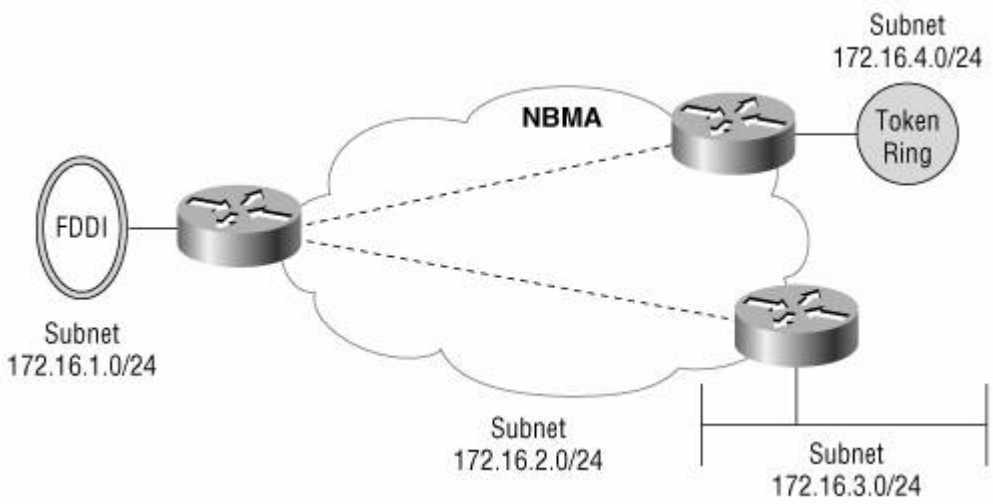
Example 8-74. The state of a virtual link can be observed with the command show ip ospf virtual-link.

```
Hokusai#show ip ospf virtual-link
Virtual Link OSPF_VL1 to router 192.168.100.33 is up
   Run as demand circuit
   DoNotAge LSA not allowed (Number of DCbitless LSA is 2).
   Transit area 100, via interface Serial0, Cost of using 128
   Transmit Delay is 1 sec, State POINT_TO_POINT,
   Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
      Hello due in 00:00:00
      Adjacency State FULL (Hello suppressed)
Hokusai#
```

## Case Study: OSPF on NBMA Networks

Nonbroadcast multiaccess networks such as X.25, Frame Relay, and ATM present a problem for OSPF. *Multiaccess* means that the NBMA "cloud" is a single network to which multiple devices are attached, the same as Ethernet or Token Ring networks (Figure 8-51). But unlike Ethernet and Token Ring, which are broadcast networks, *non-broadcast* means a packet sent into the network might not be seen by all other routers attached to the network. Because an NBMA network is multi-access, OSPF will want to elect a DR and BDR. But because an NBMA network is non-broadcast, there is no guarantee that all attached routers will receive the Hellos of all other routers. Therefore, all routers might not automatically learn about all its neighbors, and DR election would not function correctly.

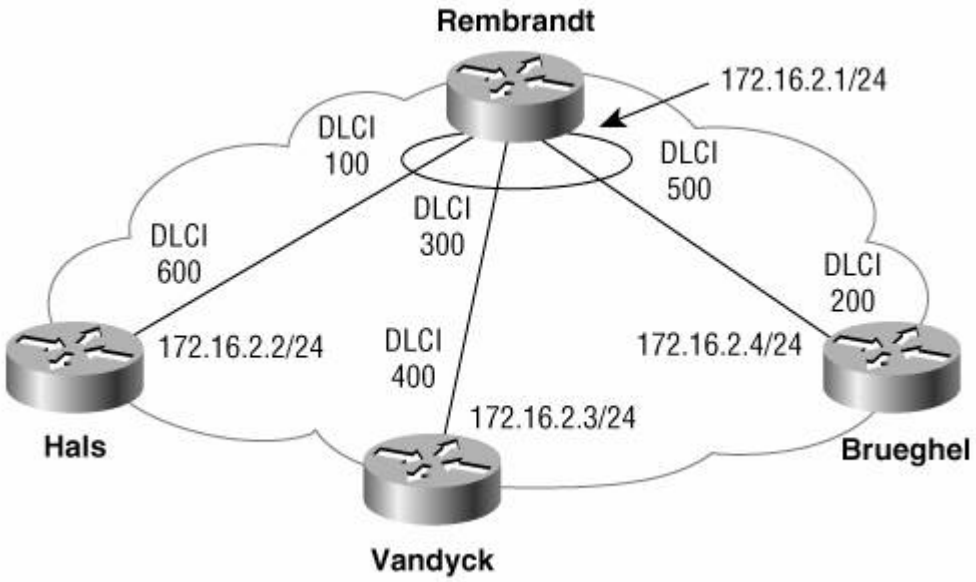
Figure 8-51. Routing protocols view NBMA networks as a single subnet to which multiple devices are connected. But when an NBMA network is partially meshed, as it is here, not all attached routers have direct connectivity with all other attached routers.




This section examines several solutions to the NBMA problem. The selection of a particular solution depends on the characteristics of the network upon which the solution is to be implemented.

The oldest solution, pertinent to pre-10.0 versions of the Cisco IOS, is to manually identify each router's neighbors and establish the DR, using the neighbor command. Figure 8-52 shows a Frame Relay network with four attached routers.

Figure 8-52. Several options exist for configuring OSPF on this NBMA network.




Because of the partially meshed hub-and-spoke configuration of the PVCs in Figure 8-52, Rembrandt must become the DR. As the hub, it is the only router directly connected to all the other routers. The configurations of the four routers are shown in Example 8-75 through Example 8-78.

Example 8-75. Rembrandt's configuration.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.1 255.255.255.0
 frame-relay map ip 172.16.2.2 100
 frame-realy map ip 172.16.2.3 300
 frame-relay map ip 172.16.2.4 500
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
 neighbor 172.16.2.2
 neighbor 172.16.2.3
 neighbor 172.16.2.4
```

Example 8-76. Hals's configuration specifying a neighbor priority.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.2 255.255.255.0
 frame-relay map ip 172.16.2.1 600
 frame-relay map ip 172.16.2.3 600
 frame-relay map ip 172.16.2.4 600
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
 neighbor 172.16.2.1 priority 10
```

Example 8-77. Vandyck's configuration specifying a neighbor priority.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.3 255.255.255.0
 frame-relay map ip 172.16.2.1 400
 frame-relay map ip 172.16.2.2 400
 frame-relay map ip 172.16.2.4 400
!
router ospf 1
 network 172.16.0.0 0.0.255.255 are a 0
 neighbor 172.16.2.1 priority 10
```

Example 8-78. Brueghel's configuration specifying a neighbor priority.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.4 255.255.255.0
 frame-relay map ip 172.16.2.1 200
 frame-relay map ip 172.16.2.2 200
 frame-relay map ip 172.16.2.3 200
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
 neighbor 172.16.2.1 priority 10
```

The neighbor command configures Rembrandt with the IP addresses of the interfaces of its three neighbors. The default priority is zero; by not changing the default at Rembrandt, none of its

neighbors is eligible to become the DR or BDR.

The other three routers are configured with only Rembrandt as a neighbor; the priority is set to 10, which means Rembrandt will become the DR. By making Rembrandt the DR, the PVCs exactly emulate the adjacencies that would have formed if the four routers had been connected to a broadcast multi-access network. OSPF packets will now be unicast to the configured neighbor addresses.

To repeat, the neighbor command is necessary only with old (pre-10.0) versions of IOS. A newer solution is to use the ip ospf network command to change the default OSPF network type.

One option with this command is to change the network type to broadcast, with ip ospf network broadcast entered at every Frame Relay interface. This change will cause the NBMA cloud to be viewed as a broadcast network; the configurations of the four routers are shown in Example 8-79 through Example 8-82.

### Example 8-79. Rembrandt's Frame Relay interface is configured as an OSPF broadcast network.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.1 255.255.255.0
 ip ospf network broadcast
 ip ospf priority 10
 frame-relay map ip 172.16.2.2 100 broadcast
 frame-realy map ip 172.16.2.3 300 broadcast
 frame-relay map ip 172.16.2.4 500 broadcast
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

### Example 8-80. Hals's Frame Relay interface is configured as an OSPF broadcast network.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.2 255.255.255.0
 ip ospf network broadcast
 ip ospf priority 0
 frame-relay map ip 172.16.2.1 600 broadcast
 frame-relay map ip 172.16.2.3 600 broadcast
 frame-relay map ip 172.16.2.4 600 broadcast
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

### Example 8-81. Vandyck's Frame Relay interface is configured as an OSPF broadcast network.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.3 255.255.255.0
 ip ospf network broadcast
 ip ospf priority 0
 frame-relay map ip 172.16.2.1 400 broadcast
 frame-relay map ip 172.16.2.2 400 broadcast
```

```
 frame-relay map ip 172.16.2.4 400 broadcast
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

## Example 8-82. Brueghel's Frame Relay interface is configured as an OSPF broadcast network.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.4 255.255.255.0
 ip ospf network broadcast
 ip ospf priority 0
 frame-relay map ip 172.16.2.1 200 broadcast
 frame-relay map ip 172.16.2.2 200 broadcast
 frame-relay map ip 172.16.2.3 200 broadcast
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

Note in this example that the priority of Rembrandt's interface is set to 10, and the priority of the other interfaces is set to 0. This will, again, ensure that Rembrandt is the DR. Note also that the static Frame Relay mapping commands are set to forward broadcast and multicast addresses.

An alternative to influencing the DR election is to implement a fully meshed topology in which every router has a PVC to every other router. From the standpoint of the router, this solution is actually the most efficient of all the NBMA implementation alternatives. The obvious drawback of this approach is monetary cost. If there are *n* routers, *n*(*n* 1)/2 PVCs will be necessary to create a fully meshed topology. For example, the 4 routers of Figure 8-52 would need 6 PVCs for a full mesh; 16 routers would require 120 PVCs.

Another option is to avoid the DR/BDR election process altogether, by changing the network type to point-to-multipoint. Point-to-multipoint networks treat the PVCs as a collection of point-to-point links; therefore, no DR/BDR election takes place. In multivendor environments, point-to-multipoint might be the only alternative to broadcast networks.

In the configurations in Example 8-83 through Example 8-86, the OSPF network type associated with each interface is changed to point-to-multipoint.

## Example 8-83. Rembrandt's Frame Relay interface is configured as an OSPF point-to-multipoint network.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.1 255.255.255.0
 ip ospf network point-to-multipoint
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

## Example 8-84. Hals's Frame Relay interface is configured as an OSPF point-to-multipoint network.

```
interface Serial0
encapsulation frame-relay
```

```
ip address 172.16.2.2 255.255.255.0
ip ospf network point-to-multipoint
!
router ospf 1
network 172.16.0.0 0.0.255.255 area 0
```

### Example 8-85. Vandyck's Frame Relay interface is configured as an OSPF point-to-multipoint network.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.3 255.255.255.0
 ip ospf network point-to-multipoint
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

### Example 8-86. Brueghel's Frame Relay interface is configured as an OSPF point-to-multipoint network.

```
interface Serial0
 encapsulation frame-relay
 ip address 172.16.2.4 255.255.255.0
 ip ospf network point-to-multipoint
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

These configurations take advantage of the Frame Relay inverse ARP function to dynamically map network-level addresses to the DLCIs, instead of using the static map commands shown in the previous examples. Static maps can still be used if desired.

The OSPF point-to-multipoint network type treats the underlying network as a collection of point-to-point links rather than a multi-access network, and OSPF packets are multicast to the neighbors. This situation can be problematic for networks whose connections are dynamic, such as Frame Relay SVCs or ATM SVCs. Beginning with IOS 11.3AA, this problem can be solved by declaring a network to be both point-to-multipoint and non-broadcast, as in the configurations in Example 8-87 through Example 8-90.

### Example 8-87. Rembrandt's Frame Relay interface is configured as an OSPF point-to-multipoint, non-broadcast network.

```
interface Serial0
 ip address 172.16.2.1 255.255.255.0
 encapsulation frame-relay
 ip ospf network point-to-multipoint non-broadcast
 map-group Leiden
 frame-relay lmi-type q933a
 frame-relay svc
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
 neighbor 172.16.2.2 cost 30
 neighbor 172.16.2.3 cost 20
 neighbor 172.16.2.4 cost 50
```

### Example 8-88. Hals's Frame Relay interface is configured as an OSPF point-to-multipoint, non-broadcast network.

```
interface Serial0
 ip address 172.16.2.2 255.255.255.0
 encapsulation frame-relay
 ip ospf network point-to-multipoint non-broadcast
 map-group Haarlem
 frame-relay lmi-type q933a
 frame-relay svc
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
 neighbor 172.16.2.1 priority 10
```

### Example 8-89. Vandyck's Frame Relay interface is configured as an OSPF point-to-multipoint, non-broadcast network.

```
interface Serial0
 ip address 172.16.2.3 255.255.255.0
 encapsulation frame-relay
 ip ospf network point-to-multipoint non-broadcast
 map-group Antwerp
 frame-relay lmi-type q933a
 frame-relay svc
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
 neighbor 172.16.2.1 priority 10
```

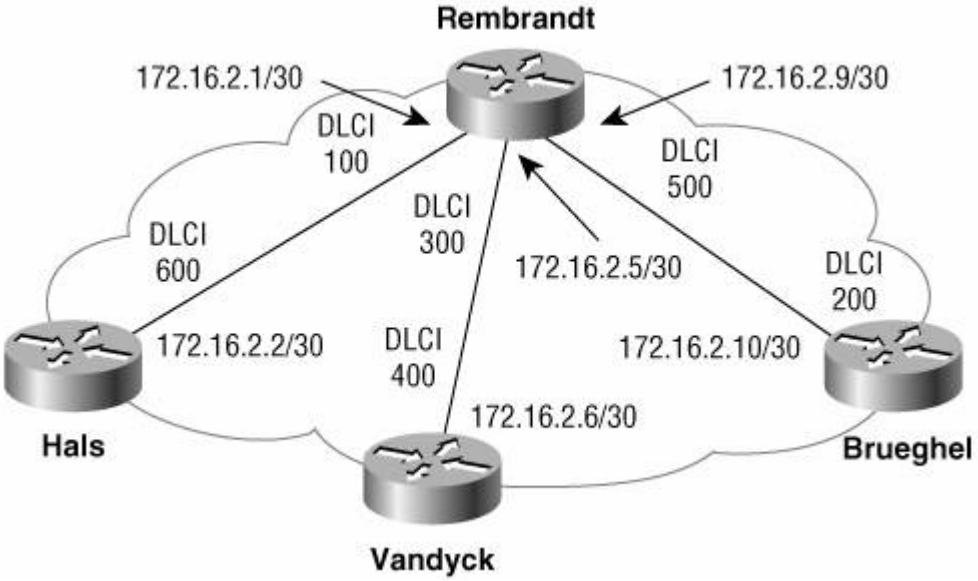
### Example 8-90. Brueghel's Frame Relay interface is configured as an OSPF point-to-multipoint, non-broadcast network.

```
interface Serial0
 ip address 172.16.2.4 255.255.255.0
 encapsulation frame-relay
 ip ospf network point-to-multipoint non-broadcast
 map-group Brussels
 frame-relay lmi-type q933a
 frame-relay svc
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
 neighbor 172.16.2.1 priority 10
```

Because the network is non-broadcast, neighbors are not discovered automatically and must be manually configured. Another feature introduced in IOS 11.3AA can be seen in Rembrandt's configuration: Cost can be assigned on a per VC basis with the `neighbor` command.

The last solution is to establish each PVC as an individual point-to-point network with its own subnet (Figure 8-53). This solution is accomplished with subinterfaces as shown in Example 8-91 though Example 8-94.

Figure 8-53. Point-to-point subinterfaces allow each PVC to be configured as an individual subnet and eliminate the problem of DR/BDR election on NBMA networks.




Example 8-91. Rembrandt is configured with point-to-point subinterfaces.

```
interface Serial0
 no ip address
 encapsulation frame-relay
 interface Serial0.100 point-to-point
  description -------------------------- to Hals
  ip address 172.16.2.1 255.255.255.252
  frame-relay interface-dlci 100
 interface Serial0.300 point-to-point
  description -------------------------- to Vandyck
  ip address 172.16.2.5 255.255.255.252
  frame-relay interface-dlci 300
 interface Serial0.500 point-to-point
  description -------------------------- to Brueghels
  ip address 172.16.2.9 255.255.255.252
  frame-relay interface-dlci 500
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

Example 8-92. Hals is configured with point-to-point subinterfaces.

```
interface Serial0
 no ip address
 encapsulation frame-relay
 interface Serial0.600
  description -------------------------- to Rembrandt
  ip address 172.16.2.2 255.255.255.252
  frame-relay interface-dlci 600
```

```
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

Example 8-93. Vandyck is configured with point-to-point subinterfaces.

```
interface Serial0
 no ip address
 encapsulation frame-relay
 interface Serial0.400
  description ---------------------------- to Rembrandt
  ip address 172.16.2.6 255.255.255.252
  frame-relay interface-dlci 400
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

Example 8-94. Brueghel is configured with point-to-point subinterfaces.

```
interface Serial0
 no ip address
 encapsulation frame-relay
 interface Serial0.200
  description ---------------------------- to Rembrandt
  ip address 172.16.2.10 255.255.255.252
  frame-relay interface-dlci 200
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
```

This configuration is the most easily managed of all the configurations of OSPF over NBMA networks. Some of the advantages are evident in the configuration code, such as the ability to use an interface number that corresponds to the DLCI and the inclusion of a description line. The major advantage, however, is the simple one-to-one relationship between routers.

An occasional objection to the use of subinterfaces is that each PVC must have its own subnet address. In most cases, this requirement should not be a problem, because OSPF supports VLSM. As the example shows, creating sub-subnets from the subnet address that was assigned to the cloud is an easy matter. And because the PVCs are now point-to-point links, IP unnumbered may be used as an alternative to subnet addresses. Another significant advantage is in the router's failure detection. On point-to-point subinterfaces, the routers on both ends of the point-to-point network can detect a failed virtual circuit (if the Frame Relay cloud provides sufficient signaling), and the routing protocol can converge to an alternate route, if one exists.

A more serious concern with subinterfaces is that they require more memory. The burden can be significant on small routers with limited memory.

## Case Study: OSPF over Demand Circuits

OSPF over demand circuits is easily configured by adding the command ip ospf demand-circuit to the interface connected to the demand circuit. Only one end of a point-to-point circuit, or the multipoint side of a point-to-multipoint circuit, needs to be declared a demand circuit. In most cases, OSPF over demand circuits should not be implemented across a broadcast medium. On such a network, the Hello packets cannot be suppressed, and the link will stay up.

If the virtual circuits in Figure 8-52 are Frame Relay SVCs, Rembrandt's configuration might be as shown in Example 8-95.

## Example 8-95. Rembrandt's OSPF demand-circuit configuration.

```
interface Serial0/0
 ip address 172.16.2.1 255.255.255.0
 encapsulation frame-relay
 ip ospf network point-to-multipoint non-broadcast
 ip ospf demand-circuit
 map-group Leiden
 frame-relay lmi-type q933a
 frame-relay svc
!
router ospf 1
 network 172.16.0.0 0.0.255.255 area 0
 neighbor 172.16.2.2 cost 30
 neighbor 172.16.2.3 cost 20
 neighbor 172.16.2.4 cost 50
```

Keep the following points in mind when implementing OSPF over demand circuits:

- LSAs with the DoNotAge bit set will be allowed into an area only if all LSAs in the area's link-state database have the DC-bit set. This setting ensures that all routers in the area are capable of understanding the DoNotAge bit.
- All routers of an area in which OSPF over demand circuits is implemented must be capable of supporting it.
- If OSPF over demand circuits is implemented in a non-stub area, the routers in all non-stub areas must support it. The reason is that the DC-bit in type 5 LSAs will be set, and these LSAs are flooded into all non-stub areas.
- An effort should be made to implement demand circuits only within stub, totally stubby, or NSSA areas. Such an implementation negates the need for all routers within the OSPF domain to support OSPF over demand circuits. It also minimizes the number of changed LSAs received as the result of topology changes in other areas, and hence prevents excess uptime of the demand circuit.
- If OSPF over demand circuits is configured and a virtual link is configured to cross the demand circuit, the virtual link will also be treated as a demand circuit. Otherwise, the virtual link traffic would keep the circuit up.
- OSPF refreshes its LSAs every 30 minutes to guard against an LSA becoming corrupted while it resides in the link-state database. Because DoNotAge LSAs are not refreshed across a demand circuit, this robustness feature is lost.
- The refresh process occurs on each side of a demand circuit out all other interfaces, but LSAs are not refreshed across the link. As a result, the sequence numbers of otherwise identical LSAs on each side of the link might be different. Network management stations may use certain MIB variables[33] to verify database synchronization; if the sequence numbers do not match across the databases, an error might be falsely reported.

[33] Specifically, ospfExternLSACksumSum and ospfAreaLSACksumSum. These are sums of the individual LSA checksum fields. Because the checksum calculation includes the sequence number, and because the sequence numbers might be different, the checksums will also be different.

# Troubleshooting OSPF

Troubleshooting OSPF can sometimes be daunting, especially in a large network. However, a routing problem with OSPF is no different than a routing problem with any other routing protocol; the cause will be one of the following:

- Missing route information
- Inaccurate route information

An examination of the route table is still the primary source of troubleshooting information. Using the `show ip ospf database` command to examine the various LSAs will also yield important information. For example, if a link is unstable, the LSA advertising will change frequently. This condition is reflected in a sequence number that is conspicuously higher than that of the other LSAs. Another sign of instability is an LSA whose age never gets very high.

Keep in mind that the link-state database of every router within an area is the same. So unless yo suspect that the database itself is being corrupted on some routers, you can examine the link-stat database for the entire area by examining a single router's link-state database. Another good practice is to keep a copy (hard or soft) of the link-state database for each area.

When examining an individual router's configuration, consider the following:

- Do all interfaces have the correct addresses and masks?
- Do the `network area` statements have the correct inverse masks to match the correct interfaces?
- Do the `network area` statements put all interfaces into the correct areas?
- Are the `network area` statements in the correct order?

When examining adjacencies (or the lack thereof), consider these questions:

- Are Hellos being sent from both neighbors?
- Are the timers set the same between neighbors?
- Are the optional capabilities set the same between neighbors?
- Are the interfaces configured on the same subnet (that is, do the address/mask pairs belong to the same subnet)?
- Are the neighboring interfaces of the same network type?
- Is a router attempting to form an adjacency with a neighbor's secondary address?
- If authentication is being used, is the authentication type the same between neighbors? Are the passwords and (in the case of MD5) the keys the same? Is authentication enabled on all routers within the area?
- Are any access lists blocking OSPF?
- If the adjacency is across a virtual link, is the link configured within a stub area?

If a neighbor or adjacency is suspected of being unstable, adjacencies can be monitored with the command debug ip ospf adj. However, this command can often present more information than you want, as Example 8-96 shows. The state changes of a neighbor are recorded in great detail. If monitoring is to be performed over an extended period, this wealth of information can overflow a router's internal logging buffers. Beginning with IOS 11.2, adjacencies can be monitored by adding the command log-adjacency-changes [detail] under a router's OSPF configuration. This command will keep a simpler log of adjacency changes, as shown in Example 8-97 and Example 8-98.

### Example 8-96. This debug output from *debug ip ospf adj* shows the result of temporarily disconnecting and then reconnecting a neighbor's Ethernet interface.

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0, changed state to down
OSPF: Interface Ethernet0/0 going Down
OSPF: 172.16.2.2 address 10.8.1.1 on Ethernet0/0 is dead, state DOWN
OSPF: Neighbor change Event on interface Ethernet0/0
OSPF: DR/BDR election on Ethernet0/0
OSPF: Elect BDR 0.0.0.0
OSPF: Elect DR 172.16.2.3
OSPF: Elect BDR 0.0.0.0
OSPF: Elect DR 172.16.2.3
        DR: 172.16.2.3 (Id) BDR: none
OSPF: 172.16.2.3 address 10.8.1.2 on Ethernet0/0 is dead, state DOWN
OSPF: Neighbor change Event on interface Ethernet0/0
OSPF: DR/BDR election on Ethernet0/0
OSPF: Elect BDR 0.0.0.0
OSPF: Elect DR 0.0.0.0
        DR: none BDR: none
OSPF: Remember old DR 172.16.2.3 (id)
OSPF: Build router LSA for area 0, router ID 172.16.2.2, seq 0x80000035
OSPF: Build router LSA for area 25, router ID 172.16.2.2,
seq 0x80000005
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0, changed state to up
OSPF: Interface Ethernet0/0 going Up
OSPF: Send with youngest Key 5
OSPF: Build router LSA for area 0, router ID 172.16.2.2, seq 0x80000036
OSPF: Build router LSA for area 25, router ID 172.16.2.2, seq 0x80000006
OSPF: Send with youngest Key 5
OSPF: Send with youngest Key 5
OSPF: Send with youngest Key 5
OSPF: Rcv DBD from 172.16.2.3 on Ethernet0/0 seq 0x728 opt 0x52 flag 0x7 len 32 mtu
1500 state INIT
OSPF: 2 Way Communication to 172.16.2.3 on Ethernet0/0, st ate 2WAY
OSPF: Nbr state is 2WAY
OSPF: Rcv DBD from 172.16.2.3 on Ethernet0/0 seq 0x728 opt 0x52 flag 0x7 len 32 mtu
1500 state 2WAY
OSPF: Nbr state is 2WAY
OSPF: end of Wait on interface Ethernet0/0
OSPF: DR/BDR election on Ethernet0/0
OSPF: Elect BDR 172.16.2.2
OSPF: Elect DR 172.16.2.3
OSPF: Elect BDR 172.16.2.2
OSPF: Elect DR 172.16.2.3
        DR: 172.16.2.3 (Id) BDR: 172.16.2.2 (Id)
OSPF: Send DBD to 172.16.2.3 on Ethernet0/0 seq 0x1B85 opt 0x52 flag 0x7 len 32
OSPF: Send with youngest Key 5
OSPF: Send with youngest Key 5
OSPF: Rcv DBD from 172.16.2.3 on Ethernet0/0 seq 0x728 opt 0x52 flag 0x7 len 32 mtu
```

```
1500 state EXSTART
OSPF: NBR Negotiation Done. We are the SLAVE
OSPF: Send DBD to 172.16.2.3 on Ethernet0/0 seq 0x728 opt 0x52 flag 0x2 len 292
OSPF: Send with youngest Key 5
OSPF: Rcv DBD from 172.16.2.3 on Ethernet0/0 seq 0x729 opt 0x52 flag 0x3 len 272
mtu 1500 state EXCHANGE
OSPF: Send DBD to 172.16.2.3 on Ethernet0/0 seq 0x729 opt 0x52 flag 0x0 len 32
OSPF: Send with youngest Key 5
OSPF: Send with youngest Key 5
OSPF: Database request to 172.16.2.3
OSPF: sent LS REQ packet to 10.8.1.2, length 12
OSPF: Send with youngest Key 5
OSPF: Rcv DBD from 172.16.2.3 on Ethernet0/0 seq 0x72A opt 0x52 flag 0x1 len 32 mtu
1500 state EXCHANGE
OSPF: Exchange Done with 172.16.2.3 on Ethernet0/0
OSPF: Send DBD to 172.16.2.3 on Ethernet0/0 seq 0x72A opt 0x52 flag 0x0 len 32
OSPF: Send with youngest Key 5
OSPF: Synchronized with 172.16.2.3 on Ethernet0/0, state FULL
OSPF: Build router LSA for area 0, router ID 172.16.2.2, seq 0x80000037
```

Example 8-97. These logging messages, resulting from the OSPF configuration command log-adjacency-changes, show the same neighbc failure as depicted in Example 8-96, but with much less detail.

```
Hurd#show logging
Syslog logging: enabled (0 messages dropped, 1 messages rate-limited, 0 flushes,

 0 overruns, xml disabled)
    Console logging: level debugging, 248 messages logged, xml disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled
    Buffer logging: level debugging, 5 messages logged, xml disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages: disabled
    Trap logging: level informational, 99 message lines logged

Log Buffer (4096 bytes):

%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from FULL to DOWN,
 Neighbor Down: Interface down or detached
%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from LOADING to FULL,
 Loading Done
```

Example 8-98. These logging messages, resulting from the OSPF configuration command log-adjacency-changes also show the same neighbor failure as depicted in Example 8-96, but have more detail than the log in Example 8-97.

```
Hurd#show logging
Syslog logging: enabled (0 messages dropped, 1 messages rate-limited, 0 flushes,

 0 overruns, xml disabled)
    Console logging: level debugging, 248 messages logged, xml disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled
    Buffer logging: level debugging, 5 messages logged, xml disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages: disabled
    Trap logging: level informational, 99 message lines logged
```

```
Log Buffer (4096 bytes):

%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from FULL to DOWN,
 Neighbor Down: Interface down or detached
%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from DOWN to INIT,
 Received Hello
%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from INIT to 2WAY,
 2-Way Received
%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from 2WAY to EXSTART,
 AdjOK?
%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from EXSTART to
 EXCHANGE, Negotiation Done
%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from EXCHANGE to
 LOADING, Exchange Done
%OSPF-5-ADJCHG: Process 1, Nbr 172.16.2.3 on Ethernet0/0 from LOADING to FULL,
 Loading Done
```

If you suspect that a link-state database is corrupted or that two databases are not synchronized, you can use the `show ip ospf database database-summary` command to observe the number of LSAs in each router's database. For a given area, the number of each LSA type should be the same in all routers. Next, the command `show ip ospf database` will show the checksums for every LSA in a router's database. Within a given area, each LSA's checksum should be the same ir every router's database. Verifying this status can be excruciatingly tedious for all but the smallest databases. Luckily, there are MIBs,[34] which can report the sum of a database's checksums to an SNMP management platform. If all databases in an area are synchronized, this sum should be the same for each database.

[34] Namely, ospfExternLsaCksumSum and ospfAreaLsaCksumSum.

When examining an area-wide problem, consider the following issues:

- Is the ABR configured correctly?
- Are all routers configured for the same area type? For example, if the area is a stub area, all routers must have the `area stub` command.
- If address summarization is configured, is it correct?

If performance is a problem, check the memory and CPU utilization on the routers. If memory utilization is above 70 percent, the link-state database might be too large; if CPU utilization is consistently above 60 percent, instabilities could exist in the topology. If memory or CPU surpasse the 50 percent mark, the network administrator should analyze the cause of the performance stre and, based on the results of the analysis, should begin planning corrective upgrades.

Stub areas and address summarization can help to both reduce the size of the link-state database and to contain instabilities. The processing of LSAs, not the SPF algorithm, puts the most burden c an OSPF router. Taken individually, type 1 and type 2 LSAs would be more processor-intensive tha summary LSAs. However, type 1 and 2 LSAs tend to be grouped, whereas summary LSAs are sent in individual packets. As a result, in reality, summary LSAs are more processor-intensive.

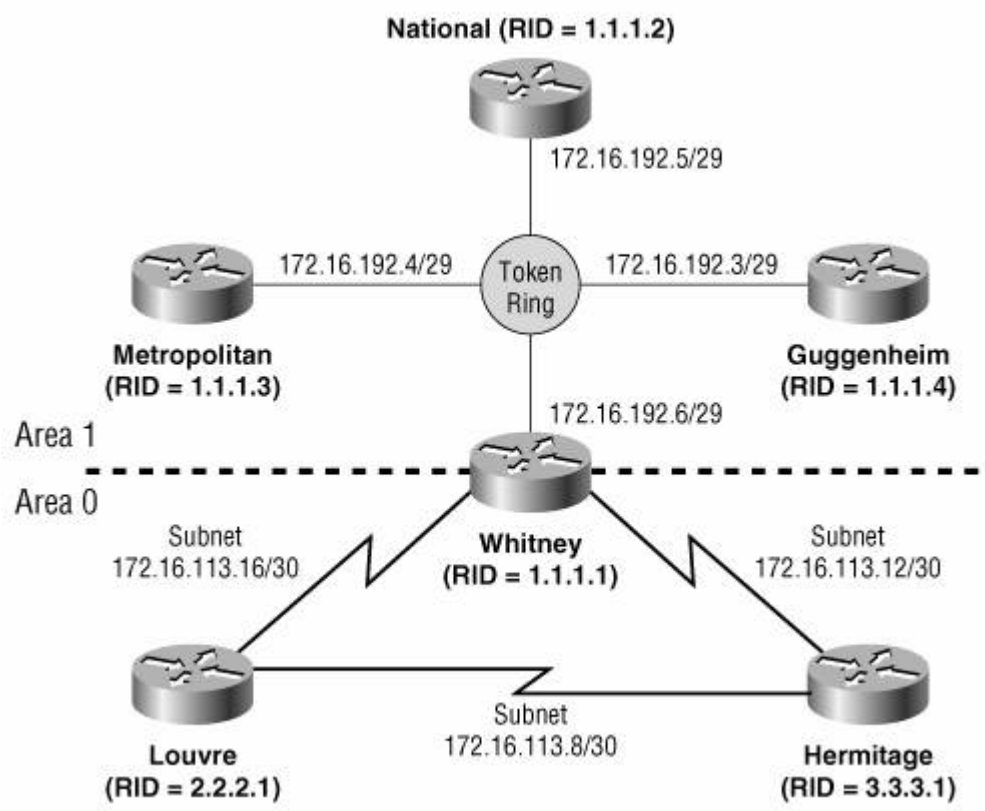
The following case studies demonstrate the most frequently used techniques and tools for troubleshooting OSPF.

## Case Study: An Isolated Area

Intra-area packets can be routed within area 1 of Figure 8-54, but all attempts at inter-area communications fail. Suspicion should immediately fall on area 1's ABR. This suspicion is reinforce

by the fact that the Internal Routers have no router entry for an ABR (Example 8-99).

Figure 8-54. The end systems and routers within area 1 can communicate, but no traffic is being passed to or from area 0.




Example 8-99. The command *show ip ospf border-routers* checks the internal route table of the Internal Routers. No router entry for an ABR i shown.

```
National#show ip ospf border-routers

OSPF Process 8 internal Routing Table

Codes: i - Intra-area route, I - Inter-area route

National#
```

The next step is to verify that the physical link to the ABR is operational and that OSPF is working properly. The same Internal Router's neighbor table (Example 8-100) shows that the neighbor sta of the ABR is full, indicating that an adjacency exists. In fact, the ABR is the DR for the Token Ring network. The existence of an adjacency confirms that the link is good and that OSPF Hellos are being exchanged with the proper parameters.

Example 8-100. The neighbor table of router National indicates that the ABR (1.1.1.1) is fully adjacent.

```
National#show ip ospf neighbor
Neighbor ID  Pri   State     Dead Time   Address         Interface
1.1.1.1        1   FULL/DR   00:00:33    172.16.192.6    TokenRing0
1.1.1.3        1   FULL/BDR  00:00:34    172.16.192.4    TokenRing0
1.1.1.4        1   FULL/ -   00:00:30    172.16.192.3    TokenRing0
National#
```

Other evidence relevant to the problem can be found in National's database and its route table. Th database (Example 8-101) contains only Router (type 1) and Network (type 2) LSAs. No Network Summary (type 3) LSAs, which advertise destinations outside of the area, are recorded. At the same time, there are LSAs originated by Whitney (1.1.1.1). This information again indicates that Whitney is adjacent but is not passing information from area 0 into area 1.

## Example 8-101. National's link-state database also shows that Whitney is adjacent, but is not advertising inter-area destinations.

```
National#show ip ospf database

        OSPF Router with ID (1.1.1.2) (Process ID 8)

                      Router Link States (Area 1)

Link ID         ADV Router  Age   Seq#         Checksum  Link count
172.16.192.6    1.1.1.1     132   0x80000034   0xAC4D    3
172.16.219.120  1.1.1.2 1   458   0x8000002B   0x6B46    2

                      Net Link States (Area 1)

Link ID         ADV Router  Age   Seq#         Checksum
172.16.192.6    1.1.1.1     132   0x8000002E   0x2078
National#
```

The only destinations outside of area 1 in National's route table (Example 8-102) are the serial lin attached to Whitney. Yet another clue is revealed here: The route entries are tagged as intra-area routes (O); if they were in area 0, as Figure 8-54 shows they should be, they would be tagged as inter-area routes (O IA). The problem is apparently on the area 0 side of the ABR.

## Example 8-102. Whitney is advertising the subnets of its serial interfaces, but they are being advertised as intra-area destinations.

```
National#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/16 is variably subnetted, 4 subnets, 3 masks
C       172.16.219.112/28 is directly connected, Serial0
C       172.16.192.0/29 is directly connected, TokenRing0
O       172.16.113.12/30 [110/70] via 172.16.192.6, 09:32:15, TokenRing0
```

```
O       172.16.113.16/30 [110/70] via 172.16.192.6, 09:32:15, TokenRing0
National#
```

An examination of Whitney's serial interfaces (Example 8-103) reveals the problem, if not the cause of the problem. Both interfaces, which should be in area 0, are instead in area 1. Both interfaces are connected to topological neighbors (Louvre and Hermitage), but no OSPF neighbors are recorded. Error messages are being displayed regularly, indicating that Whitney is receiving Hellos from Louvre and Hermitage; those Hellos have their Area fields set to zero, causing a mismatch.

### Example 8-103. Whitney's serial interfaces are configured in area 1 instead of area 0; this configuration is causing error messages when area 0 Hellos are received.

```
Whitney#show ip ospf interface serial 0
Serial0 is up, line protocol is up
  Internet Address 172.16.113.18/30, Area 1

  Process ID 8, Router ID 1.1.1.1, Network Type POINT_TO_POINT, Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:03
  Index 1/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 2, maximum is 2
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)

Whitney#show ip ospf interface serial 1
Serial0 is up, line protocol is up
  Internet Address 172.16.113.14/30, Area 1

  Process ID 8, Router ID 1.1.1.1, Network Type POINT_TO_POINT, Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_POINT,
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:06
  Index 1/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 2, maximum is 2
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 0, Adjacent neighbor count is 0
  Suppress hello for 0 neighbor(s)
Whitney#

%OSPF-4-ERRRCV: Received invalid packet: mismatch area ID,

 from backbone area must be virtual-link but not found from 172.16.113.13, Serial1

%OSPF-4-ERRRCV: Received invalid packet: mismatch area ID,

from backbone area must be virtual-link but not found from 172.16.113.17, Serial0
```

Whitney's OSPF configuration is shown in Example 8-104.

Example 8-104. Whitney's OSPF configuration.

```
router ospf 8
 network 172.16.0.0 0.0.255.255 area 1
 network 172.16.113.0 0.0.0.255 area 0
```

At first glance, this configuration might appear to be fine. However, recall from the first configuration case study that the network area commands are executed consecutively. The second network area command affects only interfaces that do not match the first command. Witl this configuration, all interfaces match the first network area command and are placed into area 1. The second command is never applied.

A correct configuration is shown in Example 8-105.

Example 8-105. Whitney's corrected OSPF configuration.

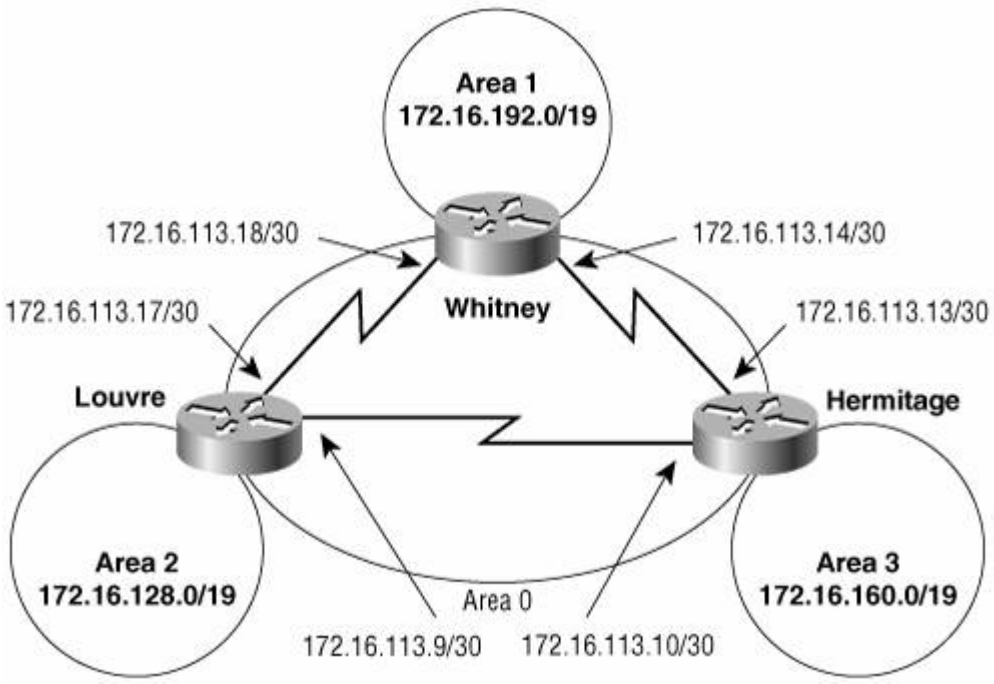
```
router ospf 8
 network 172.16.192.0 0.0.0.255 area 1
 network 172.16.113.0 0.0.0.255 area 0
```

There are, of course, several valid configurations. The important point is that the first network area command must be specific enough to match only the address of the area 1 interface, and no the addresses of the area 0 interfaces.

## Case Study: Misconfigured Summarization

Figure 8-55 shows a backbone area and three attached areas. To reduce the size of the link- state database and to increase the stability of the network, summarization will be used between areas.

Figure 8-55. The summary addresses shown for each area will be advertised into area 0. Area 0 will also be summarized into the other areas.




The individual subnets of the three nonbackbone areas are summarized with the addresses shown in Figure 8-55. For example, a few of the subnets of area 1 may be

172.16.192.0/29
172.16.192.160/29
172.16.192.248/30
172.16.217.0/24
172.16.199.160/29
172.16.210.248/30

Figure 8-56 shows that these subnet addresses can all be summarized with 172.16.192.0/19.

## Figure 8-56. A few of the subnet addresses that are summarized with 172.16.192.0/19. The bold type indicates the network bits of each address.

```
10101100000100001100000000000000  =  172.16.192.0/29
10101100000100001100000011111000  =  172.16.192.248/30
10101100000100001101100100000000  =  172.16.217.0/24
10101100000100001100011110100000  =  172.16.199.160/29
10101100000100001101001011111000  =  172.16.210.248/30
10101100000100001100000000000000  =  172.16.192.0/19
```

Whitney's configuration is shown in Example 8-106

## Example 8-106. Whitney's OSPF configuration with address summarization.

```
router ospf 8
 network 172.16.192.0 0.0.0.255 area 1
 network 172.16.113.0 0.0.0.255 area 0
 area 1 range 172.16.192.0 255.255.224.0
```

```
area 0 range 172.16.113.0 255.255.224.0
```

The other three ABRs are configured similarly. Each ABR will advertise the summary address of its attached non-backbone area into area 0 and will also summarize area 0 into the non-backbone area.

Example 8-107 shows that there is a problem. When the route table of one of area 1's Internal Routers is examined, area 0 is not being summarized properly (area 1's internal subnets are not shown, for clarity). Although the summary addresses for areas 2 and 3 are present, the individual subnets of area 0 are in the table instead of its summary address.

## Example 8-107. The individual subnets of area 0, instead of the expecte summary address, are recorded in the route table of one of area 1's internal routers.

```
National#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/16 is variably subnetted, 7 subnets, 4 masks
O IA    172.16.160.0/19 [110/80] via 172.16.192.6, 09:32:15, TokenRing0
O IA    172.16.128.0/19 [110/80] via 172.16.192.6, 09:32:15, TokenRing0

C       172.16.192.0/29 is directly connected, TokenRing0
O IA    172.16.113.12/30 [110/70] via 172.16.192.6, 09:32:15, TokenRing0
O IA    172.16.113.8/30 [110/134] via 172.16.192.6, 09:32:15, TokenRing0
O IA    172.16.113.16/30 [110/70] via 172.16.192.6, 09:32:15, TokenRing0
National#
```

You can see that the area range command for area 0 is the problem when you examine the three subnets of area 0 in binary (Figure 8-57).

## Figure 8-57. The subnets of area 0, the configured summary mask, and the correct summary address.

```
10101100000100000111000100001000  =  172.16.113.8/30
10101100000100000111000100001100  =  172.16.113.12/30
10101100000100000111000100010000  =  172.16.113.16/30
11111111111111111111000000000000  =  255.255.224.0
10101100000100000110000000000000  =  172.16.96.0
```

The problem is that the summary address specified in the area range command (172.16.113.0) is more specific than the accompanying mask (255.255.224.0). The correct address to use with the 19-bit mask is 172.16.96.0 (see Example 8-108).

## Example 8-108. Whitney's OSPF configuration with corrected address

summarization.

```
router ospf 8
 network 172.16.192.0 0.0.0.255 area 1
 network 172.16.113.0 0.0.0.255 area 0
 area 1 range 172.16.192.0 255.255.224.0
 area 0 range 172.16.96.0 255.255.224.0
```

Example 8-109 shows the resulting route table. There are other options for area 0's summary address. For example, 172.16.113.0/24 and 172.16.113.0/27 are both legitimate. The most appropriate summary address depends on the priorities of the network design. In the case of the network of Example 8-101, 172.16.96.0/19 might be selected for consistencyall summary addresses have a 19-bit mask. On the other hand, 172.16.113.0/27 might be selected for better scalability; five more subnets can be added to the backbone under this summary address, leaving wider range of addresses to be used elsewhere in the network.

## Example 8-109. Area 0 is now being summarized correctly.

```
National#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
O IA    172.16.160.0/19 [110/80] via 172.16.192.6, 00:25:09, TokenRing0
O IA    172.16.128.0/19 [110/80] via 172.16.192.6, 09:32:15, TokenRing0

C       172.16.192.0/29 is directly connected, TokenRing0
O IA    172.16.96.0/19 [110/70] via 172.16.192.6, 00:00:10, TokenRing0
National#
```

PREV NEXT

# Looking Ahead

When link-state routing protocols are mentioned, most people first think of OSPF. However, it is not the only link-state protocol for IP. The ISO's Intermediate-System to Intermediate-System (IS-IS), although designed to route other protocols, can route IP. Chapter 10, "Integrated IS-IS," examines this lesser known link-state routing protocol.

# Summary Table: Chapter 8 Command Review

| Command | Description |
|---|---|
| area *area-id* authentication [message-digest] | Enables type 1 or type 2 authentication for an area. |
| area *area-id* default-cost *cost* | Specifies a cost for the default route sent into a stub area by an ABR. |
| area *area-id* filter-list prefix *prefix_list_name* [out \| in] | Defines an LSA type 3 filter list. |
| area *area-id* nssa [no-redistribution] [default-information-originate] [no-summary] [translate type7 suppress-fa] | Configures an area as not-so-stubby (NSSA). |
| area *area-id* range *address mask*[advertise \| not-advertise] [cost] | Summarizes addresses into or out of an area. The cost of the summary address can be specified. |
| area area-id *stub* [no-summary] | Configures an area as a stub or totally stubby area. |
| area *area-id* virtual-link *router-id* | Defines a virtual link between ABRs. |
| debug ip ospf adj | Shows the events involved in the building or breaking of an OSPF adjacency. |
| [no] discard-route {internal \| external} | No discard-route removes the automatically created static route to the Null interface. |
| ip ospf authentication-key *password* | Assigns a password to an OSPF interface for use with type 1 authentication. |
| ip ospf cost *cost* | Specifies the outgoing cost of an OSPF interface. |
| ip ospf dead-interval *seconds* | Specifies the OSPF RouterDeadInterval for an interface. |
| ip ospf demand-circuit | Configures an interface as an OSPF demand circuit. |
| ip ospf hello-interval *seconds* | Specifies the OSPF HelloInterval for an interface. |
| ip ospf message-digest-key *key-id* md5 *key* | Specifies an interface's key ID and key (password) for use with type 2 authentication. |
| ip ospf name-lookup | Enables the reverse DNS lookup of names to match Router IDs in certain show commands. |
| ip ospf network [broadcast] [nonbroadcast] [point-to-multipoint] | Configures the OSPF network type. |

| Command | Description |
| --- | --- |
| ip ospf priority *number* | Sets the router priority of an interface for use in the DR/BDR election process. |
| ip ospf retransmit-interval *seconds* | Sets an interface's OSPF RxmtInterval. |
| ip ospf transmit-delay *seconds* | Sets an interface's OSPF InfTransDelay. |
| ip prefix-list *prefix_list_name* [seq *num*] {deny \| permit} *address/length* | Defines which addresses to permit or deny in a prefix list. |
| log-adjacency-changes [detail] | Logs neighbor state changes. |
| maximum-paths | Sets the number of paths over which OSPF performs load balancing. |
| neighbor *ip-address*[priority *number*] [poll-interval *seconds*] [cost *cost*] | Manually informs a router of its neighbors on a non-broadcast network. |
| network *address inverse-mask* area *area-id* | Specifies the interfaces on which OSPF is to run and specifies the area to which the interface is connected. |
| ospf auto-cost reference-bandwidth *reference-bandwidth* | Changes the default OSPF reference bandwidth used for the calculation of link costs. |
| router ospf *process-id* | Enables an OSPF routing process. |
| show ip ospf [*process-id*] | Displays general information about an OSPF routing process. |
| show ip ospf border-routers | Displays a router's internal OSPF route table. |
| show ip ospf [*process-id area-id*] database | Displays all entries in the OSPF link-state database. |
| show ip ospf [*process-id area-id*] database router [*link state-id*] | Displays type 1 LSAs in the OSPF link-state database. |
| show ip ospf [*process-id area-id*] database network [*link state-id*] | Displays type 2 LSAs in the OSPF link-state database. |
| show ip ospf [*process-id area-id*] database summary [*link state-id*] | Displays type 3 LSAs in the OSPF link-state database. |
| show ip ospf [*process-id area-id*] database asbr-summary [*link state-id*] | Displays type 4 LSAs in the OSPF link-state database. |
| show ip ospf [*process-id area-id*] database nssa-external [*link state-id*] | Displays type 7 LSAs in the OSPF link-state database. |
| show ip ospf [*process-id*] database external [*link state-id*] | Displays type 5 LSAs in the OSPF link-state database. |

| Command | Description |
| --- | --- |
| show ip ospf [*process-id area-id*] database database-summary | Displays the number of LSAs in the OSPF link-state database by type and by area ID. |
| show ip ospf interface [*type number*] | Displays OSPF-specific information about an interface. |
| show ip ospf neighbor [*type number*] [*neighbor-id*] [detail] | Displays information from the OSPF neighbor table. |
| show ip ospf virtual-links | Displays information about OSPF virtual links. |
| timer lsa-group-pacing *seconds* or timer pacing lsa-group *seconds* | Sets the minimum pacing time between two groups of LSAs whose refresh timers have expired. |

PREV NEXT

# Recommended Reading

Moy, J. "OSPF Version 2." RFC 2328: April 1998.

Moy, J. *OSPF: Anatomy of an Internet Routing Protocol*. Reading, Massachusetts: Addison-Wesley; 1998. Written by one of the original designers of OSPF and the author of the RFCs, this book is good reading not only for its excellent coverage of the protocol but also for its historic perspective. Chapter 3 is especially interesting, with its insider's perspective on the design, testing, and standardization of a routing protocol.

# Review Questions

1. What is an OSPF neighbor?
2. What is an OSPF adjacency?
3. What are the five OSPF packet types? What is the purpose of each type?
4. What is an LSA? How does an LSA differ from an OSPF Update packet?
5. What are LSA types 1 to 5 and LSA type 7? What is the purpose of each type?
6. What is a link-state database? What is link-state database synchronization?
7. What is the default HelloInterval?
8. What is the default RouterDeadInterval?
9. What is a Router ID? How is a Router ID determined?
10. What is an area?
11. What is the significance of area 0?
12. What is MaxAge?
13. What are the four OSPF router types?
14. What are the four OSPF path types?
15. What are the five OSPF network types?
16. What is a Designated Router?
17. How does a Cisco router calculate the outgoing cost of an interface?
18. What is a partitioned area?
19. What is a virtual link?
20. What is the difference between a stub area, a totally stubby area, and a not-so-stubby area?
21. What is the difference between OSPF network entries and OSPF router entries?
22. Why is type 2 authentication preferable over type 1 authentication?
23. Which three fields in the LSA header distinguish different LSAs? Which three fields in the LSA header distinguish different instances of the same LSA?

# Configuration Exercises

1 Table 8-13 shows the interfaces and addresses of 14 routers. Also shown is the OSPF area to which each interface is connected. The following facts apply:

All interfaces of each router are shown in the table.

If no area is shown (-), OSPF should not be running on the associated interface.

The second octet of the subnet address is the same as the area ID.

The first 16 bits of the address of every OSPF interface are specific to the area. For example, addresses with the prefix 10.30.x.x will be found only within area 30.

Write OSPF configurations for the routers in Table 8-13. (Tip: Draw a picture of the routers and subnets first.)

Table 8-13. The router information for Configuration Exercises 1 through 6

| Router | Interface | Address/Mask | Area ID |
|---|---|---|---|
| A | L0 | 10.100.100.1/32 | - |
| | E0 | 10.0.1.1/24 | 0 |
| | E1 | 10.0.2.1/24 | 0 |
| | E2 | 10.0.3.1/24 | 0 |
| | E3 | 10.0.4.1/24 | 0 |
| B | L0 | 10.100.100.2/32 | - |
| | E0 | 10.0.1.2/24 | 0 |
| | E1 | 10.5.1.1/24 | 5 |
| | S0 | 10.5.255.13/30 | 5 |
| | S1 | 10.5.255.129/30 | 5 |
| C | L0 | 10.100.100.3/32 | - |
| | E0 | 10.0.2.2/24 | 0 |
| | E1 | 10.10.1.1/24 | 10 |
| | S0 | 10.30.255.249/30 | 30 |
| D | L0 | 10.100.100.4/32 | - |
| | E0 | 10.0.3.2/24 | 0 |

| Router | Interface | Address/Mask | Area ID |
|---|---|---|---|
| | E1 | 10.20.1.1/24 | 20 |
| E | L0 | 10.100.100.5/32 | - |
| | E0 | 10.0.4.2/24 | 0 |
| | S0 | 10.15.255.1/30 | 15 |
| F | L0 | 10.100.100.6/32 | - |
| | E0 | 10.5.5.1/24 | 5 |
| | S0 | 10.5.255.130/30 | 5 |
| | S1 | 10.5.255.65/30 | 5 |
| G | L0 | 10.100.100.7/32 | - |
| | E0 | 10.10.1.58/24 | 10 |
| | S0 | 10.10.255.5/30 | - |
| H | L0 | 10.100.100.8/32 | - |
| | E0 | 10.20.1.2/24 | 20 |
| | E1 | 10.20.100.100/27 | 20 |
| | S0 | 10.20.255.225/30 | - |
| I | L0 | 10.100.100.9/32 | - |
| | E0 | 10.35.1.1/24 | 35 |
| | S0 | 10.5.255.66/30 | 5 |
| J | L0 | 10.100.100.10/32 | - |
| | E0 | 10.15.227.50/24 | 15 |
| | S0 | 10.15.225.2 | 15 |
| K | L0 | 10.100.100.11/32 | - |
| | E0 | 10.30.1.1/24 | 30 |
| | S0[*] | 10.30.254.193/26 | 30 |
| L | L0 | 10.100.100.12/32 | - |
| | E0 | 10.30.2.1/24 | 30 |
| | S0[*] | 10.30.254.194/26 | 30 |
| M | L0 | 10.100.100.13/32 | - |
| | E0 | 10.30.3.1/24 | 30 |
| | S0[*] | 10.30.254.195/26 | 30 |
| | S1 | 10.30.255.250/30 | 30 |

| Router | Interface | Address/Mask | Area ID |
|---|---|---|---|
| N | L0 | 10.100.100.14/32 | - |
| | E0 | 10.30.4.1/24 | 30 |
| | S0[*] | 10.30.254.196/26 | 30 |

2 Configure summarization on all ABRs in Table 8-13.

3 Modify the configurations to make area 15 a stub area.

4 Modify the configurations to make area 30 a totally stubby area.

5 Interface S0 of router H is connected to a router running another routing protocol, and the routes learned from that protocol are being redistributed into OSPF. Modify the configurations as necessary to allow these redistributed routes to be advertised throughout the OSPF domain, but do not allow any type 5 LSAs to enter area 20.

6 The serial link between routers C and M is a very low bandwidth link. Modify the configurations so that OSPF treats this link as a demand circuit.

PREV NEXT

# Troubleshooting Exercises

1. OSPF is not working between two routers. When debugging is turned on, the messages show Example 8-110 are received every 10 seconds. What is the problem?

### Example 8-110. The debug messages for Troubleshooting Exercise

```
RTR_EX1#debug ip ospf adj
OSPF adjacency events debugging is on
RTR_EX1#
OSPF: Rcv pkt from 172.16.27.1, TokenRing0, area 0.0.0.25 : src not on the sa
  network
OSPF: Rcv pkt from 172.16.27.1, TokenRing0, area 0.0.0.25 : src not on the sa
  network
OSPF: Rcv pkt from 172.16.27.1, TokenRing0, area 0.0.0.25 : src not on the sa
  network
OSPF: Rcv pkt from 172.16.27.1, TokenRing0, area 0.0.0.25 : src not on the sa
  network
OSPF: Rcv pkt from 172.16.27.1, TokenRing0, area 0.0.0.25 : src not on the sa
  network
```

2. Explain what problem is indicated by the debug messages in Example 8-111.

### Example 8-111. The debug messages for Troubleshooting Exercise

```
RTR_EX2#debug ip ospf adj
OSPF adjacency events debugging is on
RTR_EX2#
OSPF: Hello from 172.16.27.195 with mismatched Stub/Transit area option bit
OSPF: Hello from 172.20.1.1 with mismatched Stub/Transit area option bit
OSPF: Hello from 172.16.27.195 with mismatched Stub/Transit area option bit
OSPF: Hello from 172.20.1.1 with mismatched Stub/Transit area option bit
OSPF: Hello from 172.16.27.195 with mismatched Stub/Transit area option bit
OSPF: Hello from 172.20.1.1 with mismatched Stub/Transit area option bit
OSPF: Hello from 172.16.27.195 with mismatched Stub/Transit area option bit
OSPF: Hello from 172.20.1.1 with mismatched Stub/Transit area option bit
```

3. Explain what problem is indicated by the error messages in Example 8-112.

### Example 8-112. The error messages for Troubleshooting Exercise 3

```
RTR_EX3#
OSPF: Send with youngest Key 10
OSPF: Rcv pkt from 10.8.1.1, Ethernet0 : Mismatch Authentication type. Input
packet specified type 0, we use type 2
OSPF: Send with youngest Key 10
OSPF: Rcv pkt from 10.8.1.1, Ethernet0 : Mismatch Authentication type. Input
packet specified type 0, we use type 2
```

```
RTR_EX3#
```

4 Explain what problem is indicated by the error messages in Example 8-113.

### Example 8-113. The error messages for Troubleshooting Exercise 4

```
RTR_EX4#
OSPF: Send with youngest Key 10
OSPF: Rcv pkt from 10.8.1.1, Ethernet0 : Mismatch Authentication Key - Messag
igest Key 10
OSPF: Send with youngest Key 10
OSPF: Rcv pkt from 10.8.1.1, Ethernet0 : Mismatch Authentication Key - Messag
igest Key 10
RTR_EX4#
```

5 Explain what problem is indicated by the error messages in Example 8-114.

### Example 8-114. The error messages for Troubleshooting Exercise 5

```
RTR_EX5#
%OSPF-4-ERRRCV: Received invalid packet: mismatch area ID, from backbone area
be virtual-link
but not found from 10.8.1.1, Ethernet0
%OSPF-4-ERRRCV: Received invalid packet: mismatch area ID, from backbone area
be virtual-link
but not found from 10.8.1.1, Ethernet0
RTR_EX5#
```

6 The configurations for the routers in Figure 8-58 follow.

A:

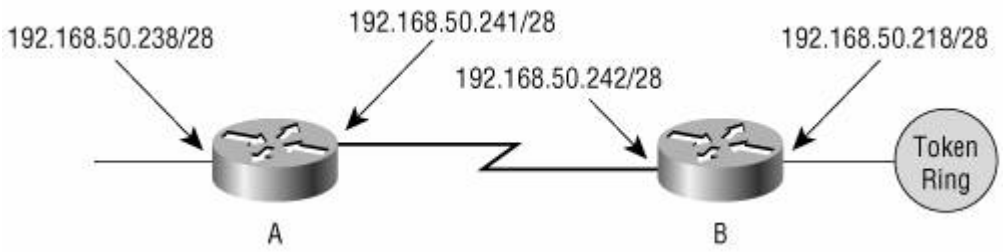
```
router ospf 15
network 192.168.50.224 0.0.0.31 area 192.168.50.0
network 192.168.50.240 0.0.0.15 area 0.0.0.0
area 192.168.50.0 authentication message-digest
```

B:

```
router ospf 51
network 192.168.50.0 0.0.0.255 area 0
```

Routers A and B are not forming an adjacency. What is wrong?

### Figure 8-58. The network for Troubleshooting Exercise 6.




7 Example 8-115 shows a link-state database from an area in which an unstable link exists. B on the information shown, which link is the likely culprit?

Example 8-115. The link-state database for Troubleshooting Exercis

```
RTR_EX7#show ip ospf database

OSPF Router with ID (10.8.20.1) (Process ID 1)

Router Link States (Area 0)

Link ID     ADV Router    Age    Seq#          Checksum    Link count
10.3.0.1    10.3.0.1      18     0x8000001B    0x6AF8      5
10.8.5.1    10.8.5.1      15     0x80000267    0xFDA0      6
10.8.20.1   10.8.20.1     478    0x8000001E   0xD451      4

Net Link States (Area 0)

Link ID     ADV Router    Age    Seq#          Checksum
10.8.1.2    10.3.0.1      18     0x80000013    0xA747
RTR_EX2#
```

PREV NEXT

# Chapter 9. OSPFv3

This chapter covers the following subjects:

- Operation of OSPFv3
- Configuring OSPFv3
- Troubleshooting OSPFv3

As you have seen in previous chapters, routing IPv6 requires modifications to a protocol; primarily, the protocol messages must be modified to carry addresses four times as long as IPv4 addresses. In theory, this also could be done with Open Shortest Path First (OSPF), either by modifying the existing Link State Advertisements (LSA) or by defining new LSAs. But development of OSPF began in the very late 1980s, when router performance was low, latency was high, and memory was expensive. None of these are valid now, and several characteristics of OSPFv2 that were intended to accommodate or compensate for those early networking realities are now irrelevant. Further, extensive operational experience with OSPFv2 revealed several areas of inefficiency.

So when extension of OSPF to support IPv6 was first considered, it was recognized that there was an opportunity to improve the protocol itself. The result is that rather than just extending OSPFv2 for IPv6, a new and improved version of OSPFOSPF version 3has been created.

# Operation of OSPFv3

OSPFv3 is specified in RFC 2740. There are some high-level similarities between the relationships of RIPng to RIPv2 and OSPFv3 to OSPFv2. Most important, OSPFv3 uses the same fundamental mechanisms as OSPFv2the SPF algorithm, flooding, DR election, areas, and so on. Constants and variables such as timers and metrics are also the same.

Another similarity to the relationship of RIPng to RIPv2 is that OSPFv3 is not backward-compatible with OSPFv2. So if you want to use OSPF to route both IPv4 and IPv6, you must run both OSPFv2 and OSPFv3. As this chapter is being written, there is discussion of adding IPv4 support to OSPFv3, but no specifications have yet been completed. I, for one, hope this support is added, because OSPFv3 is a significantly improved protocol.

It is assumed that you have read the previous chapter and understand the operation of OSPFv2. This section does not repeat that information, but presents the significant differencesprimarily operational and LSA formatsof OSPFv3.

## OSPFv3 Differences from OSPFv2

In addition to the changes in LSAs, described in the next section, there are several changes to OSPF procedures themselves. This section describes the most important of those changes:

- Per link protocol processing You have already seen that an interface to a link can have more than one IPv6 address. In fact, a single link can belong to multiple subnets, and two interfaces attached to the same link but belonging to different IPv6 subnets can still communicate. OSPFv3 changes the OSPFv2 language of "subnet" to "link," and allows the exchange of packets between two neighbors on the same link but belonging to different IPv6 subnets.

- Removal of addressing semantics As you will see in the subsequent sections, OSPFv3 Router and Network LSAs do not carry IP addresses. A new LSA is defined for that purpose. This has some scaling advantages. However, 32-bit RIDs, AIDs, and LSA IDs are maintained in IPv6. Keep in mind that although these IDs are written in dotted decimal, and are often derived in OSPFv2 networks from working IPv4 addresses, they are not IPv4 addresses. These OSPFv3 IDs are still expressed in dotted decimal, allowing easy overlay of an OSPFv3 network on an existing OSPFv2 network.

- Neighbors are always identified by Router ID OSPFv2 neighbors on broadcast and NBMA links are identified by their interface addresses, whereas neighbors on other types of links are identified by RID. OSPFv3 removes this inconsistency, so that all neighbors on all link types are identified by RID.

- Addition of a link-local flooding scope OSPFv3 retains the domain (or AS) and area flooding scopes of OSPFv2, but adds a link-local flooding scope. You already knowparticularly from Chapter 2, "IPv6 Overview"that IPv6 makes much use of the link-local scope. As you will see in the subsequent sections, a new LSAthe Link LSAhas been added, for carrying information that is only relevant to neighbors on a single link. This LSA has link-local flooding scope, meaning it cannot be flooded beyond any attached router.

- Use of link-local addresses You already know that OSPFv2 packets have a link-local scope; they are not forwarded by any router. OSPFv3 uses a router's link-local IPv6 address (recall from Chapter 2 that these addresses always begin with FF80::/10) as the source address, and as next-hop addresses.

- Support for multiple instances per link There are applications in which multiple OSPF routers can be attached to a single broadcast link but should not form a single adjacency among them. An example of this is a shared Network Access Point (NAP). For example, suppose four routers are attached to an Ethernet link. Routers 1 and 2 belong to one OSPF domain, and routers 3 and 4 belong to a different OSPF domain. There should be adjacencies between 1 and 2 and between 3 and 4, but not, for instance, between 1 and 3. You can accomplish this kind of separation of adjacencies with OSPFv2 by manipulating authentication, but it is not ideal. Among other things, the routers belonging to one adjacency will be continuously logging authentication failures from the rejected Hellos of the other adjacency. OSPFv3 allows for multiple instances per link by adding an Instance ID to the OSPF packet header to distinguish instances. An interface assigned to a given Instance ID will drop OSPF packets whose Instance ID does not match.

- Removal of OSPF-specific authentication IPv6 has, using the Authentication extension header, a standard authentication procedure. Because of this, OSPFv3 has no need for its own authentication of OSPFv3 packets; it just uses IPv6 authentication.

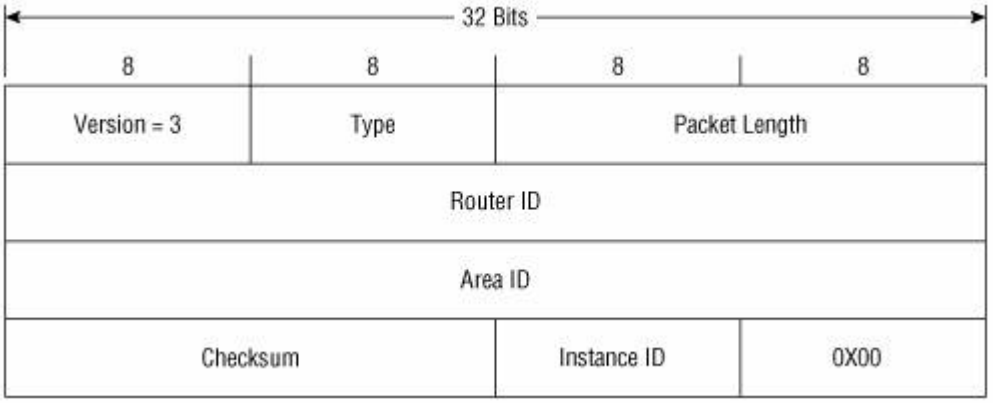
- More flexible handling of unknown LSA types Where OSPFv2 always discards unknown LSA types, OSPFv3 can either treat them as having link-local flooding scope, or store and flood them as if they are understood, while ignoring them in their own SPF algorithms. This can result in easier network changes and easier integration of new capabilities in OSPFv3 than in OSPFv2.

## OSPFv3 Messages

OSPFv2 and OSPFv3 both have the same protocol number of 89, although OSPFv3, being an IPv6 protocol, more accurately has a *Next Header* value of 89. And like OSPFv2, OSPFv3 uses multicast whenever possible. The IPv6 AllSPFRouters multicast address is FF02::5, and the AllDRouters multicast address is FF02::6. Both have link-local scope. You can easily see the similarity in the last bits with the OSPFv2 addresses of 224.0.0.5 and 224.0.0.6.

OSPFv3 uses the same five message typesHello, DD, LS Database Request, LS Database Update, and LS Acknowledgmentas OSPFv2, and numbers them the same. The message header, shown in Figure 9-1, is somewhat different from the OSPFv2 message header. The version number, of course, is 3 rather than 2. But more important, there are no fields for authentication. As discussed in the previous section, OSPFv3 uses the Authentication extension header of the IPv6 packet itself rather than its own authentication process. And there is an Instance ID, which allows multiple OSPFv3 instances to run on the same link. The Instance ID has local link significance only, because the OSPFv3 message is not forwarded beyond the link on which it is originated.

Figure 9-1. OSPFv3 packet header.



Aside from the message header, the format of only two of the OSPFv3 messages is different from their OSPFv2 counterparts: Hellos and Database Description messages.

Figure 9-2 shows the format of the OSPFv3 Hello message. Unlike OSPFv2, there is no Network Mask field because IPv6 has no need for it. Beyond that, the same fields (below the header) appear in both packets. The Options field increases in size to 24 bits, and the Router Dead Interval is decreased from 32 bits to 16 bits. The implication of this second field is that the theoretical maximum Router Dead interval is decreased from 4.3 billion seconds to 65,535 seconds. This change has little or no bearing on operational networks. The maximum configurable Router Dead interval on the most common OSPF implementations has long been 65,535 anyway, and intelligent OSPF designs do not approach even this maximum; so the change in the field size is only for conservation of unused space.

Figure 9-2. OSPFv3 Hello message.

[View full size image]

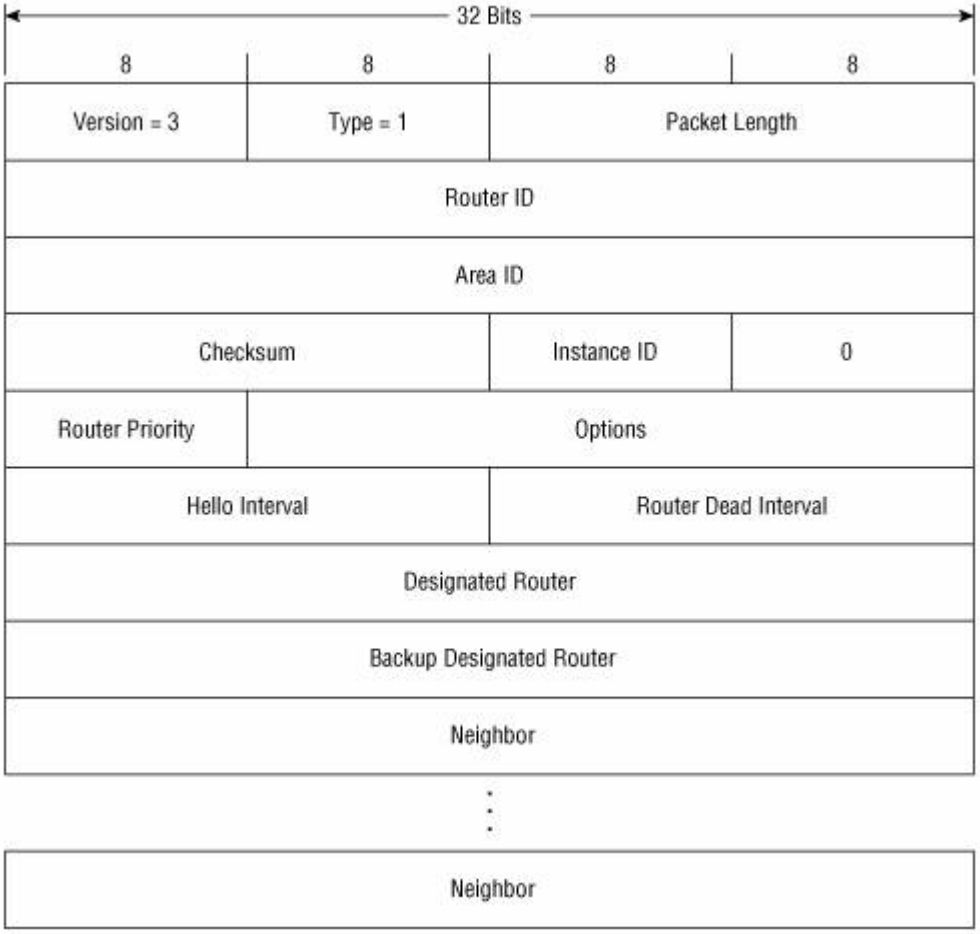
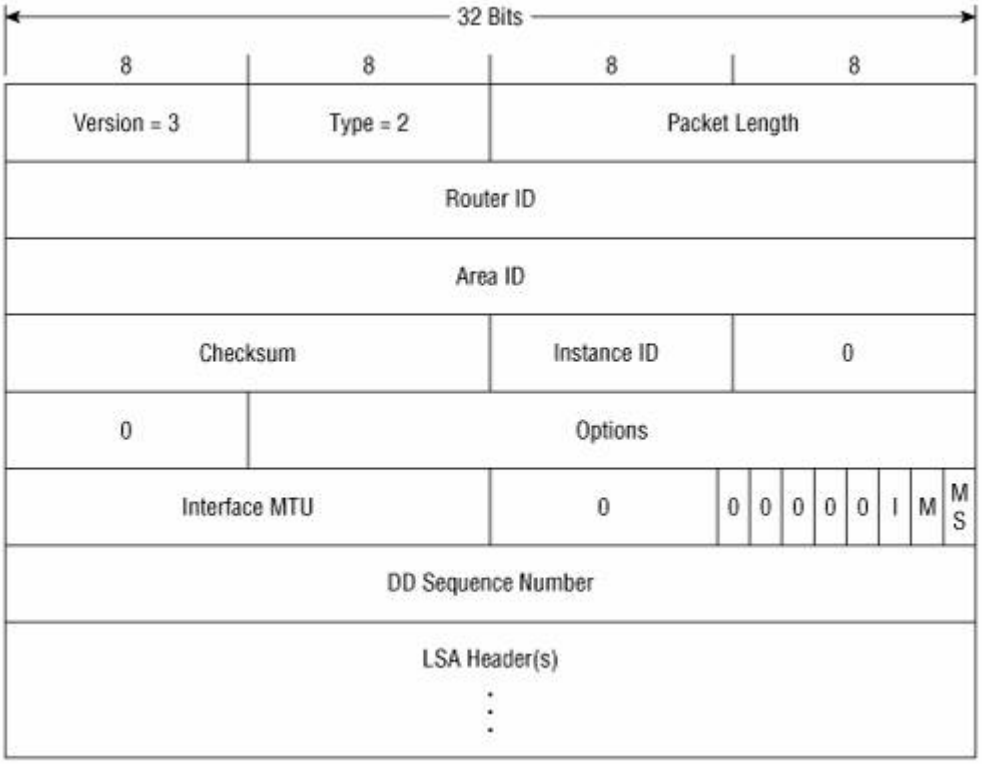



Figure 9-3 shows the format of the OSPFv3 Database Description packet. It differs from its OSPFv2 counterpart only in the larger Options field.

Figure 9-3. OSPFv3 Database Description message.

[View full size image]



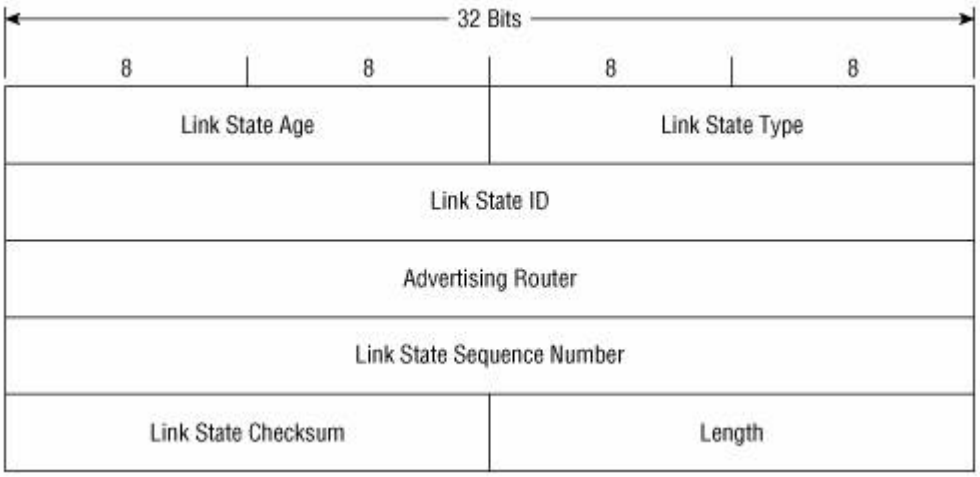

The other three message typesLink State Request, Link State Update, and Link State Acknowledgmenthave formats that do not differ from those of OSPFv2, and so are not shown in this chapter.

## An Overview of OSPFv3 LSAs

The OSPFv3 LSA header is shown in Figure 9-4. Comparing it with the OSPFv2 LSA header in Figure 8-54, you can see that it is almost identical, except that there is no Options field, and the Link State Type field is 16 bits rather than the 8-bit Type field of OSPFv2.

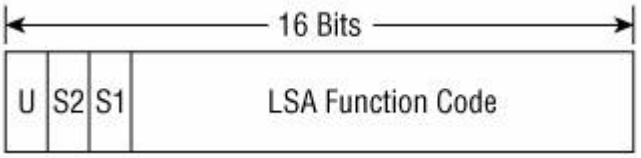
Figure 9-4. OSPFv3 LSA header.

[View full size image]



The reason that the Link State Type field is longer is that it includes three preceding bits, as shown in Figure 9-5.

Figure 9-5. Link State Type field of the OSPFv3 LSA header.




*U* indicates how a router should handle the LSA if it does not recognize the LSA's Function Code. If the bit is cleared, the unknown LSA is to be treated as if it had link-local flooding scope. If the U bit is set, the unknown LSA is to be stored and flooded as if it is understood.

*S2* and *S1* indicate the LSA's flooding scope. Table 9-1 shows the possible values of these two bits and the associated flooding scopes.

Table 9-1. S bits in the OSPFv3 LSA Link State Type field and their associated flooding scopes.

| S2 | S1 | Flooding Scope |
|---|---|---|
| 0 | 0 | Link-Local |
| 0 | 1 | Area |
| 1 | 0 | AS (Routing Domain) |
| 1 | 1 | Reserved |

LSA Function Code, the last 13 bits of the LS Type field, corresponds to the OSPFv2 Type field. Table 9-2 shows the common LSA types used by OSPFv3 and the values of their corresponding LS Types. If you decode the hex values, you will see that the default U bit of all of them is 0. The S bits of all LSAs except two indicate area scope. Of the remaining two, AS External LSAs have an AS flooding scope and Link LSAs have a link-local flooding scope. Most of the OSPFv3 LSAs have functional counterparts in OSPFv2; these OSPFv2 LSAs and their types are also shown in Table 9-2.

Table 9-2. OSPFv3 LSA types and their OSPFv2 counterparts.

| OSPFv3 LSAs | | OSPFv2 LSAs | |
|---|---|---|---|
| LS Type | Name | Type | Name |
| 0x2001 | Router LSA | 1 | Router LSA |
| 0x2002 | Network LSA | 2 | Network LSA |
| 0x2003 | Inter-Area Prefix LSA | 3 | Network Summary LSA |
| 0x2004 | Inter-Area Router LSA | 4 | ASBR Summary LSA |
| 0x4005 | AS-External LSA | 5 | AS-External LSA |
| 0x2006 | Group Membership LSA | 6 | Group Membership LSA |
| 0x2007 | Type-7 LSA | 7 | NSSA External LSA |
| 0x0008 | Link LSA | | *No Corresponding LSA* |
| 0x2009 | Intra-Area Prefix LSA | | *No Corresponding LSA* |

Although Router and Network LSAs have the same names, there is a significant difference in how the OSPFv3 and OSPFv2 Router and Network LSAs function. Specifically, OSPFv3 Router and Network LSAs do not advertise prefixes. This is an important improvement in the scaling of the protocol. These LSAs are, as you know from Chapter 8, "OSPFv2," primarily to represent the router as a node on the SPF tree. So when a Router or Network LSA is flooded, there is an assumption that a topological change has taken place and all routers in the area, on receipt of the LSA, rerun SPF. But because OSPFv2 routers also use these LSAs to advertise their connected subnets, if a subnet changes, the LSA must also be flooded to advertise the change. Even though something like an address change does not affect the SPF topology, the reception of a Router or Network LSA triggers an SPF run anyway. This can be particularly problematic for edge or access routers that have many stub links that change regularly.

OSPFv3 removes the prefix advertisement function from Router and Network LSAs, and puts it in the new Intra-Area Prefix LSA. Now Router and Network LSAs only represent the router's node information for SPF and are only flooded if information pertinent to the SPF algorithm changes. If a prefix changes, or the state of a stub link changes, that information is flooded in an Intra-Area Prefix LSA that does not trigger an SPF run.

Another difference between Router and Network LSAs between OSPFv2 and OSPFv3 is in the exchange of some information that is only relevant to directly connected neighbors. OSPFv2 puts this information in Router or Network LSAs; and although only the directly connected neighbors care about the information, it is flooded with the LSAs throughout the area. OSPFv3 takes this neighbor-specific information and puts it in the new Link LSA, which has link-local flooding scope. Although minor, the introduction of the Link LSA does add an efficiency improvement over OSPFv2.

Inter-Area Prefix, Inter-Area Router, and Type-7 LSAs, although their names have changed, have the same function as OSPFv2 Network Summary, ASBR Summary, and NSSA LSAs, respectively. AS-External and Group Membership LSAs have both the same names and the same functions in both OSPFv2 and OSPFv3.

## OSPFv3 LSA Formats

This section details the formats of the first five OSPFv3 LSA types, and the two new LSA types

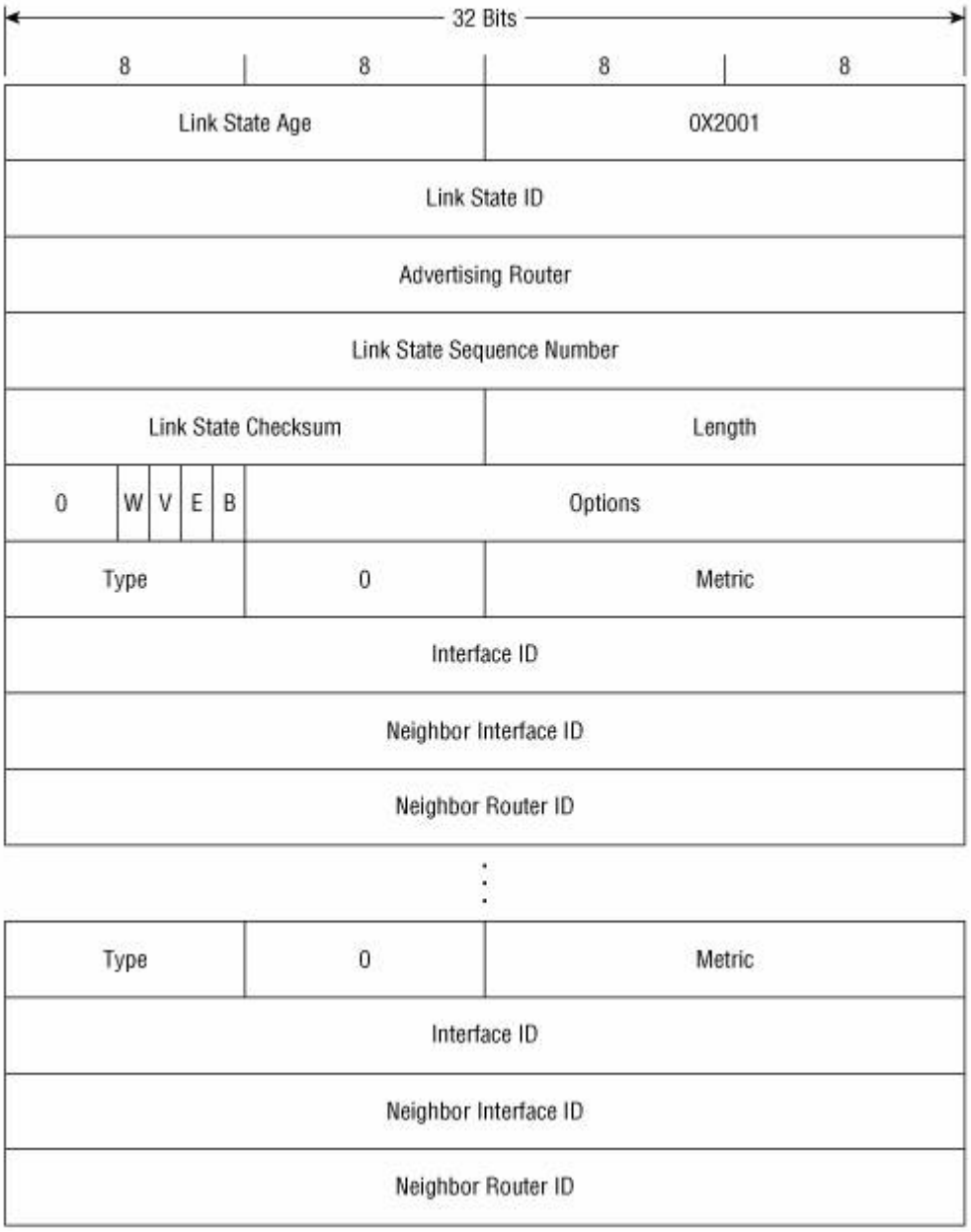
shown in Table 9-2. Although there is a specified Group Membership LSA for Multicast OSPF (MOSPF), that protocol is not covered in this book, and so the LSA is also not detailed. In most cases, only fields that differ from their OSPFv2 counterparts are discussed; you are encouraged to compare those LSAs that have corresponding OSPFv2 LSAs with the figures in Chapter 8.

## The Router LSA

Figure 9-6 shows the format of the OSPFv3 Router LSA. As discussed in the previous section, prefix information is not included in the Router LSA as it is in its OSPFv2 counterpart. The OSPFv3 Router LSA only describes the originating router and its links to neighbors, for use in SPF calculations. Prefix information is carried in the Intra-Area Prefix LSA.

Figure 9-6. OSPFv3 Router LSA.

[View full size image]



Also notice that the ToS metric fields, obsolete already in OSPFv2, are not included in this LSA.

*Options*, although in a different position within the LSA format and a longer length (24 bits) than the OSPFv2 Options field, performs the same function of identifying optional capabilities. The Options field, carried in several packets and LSAs, is described separately in a later section.

Following the Options field is a set of fields that can appear multiple times, to describe each of the attached interfaces:

- Type specifies the interface type. Table 9-3 lists the possible interface type values.

Table 9-3. Interface types specified in the Type field of the Router LSA.

| Type | Description |
|---|---|
| 1 | Point-to-point connection to another router |
| 2 | Connection to a transit network |
| 3 | Reserved |
| 4 | Virtual link |

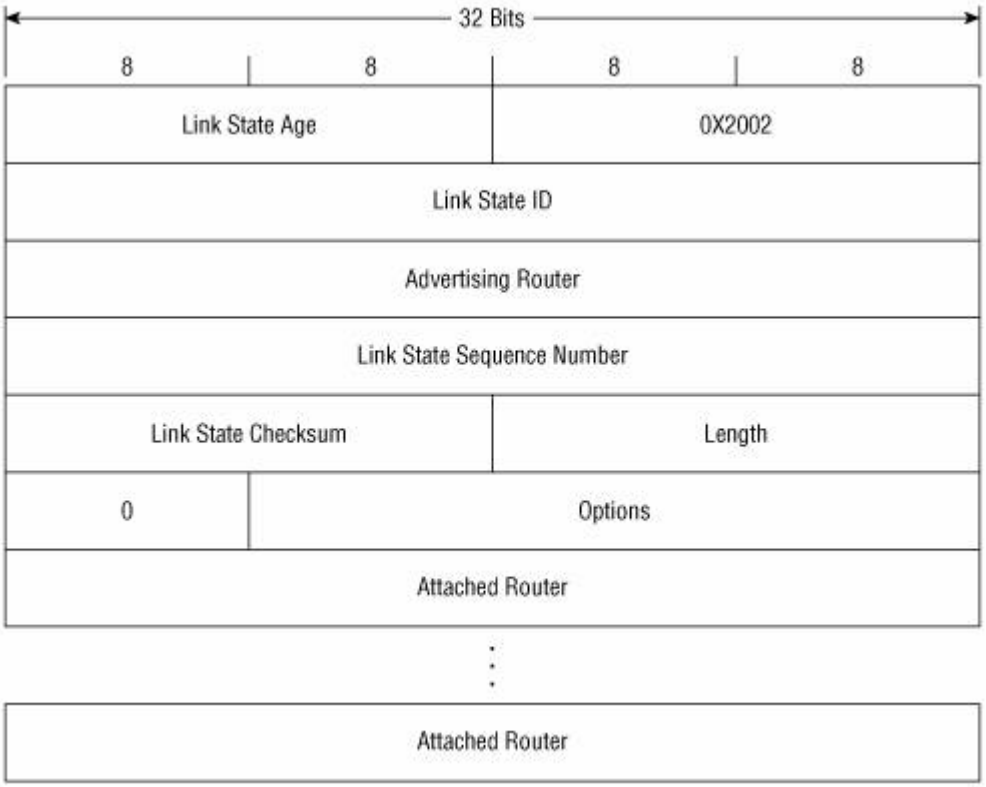
- Metric specifies the outbound cost of the interface.
- Interface ID is a 32-bit value distinguishing the interface from other interfaces on the originating router.
- Neighbor Interface ID is the Interface ID advertised by neighbors on the link in their Hellos or, in the case of type 2 links, the Interface ID of the link's DR.
- Neighbor Router ID is the neighbor's RID or, in the case of type 2 links, the DR's RID.

## Network LSA

The OSPFv3 Network LSA is shown in Figure 9-7. Functionally it is identical to the OSPFv2 Network LSA (originated by the DR to represent a pseudonode, 0 cost to attached neighbors is assumed, and so on). The only significant differences are the location of the Options field and the elimination of the Network Mask field, which has no meaning in IPv6.

Figure 9-7. OSPFv3 Network LSA.

[View full size image]

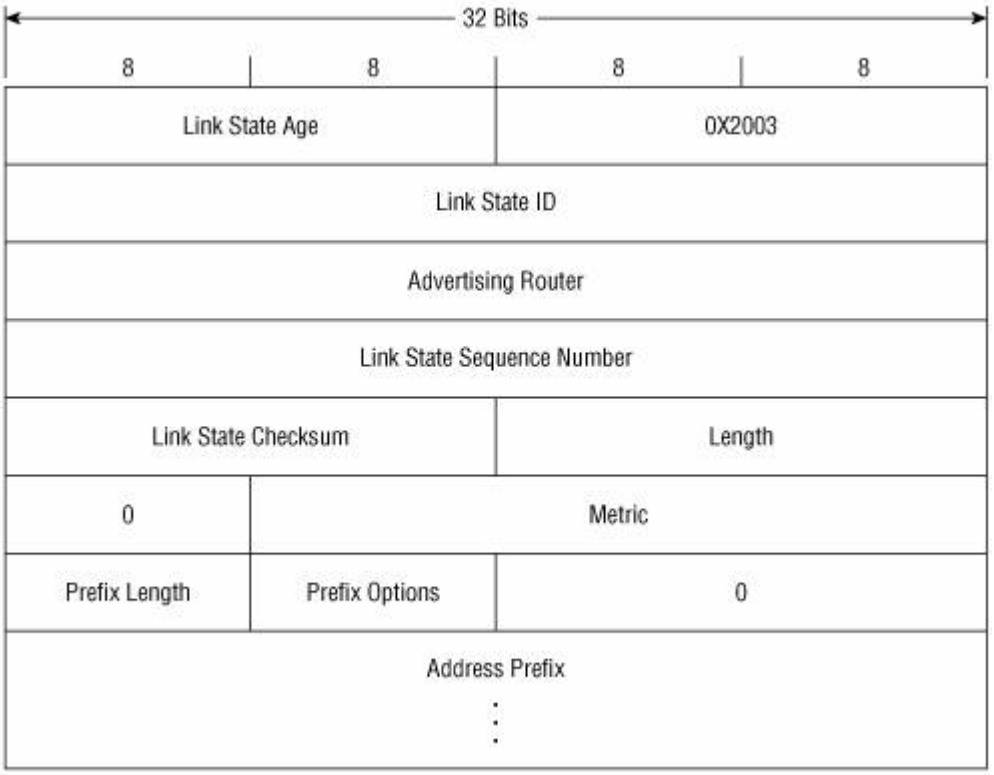



## Inter-Area Prefix LSA

The Inter-Area Prefix LSA (Figure 9-8) performs the same function as the OSPFv2 type 3 Summary LSAABRs originate them into an area to advertise networks that are outside the area but inside the OSPF domain. Only the name, more descriptive in OSPFv3, has changed. An ABR originates a separate Inter-Area Prefix LSA for each IPv6 prefix that must be advertised into an area. An ABR can also originate an Inter-Area Prefix LSA to advertise a default route into a stub area.

Figure 9-8. OSPFv3 Inter-Area Prefix LSA.

[View full size image]




## Inter-Area Router LSA

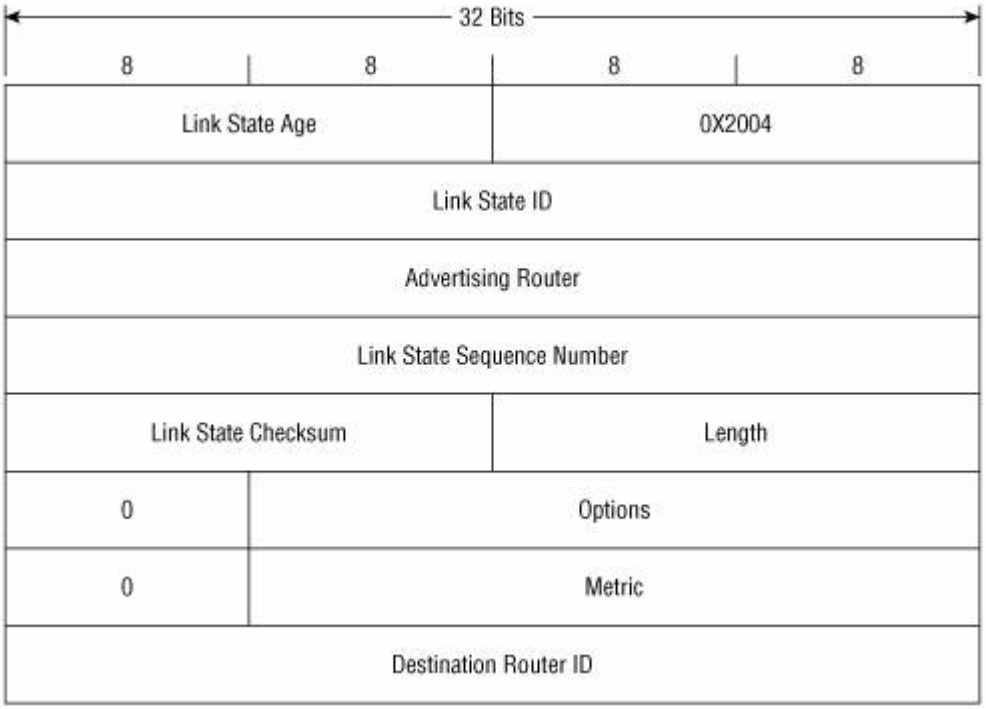
The Inter-Area Router LSA performs the same duties for OSPFv3 as the type 4 Summary LSA performs for OSPFv2. An ABR originates an Inter-Area Router LSA into an area to advertise an ASBR that resides outside of the area. The ABR originates a separate Inter-Area Router LSA for each ASBR it advertises.

Figure 9-9 shows the format of the Inter-Area Router LSA. Even though the OSPFv2 type 3 and type 4 Summary LSAs have the same formats, you can see by comparing Figure 9-8 and Figure 9-9 that the Inter-Area Network and Inter-Area Router LSAs are different. The fields are self-descriptive:

- Options specifies optional capabilities of the ASBR.
- Metric specifies the cost to the ASBR.
- Destination Router ID is the ASBR RID.

Figure 9-9. OSPFv3 Inter-Area Router LSA.

[View full size image]

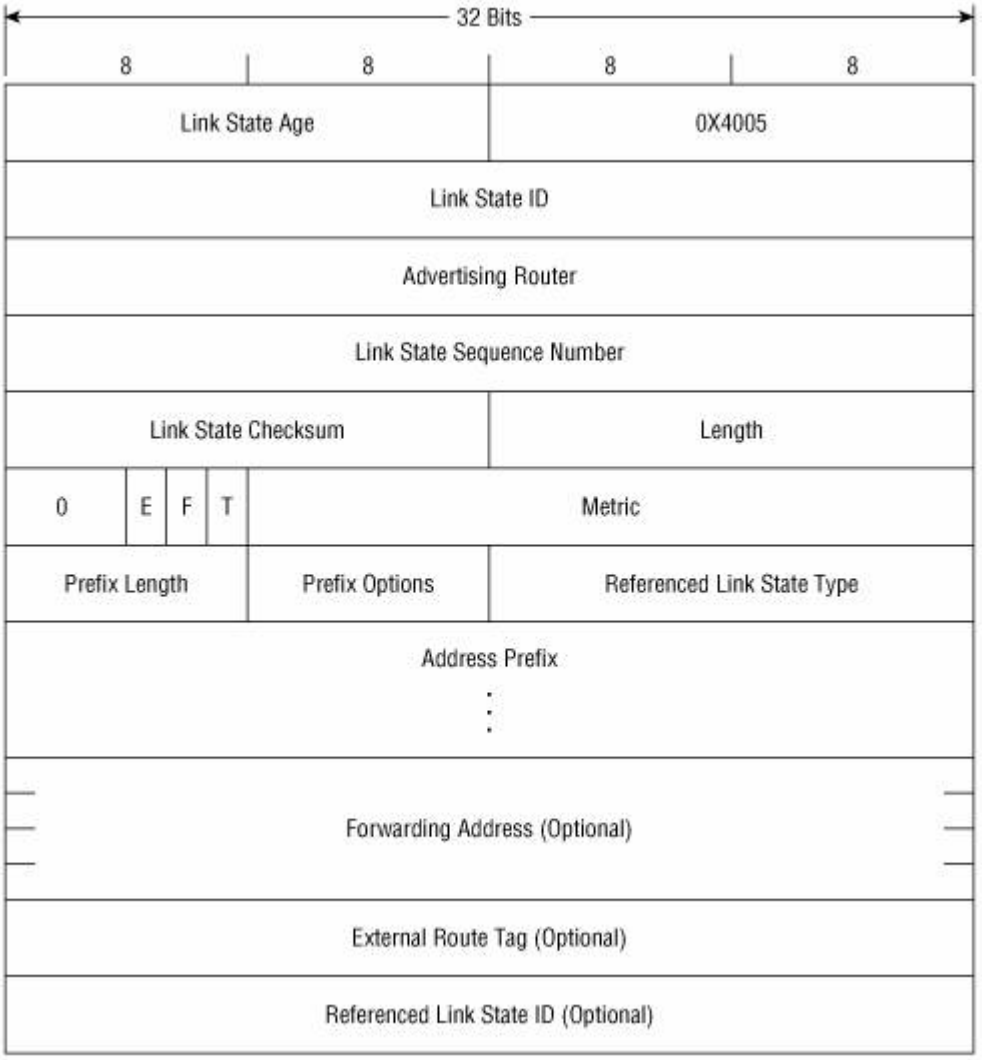



## AS-External LSA

As with OSPFv2, the AS-External LSA advertises prefixes external to the OSPF routing domain; one LSA is required for each external prefix advertised. However, the format of the OSPFv3 As-External LSA (Figure 9-10) is different from its OSPFv2 counterpart.

Figure 9-10. OSPFv3 AS-External LSA.

[View full size image]




The *E* flag performs the same function in the OSPFv3 LSA as in the OSPFv2 LSA. If set, the metric is a type 2 external metric. If the bit is cleared, the metric is a type 1 external metric.

The *F* flag indicates, when set, that a forwarding address is included in the LSA.

The *T* flag indicates, when set, that an external route tag is included in the LSA.

*Metric*, of course, specifies the cost of the route. Whether it is type 1 or type 2 depends on the value of the E flag.

*Prefix Length*, *Prefix Options*, and *Address Prefix* completely describe the enclosed prefix.

*Forwarding Address*, if included, is a fully specified 128-bit IPv6 address representing the next-hop address to the destination prefix. It is included only if the F flag is set.

*External Route Tag*, if included, performs the same function as the External Route Tag field in the OSPFv2 AS-External LSA. It is included only if the T flag is set.

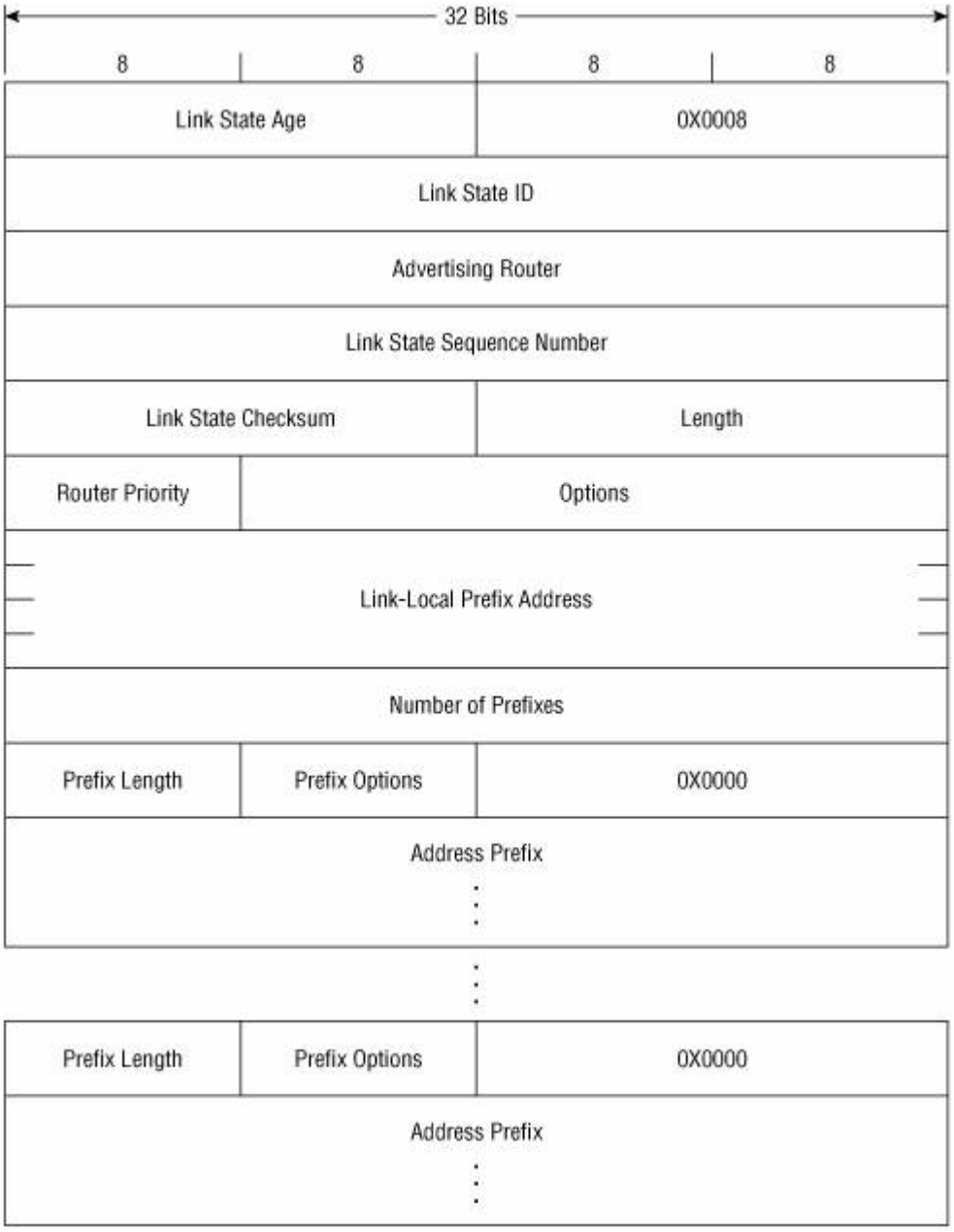
*Referenced Link State ID* and *Referenced Link State Type*, if included, allow additional information about the prefix to be included in another LSA. If used, these two fields describe the link state ID and the type of the LSA carrying the additional information. The Advertising Router field of the referenced LSA must also match the value of the Advertising Router field in the AS-External LSA. The additional information has, as with the External Route Tag, no relevance to OSPFv3 itself, but is used to communicate information across the OSPF domain between border routers. If this function is not used, the Referenced Link State Type field is set to all zeroes.

## Link LSA

The Link LSA is used for communicating information that is significant only to two directly connected neighbors. The format of the Link LSA is shown in Figure 9-11. A separate Link LSA is originated on each of a router's attached links belonging to an OSPFv3 domain, and a receiving router, because of the link-local flooding scope, never forwards the LSA to any other link.

Figure 9-11. OSPFv3 Link LSA.

[View full size image]




The LSA performs three functions:

- It provides the originating router's link-local address to all other routers attached to the link.

- It provides a list of IPv6 prefixes associated with the link.
- It provides a set of Option bits to associate with Network LSAs originated on the link.

*Router Priority* specifies the router priority assigned to the interface of the originating router.

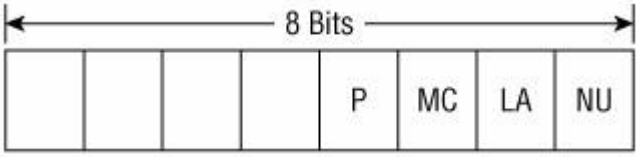
*Options* specifies the options bits that the originating router would like to set in the Network LSA that will be originated for the link. This is the same 24-bit Options field carried in OSPFv3 Hello and DD packets, in a number of OSPFv3 LSAs. The Options field is detailed in the later section, "The Options Field."

*Link-Local Prefix Address* specifies the 128-bit link-local prefix of the originating router's interface attached to the link.

*Number of Prefixes* specifies the number of IPv6 prefixes contained in the LSA, as described by the following Prefix Length, Prefix Options, and Address Prefix fields.

*Prefix Length*, *Prefix Options*, and *Address Prefix* describe one or more IPv6 prefixes associated with the link by the originating router. This set of fields is used not only in the Link LSA, but also the Intra-Area-Prefix, Inter-Area-Prefix, and AS-External LSAs. The advertised prefix can be any length between 0 and 128. When the prefix is not an even multiple of 32 bits, it is padded out with zeroes to fit the 32-bit boundaries of the Address Prefix field. The Prefix Length field specifies the length of the unpadded prefix, in bits. The Prefix Options field, shown in Figure 9-12, specifies optional handling of the prefix during routing calculations.

Figure 9-12. Prefix Options field.




The *Propagate* (P) bit is set on NSSA area prefixes that should be re-advertised at the NSSA area border.

The *Multicast* (MC) bit, when set, specifies that the prefix should be included in multicast routing calculations.

The *Local Address* (LA) bit, when set, specifies that the prefix is an interface address of the advertising router.

The *No Unicast* (NU) bit, when set, specifies that the prefix should be excluded from unicast route calculations.

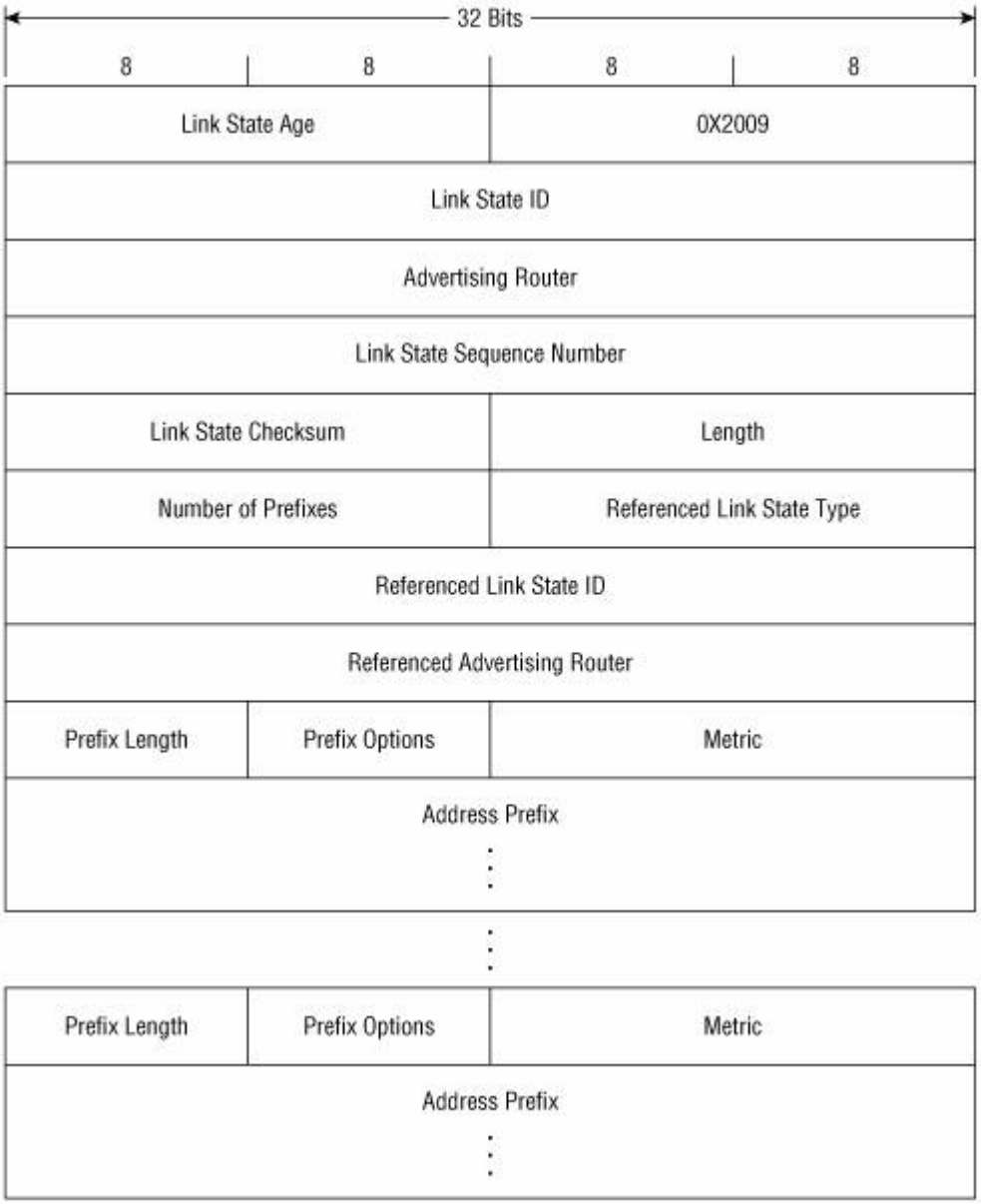
## Intra-Area Prefix LSA

The Intra-Area Prefix LSA is shown in Figure 9-13. Recall from the previous discussion of this LSA that attached prefixes, which in OSPFv2 are carried in Router and Network LSAs, in OSPFv3 are carried in Intra-Area Prefix LSAs. Prefix changesincluding additions and deletionsdo not influence the branches of the SPF tree in any way. But OSPFv2 advertises prefixes in Router and Network LSAs, causing an SPF calculation in all routers in the area whenever a prefix change occurs. With OSPFv3, however, when a link or its prefix changes, the connected router originates an Intra-Area Prefix LSA to flood the information throughout the area. This LSA does not trigger an SPF calculation; the receiving routers simply associate the new prefix information with the

originating router. Router and Network LSAs, in OSPFv3, serve only to provide topological information. As a result, this new LSA should make OSPFv3 significantly more scalable for networks with large numbers of frequently changing prefixes.

Figure 9-13. Intra-Area Prefix LSA.

[View full size image]




- Number of Prefixes specifies the number of prefixes contained in the LSA.
- Referenced Link State Type, Referenced Link State ID, and Referenced Advertising Router identify the Router or Network LSA with which the contained prefixes should be associated.

If the prefixes are associated with a Router LSA, the Referenced Link State Type is 1, the Referenced Link State ID is 0, and the Referenced Advertising Router is the RID of the originating router. If the prefixes should be associated with a network LSA, the Referenced Link State Type is 2, the Referenced Link State ID is the interface ID[1] of the link's DR, and the

Referenced Advertising Router is the RID of the DR.

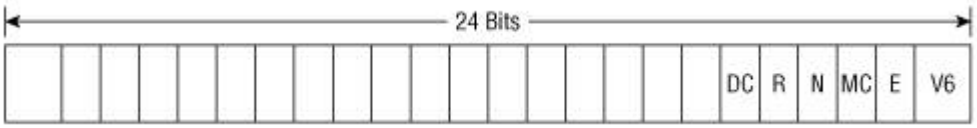
[1] The interface ID used in this and other OSPFv3 LSAs should not be confused with the 64-bit Interface ID portion of an IPv6 address. This field is a 32-bit value distinguishing this interface from other interfaces on the originating router. RFC 2740 suggests using the MIB-II If Index for the interface ID value.

Each prefix is then represented by a Prefix Length, Prefix Options, and Address Prefix field, as described previously for the Link LSA. Added to these three fields is the *Metric* field, which is the cost of the prefix.

## Options Field

The 24-bit Options field, specifying optional capabilities of the originating router, is carried in Router, Network, Inter-Area Router, and Link LSAs. This field is also carried in the Hello and Database Description packets. Figure 9-14 shows the format of the Options field. As of this writing, only the six rightmost bits are defined as options flags, and most of those are the same flags you are familiar with from OSPFv2. OSPFv3 routers ignore unrecognized options flags.

Figure 9-14. Options field.




- DC specifies support for demand circuits capability.
- R indicates whether the originator is an active router. When cleared, routes transiting the

PREV NEXT

# Configuring OSPFv3

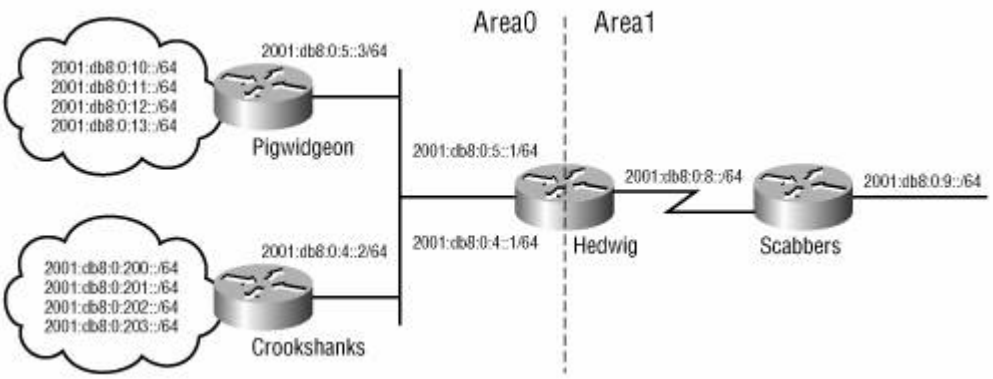
The configuration options of OSPFv3 are the same as those for OSPFv2. Process IDs and areas need to be specified. Interfaces and addresses need to be included in the process. Areas can be defined as stubs, totally stubby or not so stubby area (NSSA). Prefixes can be summarized between areas. OSPFv3 can be configured over demand circuits and on NBMA networks. Much of the configuration for OSPFv3 is the same as for OSPFv2. An IPv6 keyword is added in some cases, and an IPv6 prefix or address is used in place of the IPv4 subnet or address. This section contains case studies that show the OSPFv3 configurations that are different from OSPFv2 and some that are very similar, but need to be emphasized.

## Case Study: A Basic OSPFv3 Configuration

The configuration of OSPFv3 is similar to OSPFv2 with two exceptions. Recall from Chapter 8 that to configure OSPFv2 on a router and an interface, you first create the OSPF process using the `router ospf` command. Then, you specify address ranges that encompass interface addresses that are to participate in OSPF, using the `network area` command. The interfaces configured with IPv4 addresses that fall within the address range specified with the `network area` command participate in OSPFv2. If an interface has an IPv4 address that is not part of the address range, that interface, or that address (if there is more then one IPv4 address on an interface) will not participate in the OSPFv2 process. OSPFv3 for IPv6 is enabled by specifying an OSPF process ID and an area at the interface configuration level. If an OSPFv3 process has not yet been created, it is created automatically. All IPv6 addresses configured on the interface are included in the specified OSPF process. Figure 9-15 displays an OSPFv3 network.

Figure 9-15. Example of an OSPFv3 network.

[View full size image]




Hedwig's and Pigwidgeon's configurations are displayed in Example 9-1 and Example 9-2.

Example 9-1. OSPFv3 is configured on Hedwig with the interface command *ipv6 ospf 1 area 1*.

```
interface Serial 0/0
 ipv6 address 2001:db8:0:8::1/64
```

```
 ipv6 ospf 1 area 1
interface Ethernet0/0
 ipv6 address 2001:db8:0:4::1/64
 ipv6 ospf 1 area 0
```

### Example 9-2. Pigwidgeon is configured for OSPFv3.

```
interface Ethernet 0/0
 ipv6 address 2001:db8:0:5::3/64
 ipv6 ospf 1 area 0
interface Serial 0/0
 ipv6 address 2001:db8:0:10::1/64
 ipv6 ospf 1 area 0
```

The command ipv6 ospf area enables OSPFv3 on Serial0/0 and Ethernet0/0, places Hedwig's serial interface in area 1 and the Ethernet interface in area 0, and creates the OSPFv3 process with ID '1' on the router, as shown in Example 9-3. With OSPFv2, two commands are required to accomplish the same tasks: the router ospf 1 command creates the OSPF process, then the network area command enables OSPFv2 on interfaces. One thing to note, though, is that the network area command can enable OSPFv2 on multiple interfaces with the single command, whereas the ipv6 ospf area command has to be configured on each interface that will be running OSPFv3.

### Example 9-3. *show ipv6 protocol* displays the OSPF for IPv6 process ID and the interfaces configured in each area on the router.

```
Hedwig#show ipv6 protocol
IPv6 Routing Protocol is "connected"
IPv6 Routing Protocol is "static"
IPv6 Routing Protocol is "ospf 1"
  Interfaces (Area 0):
    Ethernet0/0
  Interfaces (Area 1):
    Serial0/0
  Redistribution:
    None
Hedwig#
```

All IPv6 addresses on an interface are included in the OSPF process that is created on the interface. For example, a second IPv6 address is added to Hedwig's Ethernet port in Example 9-4.

### Example 9-4. A second IPv6 address is added to Hedwig's Ethernet0/0. Both IPv6 addresses are included in the OSPFv3 process because OSPFv3 is configured on that interface.

```
interface Ethernet0/0
 ipv6 address 2001:db8:0:4::1/64
 ipv6 address 2001:db8:0:5::1/64
 ipv6 ospf 1 area 0
```

This second address is included automatically in the OSPFv3 process that is configured on the interface. No additional OSPFv3 commands need to be entered to make the new address part of

the routing process. IPv6 addresses on an interface cannot be selectively included in OSPFv3. Either all the addresses are included, by configuring the interface with OSPFv3, or OSPFV3 is not configured on the interface and none of the addresses are included.

OSPFv3 routers use their link-local addresses as the source of hello packets. No IPv6 prefix information is contained in hello packets. Multiple IPv6 addresses can be configured on a link. None of the addresses are defined as secondary addresses, as is done with IPv4 to configure multiple addresses on a single link. Two routers will become adjacent even if no IPv6 prefix is common between the neighbors except the link-local address. This is different from OSPFv2 for IPv4. OSPFv2 neighbors will only become adjacent if the neighbors belong to the same IP subnet, and the common subnet is configured as the primary IP address on the neighboring interfaces. In Figure 9-15, Hedwig is configured with both 2001:db8:0:4::/64 and 2001:db8:0:5::/64 on its Ethernet interface. Pigwidgeon is configured with 2001:db8:0:5::/64 and Crookshanks is configured with 2001:db8:0:4::/64 on their Ethernet interfaces. Example 9-5 shows that Crookshanks is adjacent to both Hedwig (10.1.1.1) and Pigwidgeon (10.1.3.1). Pigwidgeon is the backup designated router.

### Example 9-5. *show ipv6 ospf neighbor* shows Crookshanks's neighbors are all in the FULL state, even though they don't share a common IPv6 prefix, other then the link-local address.

```
Crookshanks#show ipv6 ospf neighbor

Neighbor ID     Pri   State           Dead Time    Interface ID    Interface
10.1.1.1          1   FULL/DROTHER    00:00:30     3               Ethernet0/0
10.1.3.1          1   FULL/BDR        00:00:37     3               Ethernet0/0
Crookshanks#
```

Other parameters must match between two neighbors before they will become adjacent. These parameters are the same as IPv4: The neighbors must share the same area id, they must have the same Hello interval and dead time, and they both must have the same value in the E-bit, indicating whether the area is a stub area or not. An OSPFv3 packet must also have the same Instance ID as the receiving interface, or the OSPFv3 packets will be dropped. Instance ID will be discussed later in this chapter, in the case study, Multiple Instances on a Link.

OSPFv3 uses a 32-bit number for a Router ID. If IPv4 is configured on the router, by default, the RID is chosen in the same way it is by OSPFv2 for IPv4. The highest IPv4 address configured on a loopback interface will become the RID, or if no loopback interfaces are configured, the highest address on any other interface will become the RID.

IPv6 neighbors are always known by their RIDs, unlike IPv4, where point-to-point network neighbors are known by RIDs and broadcast, NBMA and point-to-multipoint network neighbors are known by their interface IP addresses. The neighbor IDs shown in Example 9-5 show that the router IDs are obtained from configured IPv4 addresses.

If IPv4 is not configured in the network, and you don't want to configure IPv4 just to establish a router ID, the router ID must be configured using the IPv6 OSPF routing process command `router-id` before the OSPF process will start.

When OSPFv3 is configured on an interface, the routing process is created. Interface parameters, such as the cost of a link, are modified at the interface configuration, but global parameters are modified at the OSPF process level.

## Case Study: Stub Areas

The `ipv6 router ospf` command takes you into the global process configuration mode, just as `router ospf` does for IPv4. The same configuration customization can be done with IPv6 as with

IPv4. Stub, NSSA, and totally stubby are supported and configured in the exact same way as for OSPFv2 for IPv4, using the area stub, area nssa, and area stub no-summary commands. Area 1 in Figure 9-15 is configured as a totally stubby area with the configurations in Example 9-6 and Example 9-7 at Hedwig and Scabbers.

### Example 9-6. On Hedwig, area 1 is configured as a totally stubby area. no-summary is only configured on the ABR.

```
ipv6 router ospf 1
 area 1 stub no-summary
```

### Example 9-7. Area 1 on Scabbers is also configured as a stub because all routers in the stub area must be configured as a stub.

```
ipv6 router ospf 1
 area 1 stub
```

Example 9-8 shows Scabbers's IPv6 route table before area 1 becomes totally stubby, and Example 9-9 shows the route table of Scabbers as part of the totally stubby area.

### Example 9-8. The IPv6 route table at Scabbers contains 10 OSPF entries, all known via Hedwig.

```
Scabbers#show ipv6 route
IPv6 Routing Table - 16 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OI  2001:DB8:0:4::/64 [110/74]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
OI  2001:DB8:0:5::/64 [110/74]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
C   2001:DB8:0:8::/64 [0/0]
     via ::, Serial0/0
L   2001:DB8:0:8::2/128 [0/0]
     via ::, Serial0/0
C   2001:DB8:0:9::/64 [0/0]
     via ::, Ethernet0/0
L   2001:DB8:0:9::2/128 [0/0]
     via ::, Ethernet0/0
OI  2001:DB8:0:10::/64 [110/138]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
OI  2001:DB8:0:11::/64 [110/138]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
OI  2001:DB8:0:12::/64 [110/138]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
OI  2001:DB8:0:13::/64 [110/138]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
OI  2001:DB8:0:200::/64 [110/138]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
OI  2001:DB8:0:201::/64 [110/138]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
OI  2001:DB8:0:202::/64 [110/138]
```

```
     via FE80::202:FDFF:FE5A:E40, Serial0/0
OI  2001:DB8:0:203::/64 [110/138]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Scabbers#
```

Example 9-9. As part of a totally stubby area, Scabbers now has only a single OSPF entry, the default route.

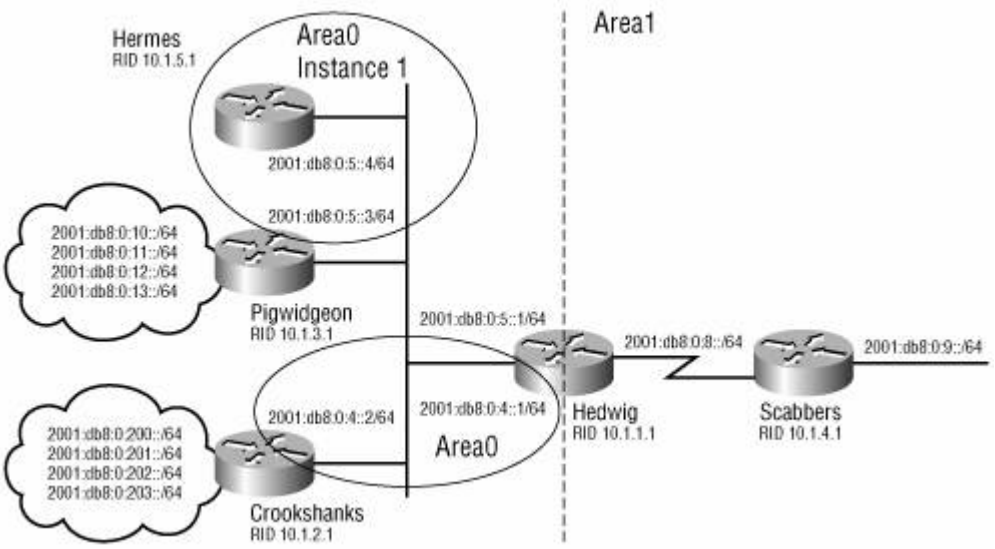
```
Scabbers#show ipv6 route
IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OI  ::/0 [110/65]
     via FE80::202:FDFF:FE5A:E40, Serial0/0
C   2001:DB8:0:8::/64 [0/0]
     via ::, Serial0/0
L   2001:DB8:0:8::2/128 [0/0]
     via ::, Serial0/0
C   2001:DB8:0:9::/64 [0/0]
     via ::, Ethernet0/0
L   2001:DB8:0:9::2/128 [0/0]
     via ::, Ethernet0/0
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Scabbers#
```

## Case Study: Multiple Instances on a Link

A new router, Hermes, has been added to the Ethernet segment in Figure 9-16. The desired OSPF design is to separate Pigwidgeon's and Hermes's OSPFv3 traffic from Hedwig's and Crookshanks's traffic. As the configuration stands, the DR and BDR are chosen among the four routers, and each router becomes adjacent with the DR and BDR. OSPFv3 Hellos contain a field for an Instance ID. The Instance ID can be used to segment the two OSPF processes running on the LAN segment. The Instance ID received in a Hello packet must match the Instance ID configured on the receiving interface or the Hello packet will be discarded. Instance ID 0 is used when none other is specified. By configuring Pigwidgeon and Hermes with a different Instance ID than is configured on Hedwig and Crookshanks, the desired adjacencies will be established.

Figure 9-16. A new router is added to the network of Figure 9-15.

[View full size image]




Pigwidgeon's configuration is modified as displayed in Example 9-10.

## Example 9-10. Pigwidgeon's Instance ID is modified from the default of 0 to create a distinct OSPF process on the Ethernet.

```
interface Ethernet 0/0
 ipv6 address 2001:db8:0:5::3/64
 ipv6 ospf 1 area 0 instance 1
```

Pigwidgeon's Instance ID is changed from the default of 0 to 1. Hedwig and Crookshanks continue to use the default Instance ID of zero. Hermes is configured similarly to Pigwidgeon to use instance ID 1. Looking at the IPv6 OSPF configuration running on Hermes's Ethernet0/0 interface, shown in Example 9-11, only Pigwidgeon (10.1.3.1) is adjacent.

Although multiple OSPFv3 processes can run on a router, only a single process or instance can run on an interface.

## Example 9-11. Hermes's IPv6 OSPF Ethernet interface configuration shows that Pigwidgeon, the only other router on the Ethernet segment using instance ID 1, is adjacent.

```
Hermes#show ipv6 ospf interface ethernet 0/0
Ethernet0/0 is up, line protocol is up
  Link Local Address FE80::206:28FF:FEB6:5BC0, Interface ID 3
  Area 0, Process ID 1, Instance ID 1, Router ID 10.1.5.1
  Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 10.1.5.1, local address FE80::206:28FF:FEB6:5BC0
  Backup Designated router (ID) 10.1.3.1, local address FE80::201:42FF:FE79:E500
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:09
  Index 1/1/1, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 4
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.1.3.1  (Backup Designated Router)
```

```
  Suppress hello for 0 neighbor(s)
Hermes#
```

## Case Study: OSPFv3 on NBMA Networks

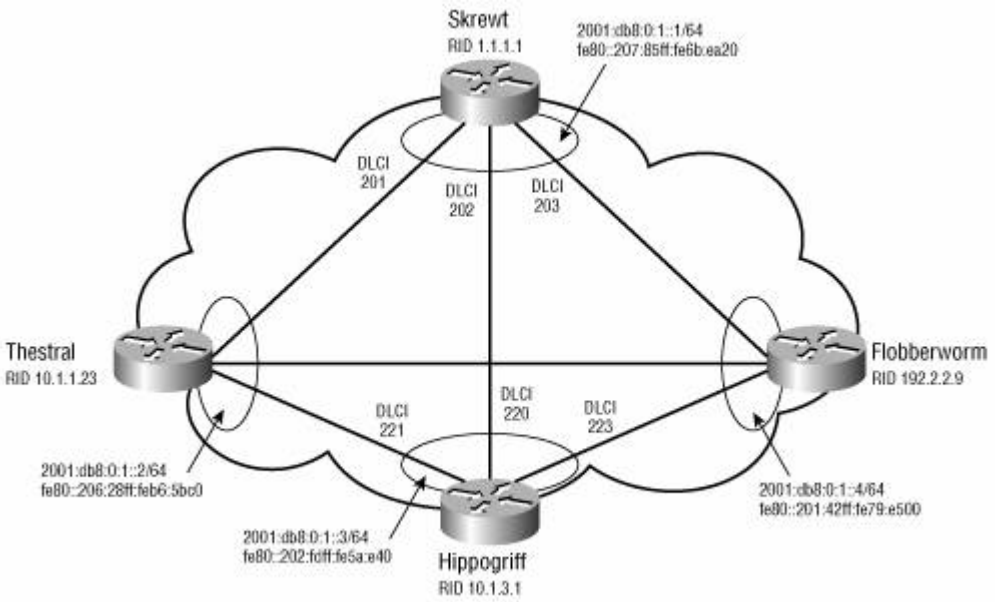
OSPFv3 on NBMA networks has the same configuration options as does OSPFv2, that is, the network, from OSPF's point of view, can remain NBMA, can be configured as broadcast or point-to-multipoint, or point-to-point networks can be configured using subinterfaces. Point-to-point links are straightforward to configure. They are configured the same way as routers connected directly via serial links. Hedwig and Scabbers, in the first case study in this chapter, called "A basic OSPFv3 configuration," are configured with directly connected serial links. If these two routers were connected with point-to-point Frame-relay subinterfaces, the OSPFv3 configuration would remain the same. The NBMA, broadcast, and point-to-multipoint networks will be discussed in this section.

Before OSPFv3 can learn about neighbors, the neighbor addresses that OSPF will use must be associated with specific virtual circuits traversing the NBMA network. Frame Relay maps are created to identify a remote IPv6 address with one of the PVCs connected to the local router.[2] Because IPv6 can have multiple addresses configured on a single interface, many map statements for each PVC might be required. All OSPFv3 packets use the Link-local address as the source address. It is also the link-local address that is used as the destination address for unicast OSPFv3 packets, and as the next-hop to which to forward a packet when routing. At a minimum, therefore, the link-local addresses must be mapped. Figure 9-17 shows a Frame Relay network with four attached routers.

[2] IPv4 uses Frame Relay inverse ARP to dynamically map addresses. IPv6's functionality to dynamically map addresses to Frame Relay PVCs is not supported in IOS at the time of this writing.

Figure 9-17. Several options exist for configuring OSPFv3 for IPv6 on this Frame Relay network.

[View full size image]




The first method of configuring the routers for OSPFv3 is to manually configure OSPFv3 neighbors. The remote routers' link-local addresses are mapped to the correct DLCI, and the IPv6 OSPFv3 neighbors are defined, both as part of the interface's configuration. Skrewt's

configuration is shown in Example 9-12.

## Example 9-12. OSPFv3 neighbors are configured on Skrewt manually.

```
interface Serial 0/0
 encapsulation frame-relay
 ipv6 address 2001:DB8:0:1::1/64
 ipv6 ospf 1 area 1
 frame-relay map ipv6 fe80::206:28ff:feb6:5bc0 201
 frame-relay map ipv6 fe80::202:fdff:fe5a:e40 202
 frame-relay map ipv6 fe80::201:42ff:fe79:e500 203
 ipv6 ospf neighbor fe80::206:28ff:feb6:5bc0 priority 5
 ipv6 ospf neighbor fe80::202:fdff:fe5a:e40 priority 10
 ipv6 ospf neighbor fe80::201:42ff:fe79:e500
```

Notice that the Link-local addresses are used in the frame-relay map statements and in the ospf neighbor statements. To specify which routers are to become the DR and BDR, assign priorities using an optional keyword with ipv6 ospf neighbor command.

Another configuration option is to enable OSPFv3 to dynamically discover neighbors. Two configuration items are required: The OSPF network gets defined as a broadcast network using the command ipv6 ospf network broadcast, and the frame-relay map statements are configured to forward broadcasts over the PVC using the broadcast keyword on the map statement. Skrewt's configuration changes to Example 9-13.

## Example 9-13. Skrewt's PVCs are configured as broadcast PVCs, and the OSPFv3 network running over the Frame Relay link is configured as a broadcast network.

```
interface Serial 0/0
 encapsulation frame-relay
 ipv6 address 2001:DB8:0:1::1/64
 ipv6 ospf network broadcast
 ipv6 ospf 1 area 1
 ipv6 ospf priority 20
 frame-relay map ipv6 fe80::206:28ff:feb6:5bc0 201 broadcast
 frame-relay map ipv6 fe80::202:fdff:fe5a:e40 202 broadcast
 frame-relay map ipv6 fe80::201:42ff:fe79:e500 203 broadcast
```

The other routers on the multiaccess network are configured similarly. Notice that there is no longer any need to define IPv6 OSPF neighbors. If the underlying NBMA network is not fully meshed (a PVC from every router to every other router), the DR and BDR must be carefully selected. Both the DR and BDRs must have full virtual circuit connectivity to every other router so the correct adjacencies are formed. Use the interface command ipv6 ospf priority to specify a high priority on the router you wish to be the DR. This method of assigning the DR is discussed in the case study, OSPF on NBMA Networks, in Chapter 8. Example 9-14 shows Skrewt's IPv6 OSPF configuration on serial 0/0.

## Example 9-14. An interface's OSPFv3 for IPv6 configuration is displayed using the *show ipv6 ospf interface* command.

```
Skrewt#show ipv6 ospf interface serial 0/0
Serial0/0 is up, line protocol is up
  Link Local Address FE80::207:85FF:FE6B:EA20, Interface ID 4
```

```
  Area 1, Process ID 1, Instance ID 0, Router ID 1.1.1.1
  Network Type BROADCAST, Cost: 64
  Transmit Delay is 1 sec, State DR, Priority 20
  Designated Router (ID) 1.1.1.1, local address FE80::207:85FF:FE6B:EA20
  Backup Designated router (ID) 10.1.3.1, local address FE80::202:FDFF:FE5A:E40
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:08
  Index 1/1/1, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 4
  Last flood scan time is 0 msec, maximum is 4 msec
  Neighbor Count is 3, Adjacent neighbor count is 3
    Adjacent with neighbor 10.1.1.23
    Adjacent with neighbor 10.1.3.1  (Backup Designated Router)
    Adjacent with neighbor 192.2.2.9
  Suppress hello for 0 neighbor(s)
Skrewt#
```

Note that the network type is BROADCAST. Also note that the neighbor count is 3, and there are 3 adjacent neighbors. This router is the DR, so it has become adjacent to all other routers on the network. The neighbors that are not DR or BDR, such as Flobberworm, have three neighbors, but only two adjacent neighbors, as shown in Example 9-15, because routers on broadcast networks become adjacent to the DR and BDR only.

## Example 9-15. With four routers on a broadcast network, a non-DR or BDR router will have three neighbors, but only be fully adjacent to two of the neighbors.

```
Flobberworm#show ipv6 ospf neighbor

Neighbor ID     Pri   State           Dead Time    Interface ID    Interface
1.1.1.1          20   FULL/DR         00:00:36     3               Ethernet0/0
10.1.1.23         1   2WAY/DROTHER    00:00:36     3               Ethernet0/0
10.1.3.1         15   FULL/BDR        00:00:37     3               Ethernet0/0
Flobberworm#
```

To avoid the DR/BDR election process, the OSPF network type can be changed to point-to-multipoint, as can be seen in Skrewt's modified configuration in Example 9-16.

## Example 9-16. Skrewt's configuration shows that the PVCs continue to broadcast, and that the OSPFv3 network type has been changed to point-to-multipoint.

```
Interface Serial 0/0
 encapsulation frame-relay
 ipv6 address 2001:DB8:0:1::1/64
 ipv6 ospf network point-to-multipoint
 ipv6 ospf 1 area 1
 ipv6 ospf priority 20
 frame-relay map ipv6 fe80::206:28ff:feb6:5bc0 201 broadcast
 frame-relay map ipv6 fe80::202:fdff:fe5a:e40 202 broadcast
 frame-relay map ipv6 fe80::201:42ff:fe79:e500 203 broadcast
```

Skrewt's IPv6 OSPF interface configuration (Example 9-17) shows that the default timers are

different for point-to-multipoint networks then they are for broadcast networks. Hellos are sent every 30 seconds on point-to-multipoint networks and every 10 seconds on broadcast networks. The OSPFv3 interface configuration also does not show any DR, and the router is adjacent with all three neighbors.

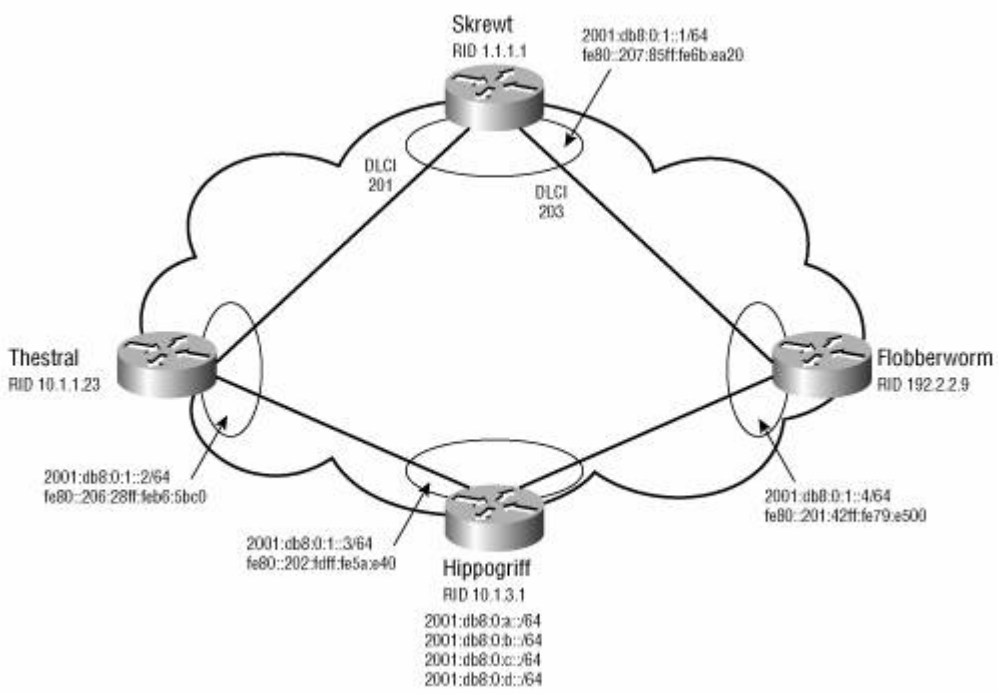
### Example 9-17. The IPv6 OSPF interface configuration of an OSPF point-to-multipoint network is displayed with the command *show ipv6 ospf interface*.

```
Skrewt#show ipv6 ospf interface serial 0/0
Serial0/0 is up, line protocol is up
  Link Local Address FE80::207:85FF:FE6B:EA20, Interface ID 4
  Area 1, Process ID 1, Instance ID 0, Router ID 1.1.1.1
  Network Type POINT_TO_MULTIPOINT, Cost: 64
  Transmit Delay is 1 sec, State POINT_TO_MULTIPOINT,
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
    Hello due in 00:00:13
  Index 1/1/1, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 2, maximum is 4
  Last flood scan time is 0 msec, maximum is 4 msec
  Neighbor Count is 3, Adjacent neighbor count is 3
    Adjacent with neighbor 10.1.1.23
    Adjacent with neighbor 10.1.3.1
    Adjacent with neighbor 192.2.2.9
  Suppress hello for 0 neighbor(s)
Skrewt#
```

An NBMA network configured as an OSPF point-to-multipoint network need not have a full mesh of underlying PVCs. Figure 9-18 shows that some PVCs have been removed from the network in Figure 9-17. The map statement for DLCI 202 has been removed from Skrewt in Example 9-18.

### Figure 9-18. The network of Figure 9-17, with some PVCs removed.

[View full size image]




Example 9-18. Skrewt's Frame Relay configuration maps IPv6 addresses to the remaining two PVCs.

```
interface Serial0/0
 no ip address
 encapsulation frame-relay
 ipv6 address 2001:DB8:0:1::1/64
 ipv6 ospf network point-to-multipoint
 ipv6 ospf priority 20
 ipv6 ospf 1 area 1
 frame-relay map ipv6 FE80::201:42FF:FE79:E500 203 broadcast
 frame-relay map ipv6 FE80::206:28FF:FEB6:5BC0 201 broadcast
```

Hippogriff and the IPv6 prefixes known to Hippogriff are all accessible from Skrewt, as can be seen in Skrewt's IPv6 route table shown in Example 9-19. Skrewt and Hippogriff are both adjacent to both Thestral and Flobberworm, and Hippogriff's IPv6 serial0/0 IPv6 address, 2001:db8:0:1::3, in addition to the IPv6 prefixes advertised by Hippogriff, are routed to via Skrewt's serial0/0 interface, and to the next-hop link-local addresses FE80::206:28FF:FEB6:5BC0 and FE80::201:42FF:FE79:E500.

Example 9-19. Skrewt's route table shows that even the router that does not have a connected PVC to Skrewt is still accessible via the routers that do have connected PVCs in the point-to-multipoint network.

```
Skrewt#show ipv6 route
IPv6 Routing Table - 16 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
C   2001:DB8:0:1::/64 [0/0]
```

```
     via ::, Serial0/0
L   2001:DB8:0:1::1/128 [0/0]
     via ::, Serial0/0
O   2001:DB8:0:1::2/128 [110/64]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
O   2001:DB8:0:1::3/128 [110/128]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
     via FE80::201:42FF:FE79:E500, Serial0/0
O   2001:DB8:0:1::4/128 [110/64]
     via FE80::201:42FF:FE79:E500, Serial0/0
O   2001:DB8:0:5::/64 [110/74]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
O   2001:DB8:0:A::/64 [110/138]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
     via FE80::201:42FF:FE79:E500, Serial0/0
O   2001:DB8:0:B::/64 [110/138]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
     via FE80::201:42FF:FE79:E500, Serial0/0
O   2001:DB8:0:C::/64 [110/138]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
     via FE80::201:42FF:FE79:E500, Serial0/0
O   2001:DB8:0:D::/64 [110/138]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
     via FE80::201:42FF:FE79:E500, Serial0/0
O   2001:DB8:0:50::/64 [110/74]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
O   2001:DB8:0:51::/64 [110/74]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
O   2001:DB8:0:52::/64 [110/74]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
O   2001:DB8:0:53::/64 [110/74]
     via FE80::206:28FF:FEB6:5BC0, Serial0/0
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Skrewt#
```

The IPv6 addresses 2001:db8:0:1::2/64, 2001:db8:0:1::3/64, and 2001:db8:0:1::4/64, that have been configured on the Frame Relay point-to-multipoint interfaces are all advertised by OSPF with 128-bit prefix lengths. These addresses, along with the other IPv6 prefixes advertised by Hippogriff into OSPF, are reachable via the link-local addresses of Skrewt's two adjacent routers.

PREV NEXT

# Troubleshooting OSPFv3

Troubleshooting OSPFv3 for IPv6 should be handled in the same way as OSPFv2 for IPv4. The major difference will be the addressing: OSPFv3 uses the link-local addresses as the source and, when sending directly to a neighbor, destination of packets.

## Case Study: Frame Relay Mapping

Upon initial configuration of Figure 9-17, Skrewt and Hippogriff are configured with the configurations in Example 9-20 and Example 9-21.

### Example 9-20. Skrewt's initial Frame Relay mapping configuration.

```
interface Serial 0/0
 encapsulation frame-relay
 ipv6 address 2001:DB8:0:1::1/64
 ipv6 ospf network broadcast
 ipv6 ospf 1 area 1
 frame-relay map ipv6 2001:db8:0:1::2 201 broadcast
 frame-relay map ipv6 2001:db8:0:1::3  202 broadcast
 frame-relay map ipv6 2001:db8:0:1::4 203 broadcast
 ipv6 ospf 1 area 1
```

### Example 9-21. Hippogriff's initial Frame Relay mapping configuration.

```
interface Serial0/0
 no ip address
 encapsulation frame-relay
 ipv6 address 2001:DB8:0:1::3/64
 ipv6 ospf network broadcast
 ipv6 ospf 1 area 1
 frame-relay map ipv6 2001:DB8:0:1::1 220 broadcast
 frame-relay map ipv6 2001:DB8:0:1::2 221 broadcast
 frame-relay map ipv6 2001:DB8:0:1::4 223 broadcast
```

The other routers are configured similarly.

The routers are not becoming adjacent, as shown in Skrewt's OSPFv3 neighbor table in Example 9-22.

### Example 9-22. Skrewt is not becoming adjacent to the DR or BDR.

```
Skrewt#show ipv6 ospf neighbor

Neighbor ID     Pri   State           Dead Time    Interface ID    Interface
10.1.3.1          1   EXSTART/DR      00:00:34     4               Serial0/0
10.1.1.23         1   EXSTART/BDR     00:00:31     4               Serial0/0
192.2.2.9         1   2WAY/DROTHER    00:00:36     4               Serial0/0
```

```
Skrewt#
```

Debugging IPv6 OSPF Hellos and adjacencies, Example 9-23, show that Hellos are being received and two-way communication is established. The DR and BDR are elected. Database synchronization is attempted to complete the adjacencies with the DR and BDR. Skrewt attempts to send database description packets (DBD) to the potential DR, but never receives an acknowledgment, although Skrewt does continue to receive Hellos from the potential DR. The neighbor state EXSTART with the DR and BDR indicates that a master/slave relationship is being established and an initial DBD packet has been sent.

## Example 9-23. *debug ipv6 ospf hello* and *debug ipv6 ospf adj* show Hellos, two-way communication establishment, DR/BDR election, and the sending of DBD packets.

```
Skrewt#debug ipv6 ospf hello
Skrewt#debug ipv6 ospf adj
OSPFv3: Rcv hello from 10.1.1.23 area 1 from Serial0/0 FE80::202:FDFF:FE5A:E40
interface ID 4
OSPFv3: 2 Way Communication to 10.1.1.23 on Serial0/0, state 2WAY
OSPFv3: Neighbor change Event on interface Serial0/0
OSPFv3: DR/BDR election on Serial0/0
OSPFv3: Elect BDR 10.1.1.23
OSPFv3: Elect DR 10.1.1.23
        DR: 10.1.1.23 (Id) BDR: 10.1.1.23 (Id)
OSPFv3: Send DBD to 10.1.1.23 on Serial0/0 seq 0xF78 opt 0x0013 flag 0x7 len 28
OSPFv3: Rcv hello from 10.1.3.1 area 1 from Serial0/0 FE80::201:42FF:FE79:E500
interface ID 4
OSPFv3: 2 Way Communication to 10.1.3.1 on Serial0/0, state 2WAY
OSPFv3: Neighbor change Event on interface Serial0/0
OSPFv3: DR/BDR election on Serial0/0
OSPFv3: Elect BDR 10.1.1.23
OSPFv3: Elect DR 10.1.3.1
        DR: 10.1.3.1 (Id) BDR: 10.1.1.23 (Id)
OSPFv3: Send DBD to 10.1.3.1 on Serial0/0 seq 0x1C93 opt 0x0013 flag 0x7 len 28
OSPFv3: Remember old DR 10.1.1.23 (id)
OSPFv3: End of hello processing
OSPFv3: Send DBD to 10.1.1.23 on Serial0/0 seq 0xF78 opt 0x0013 flag 0x7 len 28
OSPFv3: Retransmitting DBD to 10.1.1.23 on Serial0/0 [1]
OSPFv3: Send DBD to 10.1.3.1 on Serial0/0 seq 0x1C93 opt 0x0013 flag 0x7 len 28
OSPFv3: Retransmitting DBD to 10.1.3.1 on Serial0/0 [1]
OSPFv3: Send DBD to 10.1.1.23 on Serial0/0 seq 0xF78 opt 0x0013 flag 0x7 len 28
OSPFv3: Retransmitting DBD to 10.1.1.23 on Serial0/0 [2]
OSPFv3: Rcv hello from 10.1.1.23 area 1 from Serial0/0 FE80::202:FDFF:FE5A:E40
interface ID 4
OSPFv3: End of hello processing
Skrewt#
```

No acknowledgments are being received. Adding a debug of IPv6 OSPF packets shows additional information about the DBD packets being sent[3] (Example 9-24).

[3] Debugging IPv6 packets in a live network is not recommended.

## Example 9-24. Adding *debug ipv6 ospf packet* shows packet encapsulation failures.

```
Skrewt#debug ipv6 ospf packet
```

```
OSPFv3: Send DBD to 10.1.1.23 on Serial0/0 seq 0x2422 opt 0x0013 flag 0x7 len 28
IPV6: source FE80::207:85FF:FE6B:EA20 (local)
       dest FE80::202:FDFF:FE5A:E40 (Serial0/0)
       traffic class 224, flow 0x0, len 68+0, prot 89, hops 1, originating
 IPv6: Encapsulation failed
OSPFv3: Retransmitting DBD to 10.1.1.23 on Serial0/0 [4]
```

The DBD packets are not multicast, as Hellos are. They are sent to the IPv6 address of the neighbor router. Recall that OSPFv3 uses the link-local addresses for packet exchange, as can be seen in the output of the IPv6 packet debug. Skrewt does not have a Frame Relay map to FE80::202:FDFF:FE5A:E40 on interface Serial0/0. This is why the encapsulation failed. Frame Relay maps must be configured mapping the link-local address of the neighbor routers to the local DLCI.

To easily obtain the link-local addresses of an IPv6 interface, use `show ipv6 interface serial0/0`, as shown in Example 9-25 .

## Example 9-25. IPv6 information pertinent to an interface can be viewed with the command *show ipv6 interface*.

```
Skrewt#show ipv6 interface serial0/0
Serial0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::207:85FF:FE6B:EA20
  Global unicast address(es):
    2001:DB8:0:1::1, subnet is 2001:DB8:0:1::/64
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::5
    FF02::1:FF00:1
    FF02::1:FF6B:EA20
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is not supported
  ND reachable time is 30000 milliseconds
  Hosts use stateless autoconfig for addresses.
Skrewt#
```

Skrewt's and Hippogriff's configurations are changed as displayed in Example 9-26 and Example 9-27.

## Example 9-26. Skrewt's Frame Relay mapping statements are changed to the link-local IPv6 addresses.

```
interface Serial0/0
 encapsulation frame-relay
 ipv6 address 2001:DB8:0:1::1/64
 ipv6 ospf network broadcast
 ipv6 ospf 1 area 1
 frame-relay map ipv6 fe80::206:28ff:feb6:5bc0 201 broadcast
 frame-relay map ipv6 fe80::202:fdff:fe5a:e40 202 broadcast
 frame-relay map ipv6 fe80::201:42ff:fe79:e500 203 broadcast
```

### Example 9-27. Hippogriff's Frame Relay mapping statements are changed to the IPv6 link-local addresses.

```
interface Serial0/0
 no ip address
 encapsulation frame-relay
 ipv6 address 2001:DB8:0:1::3/64
 ipv6 ospf network broadcast
 ipv6 ospf 1 area 1
frame-relay map ipv6 fe80::207:85ff:fe6b:ea20 220 broadcast
frame-relay map ipv6 fe80::206:28ff:feb6:5bc0 221 broadcast
frame-relay map ipv6 fe80::201:42ff:fe79:e500 223 broadcast
```

The IPv6 address that must be mapped to the DLCI is the neighbor's link-local address. Skrewt's correct IPv6 OSPF neighbor table is shown in Example 9-28 .

### Example 9-28. After mapping the neighbors' link-local addresses to the local DLCIs, Skrewt is fully adjacent to the DR and BDR.

```
Skrewt#show ipv6 ospf neighbor

Neighbor ID     Pri   State           Dead Time    Interface ID    Interface
192.2.2.9         1   FULL/DR         00:00:32     4               Serial0/0
10.1.1.23         1   2WAY/DROTHER    00:00:37     4               Serial0/0
10.1.3.1          1   FULL/BDR        00:00:38     4               Serial0/0
Skrewt#
```

PREV NEXT

# Looking Ahead

We have spent two chaptersone of them very longdescribing the major aspects of OSPF, which is not only the most well-known of the link-state routing protocols, but probably the most widely used IP routing protocol. In the next chapter we will look at a less-known link-state protocol: IS-IS. Some might call this a "more exotic" protocol, but as you will see, it is actually much simpler than OSPF.

# Summary Table: Chapter 9 Command Review

| Command | Description |
|---|---|
| area *area-id* nssa [no-redistribution] [default-information-originate [metric] [metric-type]] [no-summary] | Defines an area as a not so stubby area and gives the option of configuring related parameters. |
| area *area-id* range {*ipv6-prefix* / *prefix-length*} [advertise \| not-advertise] [cost *cost*] | Creates a summary prefix out of more specific routes in the specified area. The summary is flooded (or not flooded, depending on the advertise \| not-advertise option) to the other areas. |
| area *area-id* stub [no-summary] | Defines an area as a stub or totally stubby area. |
| debug ipv6 ospf packet | Provides debugging information about each IPv6 OSPF packet received. |
| debug ipv6 ospf [adj \| ipsec \| database-timer \| flood \| hello \| lsa-generation \| retransmission] | Provides debugging information about specific OSPF packet types. |
| ipv6 ospf *process-id* area *area-id* [instance *instance-id*] | Creates the OSPFv3 process (if it is not already created) and enables OSPF on an interface. |
| ipv6 ospf neighbor *ipv6-address* [priority *number*] [poll-interval *seconds*] [cost *number*] [database-filter all out] | Manually configures OSPFv3 neighbors. |
| ipv6 ospf network {broadcast \| non-broadcast \| {point-to-multipoint [non-broadcast] \| point-to-point}} | Configures the OSPF network type to a type other then the default for a given medium. |
| ipv6 ospf priority *number-value* | Configures the OSPFv3 priority to give to a router for use in DR and BDR election. |
| ipv6 router ospf *process-id* | Enables OSPFv3 router level configuration. |
| show ipv6 ospf [*process-id*] [*area-id*] | Displays general information about the OSPFv3 process. |
| show ipv6 ospf [*process-id*] [*area-id*] neighbor [interface-*type interface-number*] [*neighbor-id*] [detail] | Displays information about OSPFv3 neighbors. |
| show ipv6 ospf [*process-id*] [*area-id*] interface [*interface-type interface-number*] | Displays OSPFv3 information related to the specified interface. |
| summary-prefix *prefix* [not-advertise \| tag *tag-value*] | Summarizes prefixes that are redistributed into OSPFv3. |

# Recommended Reading

Coltun R., D. Ferguson, and J. Moy, "OSPF for IPv6," RFC 2740, December 1999.

# Review Questions

1. Can OSPFv3 support IPv4v?
2. What is meant, in OSPFv3, that it can support multiple instances per link? What field in the OSPFv3 message header makes this possible?
3. How are OSPFv3 packets authenticated?
4. What is the OSPFv3 Next Header number?
5. What are the two reserved OSPFv3 multicast addresses?
6. Does OSPFv3 use any different message types than OSPFv2?
7. What is the purpose of the first three bits of the Link State Type field of the OSPFv3 LSA header?
8. What flooding scope is supported by OSPFv3 but not by OSPFv2? What LSA uses this flooding scope?
9. What is the most significant difference between OSPFv2 Router and Network LSAs, and OSPFv3 Router and Network LSAs?
10. What is the purpose of the Intra-Area Prefixes LSA?
11. What is the purpose of the Link LSA?

# Configuration Exercises

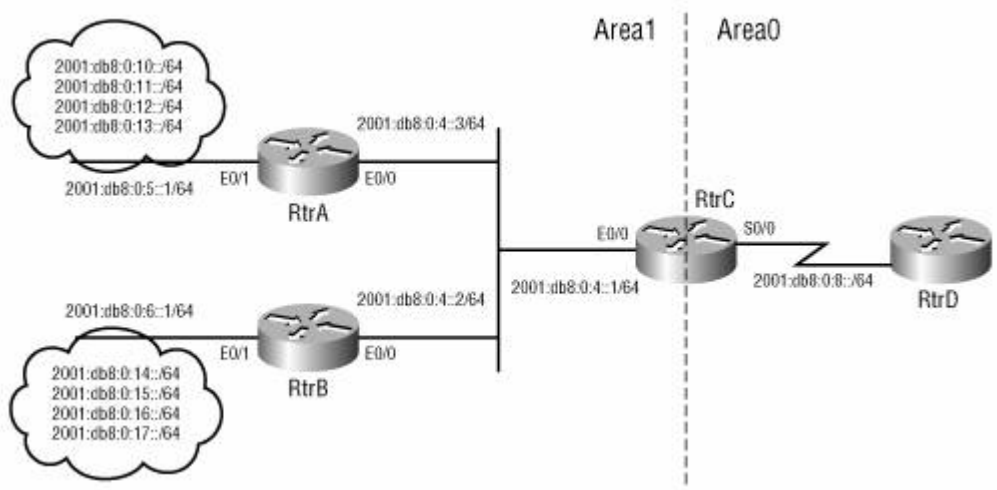
1 Which OSPFv3 command is required to include a second IPv6 prefix that has been configured on an interface in the OSPFv3 process?

2 Can two routers become adjacent if one is configured with the IPv6 address 2001:db8:0:1::1/64, and the other is configured with the address 2001:db8:0:100::1/126?

3 Provide the IPv6 OSPFv3 configuration of the routers in Figure 9-19.

Figure 9-19. IPv6 OSPFv3 network.

[View full size image]




4 Configure area 1 in Figure 9-19 as a totally stubby area and summarize addresses into area 0.

# Chapter 10. Integrated IS-IS

This chapter covers the following subjects:

- Operation of Integrated IS-IS
- Configuring Integrated IS-IS
- Troubleshooting Integrated IS-IS

When the terms *link-state protocol* and *IP* are mentioned together, almost everyone thinks of Open Shortest Path First (OSPF). Some might say, "Oh, yeah, there's also IS-IS, but Idunnomuchaboutit." Only a few will think of Integrated IS-IS as a serious alternative to OSPF. However, those few do exist, and there are networksprimarily ISPs and carriersthat route IP with IS-IS.

IS-IS, which stands for Intermediate System to Intermediate System, is the routing protocol for the ISO's Connectionless Network Protocol (CLNP). It is described in ISO 10589.[1] The first production incarnation of the protocol was developed by Digital Equipment Corporation for its DECnet Phase V.

[1] International Organization for Standardization, "Intermediate System to Intermediate System Intra-Domain Routeing Information Exchange Protocol for Use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)," ISO/IEC 10589, 1992.

The ISO was working on IS-IS at more or less the same time that the Internet Architecture Board (IAB) was working on OSPF, and there was a proposal that IS-IS be adopted as the routing protocol for TCP/IP in place of OSPF. This proposal was driven by the opinion, popularly held in the late 1980s and early 1990s, that TCP/IP was an interim protocol suite that would eventually be replaced by the OSI suite. Adding impetus to this movement toward OSI were specifications such as the United States' Government Open Systems Interconnection Profile (GOSIP) and the European Procurement Handbook for Open Systems (EPHOS).

To support the predicted transition from TCP/IP to OSI, an extension to IS-IS, known as *Integrated IS-IS*, was proposed.[2] The purpose of Integrated IS-IS, also known as *Dual IS-IS*, was to provide a single routing protocol with the capabilities of routing both CLNS[3] and IP. The protocol was designed to operate in a pure CLNS environment, a pure IP environment, or a dual CLNS/IP environment.

[2] Ross Callon, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments," RFC 1195, December 1990.

[3] Connectionless-Mode Network Servicethe network layer protocol of CLNP.

Saying that battle lines were drawn might be overly dramatic, but at the least strong pro-ISO and pro-OSPF factions developed. It can be enlightening to read and contrast the coverage of OSPF and IS-IS in the well-known books by Christian Huitema,[4] a past chairman of the IAB, and Radia Perlman,[5] the chief designer of IS-IS. In the end, the Internet Engineering Task Force (IETF) adopted OSPF as the recommended IGP. Technical differences certainly influenced the decision, but so did political considerations. ISO standardization is a slow, four-step process that depends upon the voted approval of many committees. The IETF, on the other hand, is much more freewheeling. A statement made in 1992 has been its informal motto: "We reject kings, presidents, and voting; we believe in rough consensus and running code."[6] Letting OSPF evolve through the RFC process made more sense than adopting the more formalized IS-IS.

[4] Christian Huitema, Routing in the Internet, Prentice Hall PTR, Englewood Cliffs, NJ, 1995.

[5] Radia Perlman, Interconnections: Bridges and Routers, Addison-Wesley, Reading, MA, 1992.

[6] Dave Clark, quoted in Huitema, p. 23.

On the other hand, the very small but very sophisticated IS-IS user community has proven to be an advantage for both IS-IS implementers and users. Agreement on extensions tends to be reached quickly, and because the users are mostly high-end ISPs and carriers, they hold tremendous leverage over their router vendors. Having a carrier-grade IS-IS implementation is a must for any router vendor that wishes to enter the high-performance network market.

Despite the political friction, the OSPF Working Group learned from and capitalized on many of the basic mechanisms designed for IS-IS. On the surface, OSPF and IS-IS have many features in common:

- They both maintain a link-state database from which a Dijkstra-based SPF algorithm computes a shortest-path tree.
- They both use Hello packets to form and maintain adjacencies.
- They both use areas to form a two-level hierarchical topology.
- They both have the capability of providing address summarization between areas.
- They both are classless protocols.
- They both elect a designated router to represent broadcast networks.
- They both have authentication capabilities.

Below these similarities, there are distinct differences. This chapter begins by examining those differences. Integrated IS-IS (henceforth referred to simply as IS-IS) is covered only as an IP routing protocol; CLNS is discussed only when it is relevant to using IS-IS to route IP. And as mentioned already, IS-IS is almost exclusively found in service provider networks where reliability and scalability are paramount; so an additional focus of this chapter is to look at the features of IS-IS that meet these requirements in very large networks.

PREV NEXT

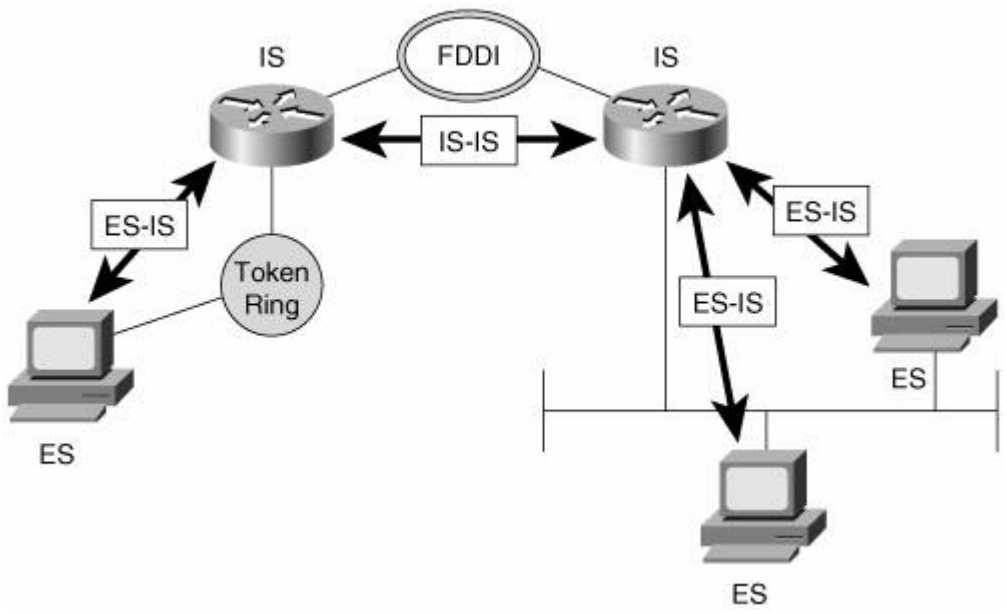
# Operation of Integrated IS-IS

The ISO often uses different terms than the IETF to describe the same entities, a fact that can sometimes cause confusion. ISO terms are introduced and defined in this section, but in most cases the more familiar IETF terminology used throughout the rest of this book is used in this chapter.[7] Some ISO terms are so fundamental that they should be discussed before getting into any specifics of the IS-IS protocol.

[7] The temptation to use the ISO/European spelling of certain common terms such as "routeing" and "neighbour" was successfully resisted.

A router is an *Intermediate System* (IS), and a host is an *End System* (ES). Accordingly, a protocol that provides communication between a host and a router is known as ES-IS, and a protocol that routers use to speak to each other (a routing protocol) is IS-IS (Figure 10-1). Whereas IP uses router discovery mechanisms, such as Proxy ARP, IRDP, or in the case of IPv6 NDP, or simply has a default gateway configured on hosts, CLNP uses ES-IS to form adjacencies between End Systems and Intermediate Systems. ES-IS has no relevance to IS-IS for IP and is not covered in this book.

Figure 10-1. Hosts are End Systems in ISO terminology, and routers are Intermediate Systems.




An interface attached to a subnetwork is a *Subnetwork Point of Attachment* (SNPA). An SNPA is somewhat conceptual because it actually defines the point at which subnetwork services are provided, rather than an actual physical interface. The conceptual nature of SNPAs fits with the conceptual nature of subnetworks themselves, which can consist of multiple data links connected by data-link switches.

A data unit passed from an OSI layer of one node to the peer OSI layer of another node is called a *Protocol Data Unit* (PDU). So a frame is a Data Link PDU (DLPDU), and a packet is a Network PDU (NPDU). The data unit that performs the equivalent function of the OSPF LSA is the Link State PDU (LSP).[8] Unlike LSAs, which are encapsulated behind an OSPF header and then an IP

packet, an LSP is itself a packet.

[8] Some documentation, such as RFC 1195, defines the LSP as Link State Packet.

## IS-IS Areas

Although both IS-IS and OSPF use areas to create a two-level hierarchical topology, a fundamental difference exists in the way in which the two protocols define their areas. OSPF area borders are marked by routers, as shown in Figure 10-2. Some interfaces are in one area, and other interfaces are in another area. When an OSPF router has interfaces in more than one area, it is an Area Border Router (ABR).

Figure 10-2. OSPF area borders fall within routers, and the routers that connect the areas are ABRs.

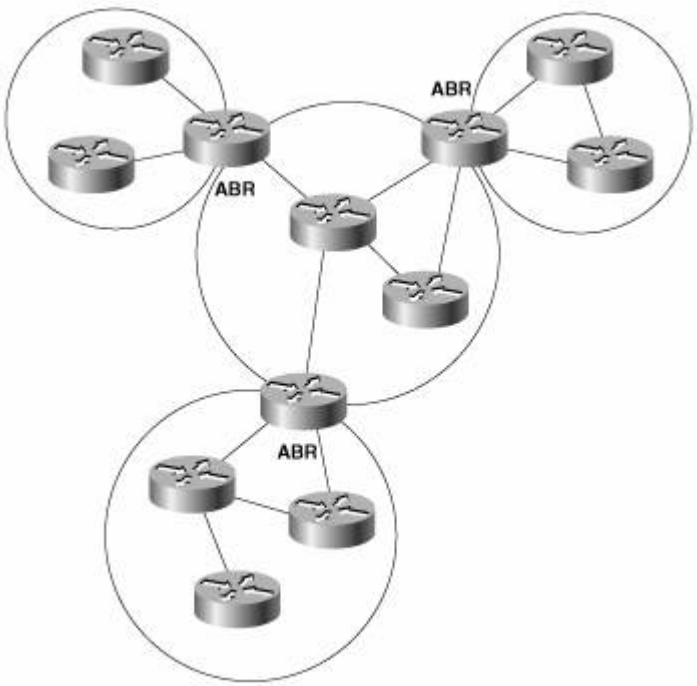
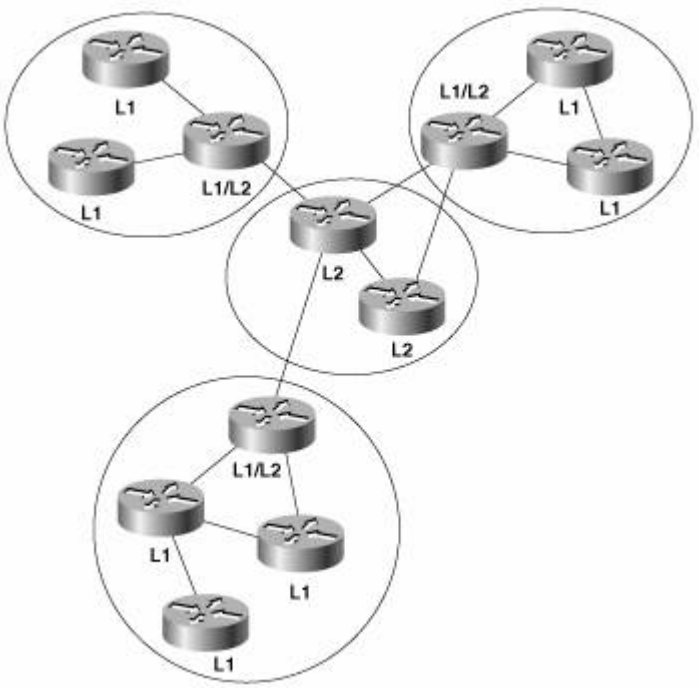



Figure 10-3 shows the same topology as shown in Figure 10-2 but with IS-IS areas. Notice that all the routers are completely within an area, and the area borders are on links, not on routers. The IS-IS "backbone area" is a *Level 2* area, while non-backbone areas are *Level 1* areas.

Figure 10-3. IS-IS areas fall on links, and the routers that connect the areas are level 2 routers.




An intermediate system can be a level 1 (L1) router, a level 2 (L2) router, or both (L1/L2). L1 routers are analogous to OSPF nonbackbone Internal Routers, L2 routers are analogous to OSPF backbone routers, and L1/L2 routers are analogous to OSPF ABRs. In Figure 10-3, the L1/L2 routers are connected to L1 routers and to L2 routers. These L1/L2 routers must maintain both a level 1 link-state database and a level 2 link-state database, in much the same way that an OSPF ABR must maintain a separate database for each area to which it is attached. Cisco routers are configured as L1-only, L2-only, or L1/L2 with the command `is-type`. By default, they are L1/L2.

While the area topology shown in Figure 10-3 is useful for contrasting IS-IS areas with OSPF areas, it can also be slightly misleading; IS-IS areas are not quite as clear-cut as OSPF areas are. Rather than think of them as topological areas, IS-IS areas are best thought of as a set of adjacencies. An adjacency, in turn, can be either L1 or L2. So, for example, a given L1 area is actually a contiguous set of L1 adjacencies between routers with the same AID. This is especially important for understanding L2 areas, which are always defined as a set of L2 adjacencies. An L1/L2 router, then, is a router in an L1 area that has both one or more L1 adjacencies and one or more L2 adjacencies. An L2 router is one that has only L2 adjacencies.

It is also possible for both an L1 and an L2 adjacency to exist between two neighbors; because IS-IS areas are defined as a series of adjacencies, the implication of having both an L1 and an L2 adjacency between the same two neighbors is that IS-IS areas can overlap. This is in contrast to OSPF, in which area boundaries are more distinct.

As with OSPF, inter-area traffic must traverse an L2 area to prevent inter-area routing loops. Every L1 router within an area (including the area's L1/L2 routers) maintains an identical link-state database. Unlike OSPF ABRs, L1/L2 routers do not by default advertise L2 routes to L1 routers. Therefore, an L1 router has no knowledge of destinations outside of its own area. In this sense, an L1 area is similar to an OSPF totally stubby area. To route a packet to another area, an L1 router must forward the packet to an L1/L2 router. When an L1/L2 router sends its level 1 LSP into an area, it signals other L1 routers that it can reach another area by setting a bit known as the *Attached* (ATT) bit[9] in the LSP.

[9] There are actually four ATT bits, relevant to different metrics. These bits are further explained in the section "IS-IS PDU Formats."

ISO 10589 describes a procedure by which IS-IS routers can create virtual links to repair partitioned areas, just as OSPF can. This feature is not supported by Cisco or by most other router vendors and is not further described here. The primary reason vendors do not support IS-

IS virtual links is simply that their customers have not asked for them, and this goes to a fundamental difference in OSPF and IS-IS application. OSPF, with its rich area utilities and feature set, is the protocol of choice for enterprise networks. Multi-area IS-IS, on the other hand, is more complex to run and so is seldom if ever found in enterprise networks. Carriers and ISPs run BGP-based networks, and their IGP is used mainly for finding BGP session endpoints. So they want their IGP to be as simple as possibleoften making their entire routing domain a single IGP area. IS-IS is certainly simpler than OSPF in many aspects, and many will argue that it is more scalable in this type of "single large area" application. So when you encounter IS-IS in a service provider network, what you will usually encounter is a single L2 area.[10]

[10] L2 areas are used so that on the outside chance that a separate area must be added, the new area can be made L1 and easily connected to the existing L2 area.

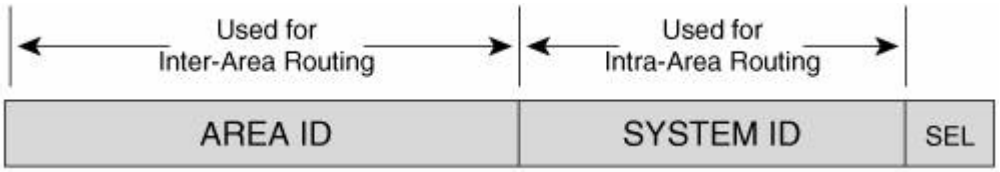
Because an IS-IS router resides completely within a single area, the Area ID (or area address) is associated with the entire router rather than an interface. A unique feature of IS-IS is that a router can have, by default, up to three area addresses, which can be useful during area transitions. Using the IOS command max-area-addresses, you can increase an area's addresses up to 254. A configuration case study later in this chapter, "An Area Migration," demonstrates the use of multiple area addresses. Each IS-IS router must also have a way to uniquely identify itself within the routing domain. This identification is the function of the *System ID*, which is analogous to the OSPF Router ID. Both the Area ID and the System ID are defined on an IS-IS router by a single address, the Network Entity Title (NET).

## Network Entity Titles

Even when IS-IS is used to route only TCP/IP, IS-IS is still an ISO CLNP protocol. Consequently, the packets by which IS-IS communicates with its peers are CLNS PDUs, which in turn means that even in an IP-only environment, an IS-IS router must have an ISO address. The ISO address is a network address, known as *Network Entity Title* (NET), described in ISO 8348.[11] The length of a NET can range from 8 to 20 octets; the NET describes both the Area ID and the System ID of a device, as shown in Figure 10-4.

[11] International Organization for Standardization, "Network Service Definition Addendum 2: Network Layer Addressing," ISO/IEC 8348/Add.2, 1988.

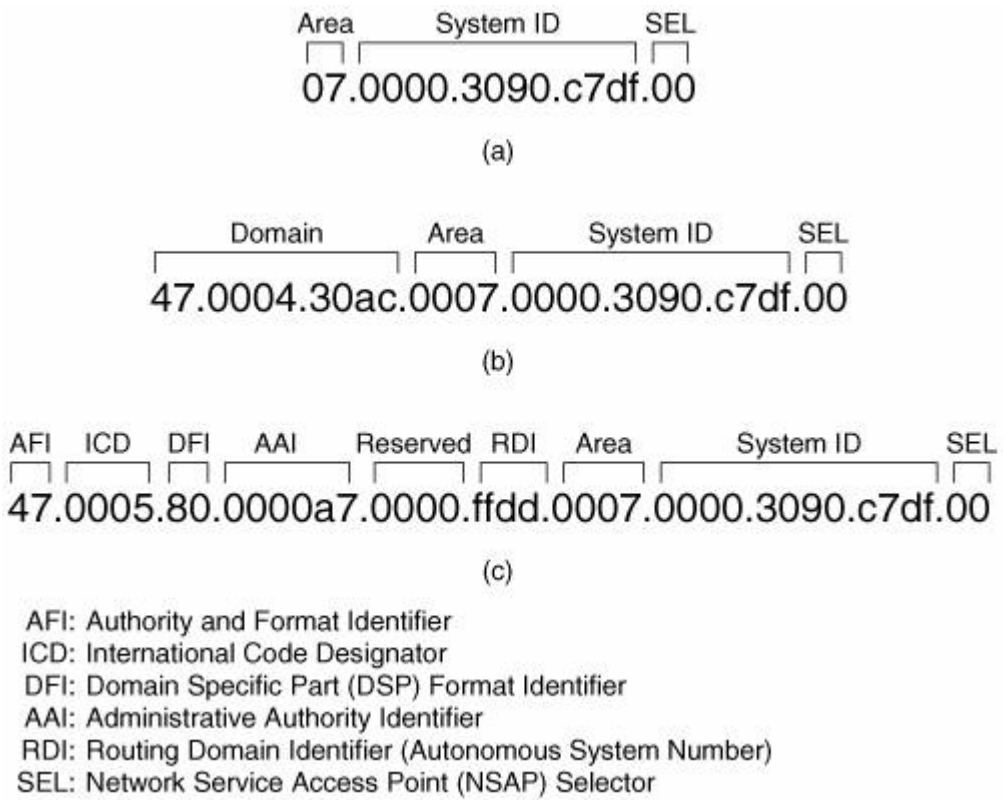
Figure 10-4. The NET specifies the Area ID and the System ID of an IS or ES.




The ISO designed the NET to be many things to many systems, and depending upon your viewpoint, the address format is either extremely flexible and scalable or it is a cumbersome muddle of variable fields. Figure 10-5 shows just three of the many forms an ISO NET can take. Although the fields preceding the System ID are different in each example, the System ID itself is the same. ISO 10589 specifies that the field can be from one to eight octets in length, but that the System ID of all nodes within the routing domain must use the same length. In practice, the length is always six octets[12] and is often adapted from the Media Access Control (MAC) identifier of an interface on the device. The System ID must be unique for each node within the routing domain.

[12] The Cisco IS-IS implementation requires a System ID of 6 octets.

Figure 10-5. Three NET formats: A simple eight-octet Area ID/System ID format (a); an OSI NSAP format (b); and a GOSIP NSAP format[13] (c).




[13] GOSIP Advanced Requirements Group, "Government Open Systems Interconnection Profile (GOSIP) Version 2.0 [Final Text]," Federal Information Processing Standard, U.S. Department of Commerce, National Institute of Standards and Technology, October 1990.

Also of interest in the examples of Figure 10-5 is the NSAP Selector (SEL). In each case, this one-octet field is set to 0x00. A Network Service Access Point (NSAP) describes an attachment to a particular service at the network layer of a node. So when the SEL is set to something other than 0x00 in an ISO address the address is an NSAP address. This situation is similar to the combination of IP destination address and IP protocol number in an IP packet, which addresses a particular service at the network layer of a particular device's TCP/IP stack. When the SEL is set to 0x00 in an ISO address, the address is a NET, the address of a node's network layer itself.

As the variety of formats in Figure 10-5 implies, a detailed discussion of the configuration of NETs is beyond the scope of this book. A good reference for further study is RFC 1237.[14] In an IP-only environment, the NET might be assigned based on a standard such as GOSIP. If you are free to choose any NET for an IP-only environment, choose the simplest format that will serve the needs of your network.

[14] Richard Colella, Ella Gardner, and Ross Callon, "Guidelines for OSI NSAP Allocation in the Internet," RFC 1237, July 1991.

Regardless of the format, the following three rules apply:

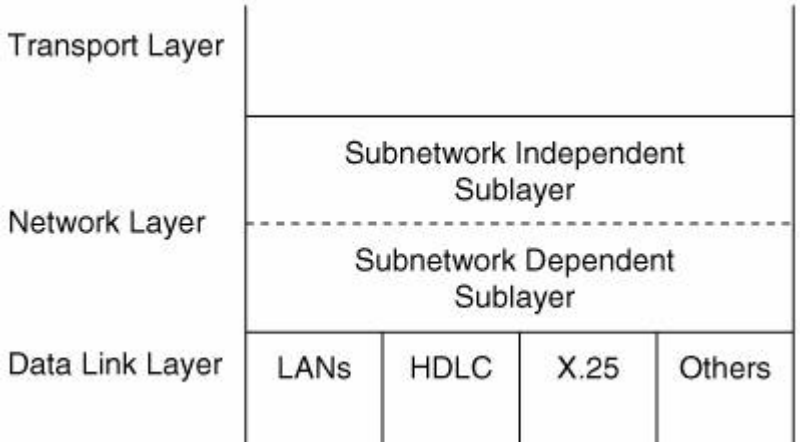
- The NET must begin with a single octet (for example, 47.xxxx. . .).
- The NET must end with a single octet, which should be set to 0x00 (. . .xxxx.00). IS-IS will function if the SEL is nonzero but a dual CLNP/IP router might experience problems.

- On Cisco routers, the System ID of the NET must be six octets.

## IS-IS Functional Organization

One of the primary reasons for having a layered network architecture like the OSI model is so that the functions of each layer are independent from the layer below. The network layer, for example, must adapt to many types of data links, or subnetworks. To further this adaptability, the network layer consists of two sublayers (Figure 10-6). The *Subnetwork Independent sublayer* provides consistent, uniform network services to the transport layer. The *Subnetwork Dependent sublayer* accesses the services of the data-link layer on behalf of the Subnetwork Independent sublayer. As the two names imply, the Subnetwork Dependent sublayer depends upon a specific type of data link, so that the Subnetwork Independent sublayer can be independent of the data link.

Figure 10-6. The OSI network layer consists of two sublayers.




The organization of the network layer, specified in ISO 8648,[15] is actually more complex than that shown in Figure 10-6. The two basic sublayers are introduced here because ISO 10589 presents much of its description of the operation of IS-IS within the framework of the functions of these sublayers.

[15] International Organization for Standardization, "Internal Organisation of the Network Layer," ISO 8648, 1990.

### Subnetwork Dependent Functions

The subnetwork dependent functions are, of course, the functions of the Subnetwork Dependent sublayer. Their job is to hide the characteristics of different kinds of data links (subnetworks) from the functions of the Subnetwork Independent sublayer. The following subnetwork dependent functions are important to routing:

- The transmission and reception of PDUs over the specific attached subnetwork
- The exchange of IS-IS Hello PDUs to discover neighbors and establish adjacencies on the subnetwork
- The maintenance of the adjacencies
- Link demultiplexing, or the transfer of OSI PDUs to the OSI process and the transfer of IP

packets to the IP process

In contrast to the four network types defined by OSPF, IS-IS defines only two: broadcast subnetworks and point-to-point or general topology subnetworks. *Broadcast subnetworks* are defined the same as they are under OSPFmulti-access data links that support multicasting. Point-to-point (nonbroadcast) subnetworks can be permanent, such as a T1 link, or dynamically established, such as X.25 SVCs.

## Neighbors and Adjacencies

IS-IS routers discover neighbors and form adjacencies by exchanging IS-IS Hello PDUs. Hellos are transmitted every 10 seconds, and on Cisco routers this interval can be changed on a per interface basis with the command `isis hello-interval`. Although IS-IS Hellos are slightly different for broadcast and point-to-point subnetworks, the Hellos include the same essential information, described in the section "IS-IS PDU Formats." An IS-IS router uses its Hello PDUs to identify itself and its capabilities and to describe the parameters of the interface on which the Hellos are sent. If two neighbors are in agreement about their respective capabilities and interface parameters, they become adjacent. IS-IS is, however, less strict than OSPF in accepting adjacencies. In most cases, capabilities advertised by one neighbor but not supported by the other neighbor will not prevent an adjacency from forming; the capability is just ignored. The neighbors can even advertise different Hello intervals.

IS-IS forms separate adjacencies for L1 and L2 neighbors. L1-only routers form L1 adjacencies with L1 and L1/L2 neighbors, and L2-only routers form L2 adjacencies with L2 and L1/L2 neighbors. Neighboring L1/L2 routers can form both an L1 adjacency and an L2 adjacency. An L1-only router and an L2-only router will not establish an adjacency. As mentioned previously, Cisco routers are by default L1/L2.

While the type of router (L1-only, L2-only, or L1/L2) influences the type of adjacency that is formed, so do the area IDs configured on the two neighbors in question. The following rules apply:

- Two L1-only routers form an L1 adjacency only if their AIDs match.
- Two L2-only routers form an L2 adjacency, even if their AIDs are different.
- An L1-only router forms an L1 adjacency with an L1/L2 router only if their AIDs match.
- An L2-only router forms an L2 adjacency with an L1/L2 router even if their AIDs are different.
- Two L1/L2 routers form both L1 and L2 adjacencies if their AIDs match.
- Two L1/L2 routers form only an L2 adjacency if their AIDs do not match.

Once an adjacency is established, the Hellos act as keepalives. Each router sends a *hold time* in its Hellos, informing its neighbors how long they should wait to hear the next Hello before declaring the router dead. The default hold time on Cisco routers is three times the Hello interval and can be changed on a per interface basis with the command `isis hello-multiplier`. A significant difference from OSPF is that the hello intervals and hold times between two IS-IS neighbors do not have to match. Each router honors the hold time advertised by its neighbor.

Another interesting difference between OSPF and IS-IS adjacencies is when two routers become adjacent. OSPF can be a bit confusing here; two routers are considered adjacent as soon as two-way communication is established, but not *fully adjacent* until database synchronization is completed. IS-IS considers routers adjacent as soon as they have exchanged Hellos.

The IS-IS neighbor table can be observed with the command `show clns is-neighbors` (Example 10-1). The first four columns of the display show the System ID of each neighbor, the interface on which the neighbor is located, the state of the adjacency, and the adjacency type. The state

will be either Init, indicating that the neighbor is known, but is not adjacent, or Up, indicating that the neighbor is adjacent. The priority is the router priority used for electing a Designated Router on a broadcast network, as described in the next section.

Example 10-1. The command *show clns is-neighbors* displays the IS-IS neighbor table.

```
Brussels#show clns is-neighbors
System Id      Interface  State  Type  Priority  Circuit Id         Format
0000.0C04.DCC0 Se0        Up     L1    0         06                 Phase V
0000.0C04.DCC0 Et1        Up     L1    64        0000.0C76.5B7C.03  Phase V
0000.0C0A.2C51 Et0        Up     L2    64        0000.0C76.5B7C.02  Phase V
0000.0C0A.2AA9 Et0        Up     L1L2  64/64     0000.0C76.5B7C.02  Phase V
Brussels#
```

The sixth column shows the *Circuit ID*, a one-octet number that the router uses to uniquely identify the IS-IS interface. If the interface is attached to a broadcast multiaccess network, the Circuit ID is concatenated with the System ID of the network's Designated Router, and the complete number is known as the *LAN ID*. In this usage, the Circuit ID is more correctly called the *Pseudonode ID*. For example, in Example 10-1 the LAN ID of the link attached to interface E0 is 0000.0c76.5b7c.02. The System ID of the Designated Router is 0000.0c76.5b7c, and the Pseudonode ID is 02.

The last column indicates the format of the adjacency. For Integrated IS-IS the format will always be Phase V, indicating OSI/DECnet Phase V. The only other adjacency format is DECnet Phase IV.

## Designated Routers

IS-IS elects a Designated Router (or more officially, a Designated IS) on broadcast multi-access networks for the same reason OSPF does. Rather than having each router connected to the LAN advertise an adjacency with every other router on the network, the network itself is considered a routera pseudonode. Each router, including the Designated Router, advertises a single link to the pseudonode. The DR also advertises, as the representative of the pseudonode, a zero-cost link to all of the attached routers.

Unlike OSPF, however, an IS-IS router attached to a broadcast multi-access network establishes adjacencies with all of its neighbors on the network, not just the DR. This is in keeping with the mention in the previous section that IS-IS adjacencies are established as soon as Hellos are exchanged. Each router multicasts its LSPs to all of its neighbors, and the DR uses a system of PDUs called Sequence Number PDUs (SNPs) to ensure that the flooding of LSPs is reliable. The reliable flooding process, and SNPs, are described in a later section, "Update Process."

The IS-IS Designated Router election process is quite simple. Every IS-IS router interface is assigned both an L1 priority and an L2 priority in the range of 0 to 127. Cisco router interfaces have a default priority of 64 for both levels, and this value can be changed with the command isis priority.

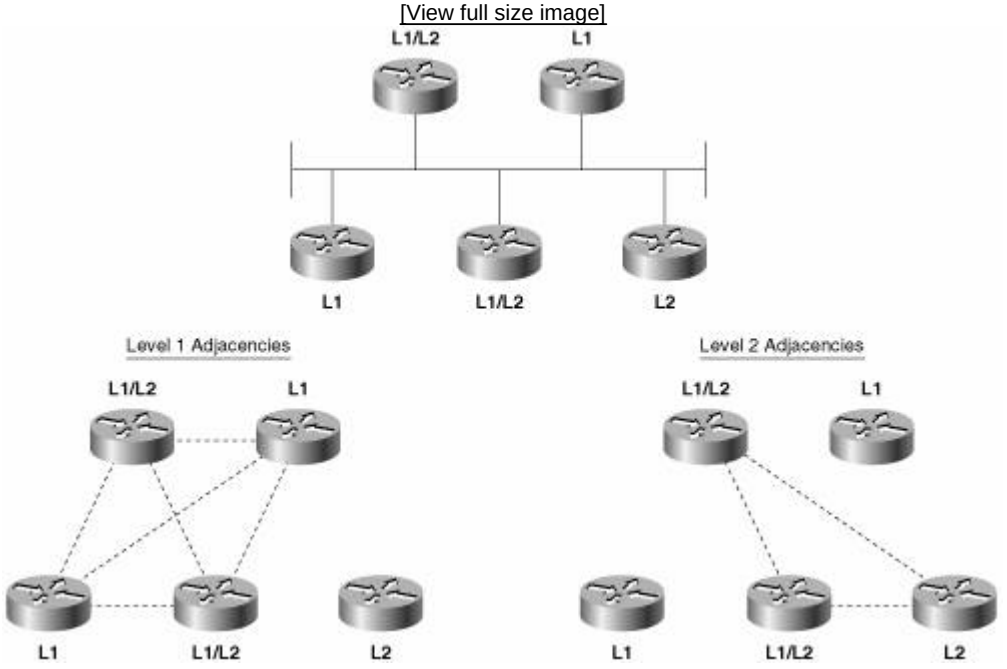
The router advertises its priority in the Hellos sent from each interfacethe L1 priority is advertised in L1 Hellos, and the L2 priority is advertised in L2 Hellos. Unlike OSPF, in which a priority of 0 makes the router ineligible to become the DR, an IS-IS router with a priority of 0 is merely the lowest priority, but can still become the DR. Interfaces to nonbroadcast networks, on which a DR is not elected, always have a priority of 0 (notice the priority of the serial interface in Example 10-1). The router with the highest priority becomes the DR. In a tie, the router whose attached interface has the numerically highest MAC address becomes the DR.

As the L1 and L2 priorities suggest, separate DRs are elected on a network for level 1 and level

2. This outcome is necessary because of the separate L1 and L2 adjacencies that can exist on a single LAN, as shown in Figure 10-7. Because an interface can have separate priorities for each level, the L1 DR and the L2 DR on a LAN might or might not be the same router.

## Figure 10-7. Separate adjacencies are established for level 1 and for level 2, so separate DRs must be elected for level 1 and level 2.

[View full size image]




The DR assigns the LAN ID to the network. As discussed in the preceding section, the LAN ID is a concatenation of the DR's System ID and its Pseudonode ID for the attached network. All other routers on the network will use the LAN ID assigned by the DR.

Example 10-2 shows the neighbor table of a router whose E0 interface is attached to the same network as the E0 interface of the router of Example 10-1. By comparing these two tables, you can see that three routers are attached to the Ethernet: 0000.0c0a.2aa9, 0000.0c0a.2c51, and 0000.0c76.5b7c. All have a priority of 64, so the one with the numerically highest MAC address is the DR. That router, 0000.0c76.5b7c, identifies the network with Circuit ID 02. Therefore, the LAN ID shown in both Example 10-1 and Example 10-2 is 0000.0c76.5b7c.02.

Appending the Circuit ID to the System ID is necessary because the same router can be the DR on more than one network. Notice in Example 10-1 that the same router is the DR on the network attached to E0 and the network attached to E1. It is the Circuit ID, 02 on E0 and 03 on E1, which makes the LAN ID unique for each network.

## Example 10-2. The E0 interface of this router is attached to the same network as the E0 interface of the router in Example 10-1.

```
London#show clns is-neighbors
System Id       Interface  State  Type  Priority  Circuit Id         Format
0000.0C76.5B7C  Et0        Up     L2    64        0000.0C76.5B7C.02  Phase V
0000.0C0A.2AA9  Et0        Up     L2    64        0000.0C76.5B7C.02  Phase V
0000.3090.6756  Se0        Up     L1    0         02                 Phase V
London#
```

The IS-IS DR process is more primitive (or less complex, depending upon one's viewpoint) than the OSPF DR process in two ways. First, IS-IS does not elect a Backup Designated Router. If the IS-IS DR fails, a new DR is elected. Second, the IS-IS DR is not as stable as the OSPF DR. If an OSPF router becomes active on a network with an existing DR, the new router will not become the DR even if its priority or Router ID is higher. As a result, the OSPF DR usually is the router that has been active on the network for the longest time. In contrast to the OSPF rules, a new IS-IS router with a higher priority than the existing DR, or an equal priority and a higher MAC address, will become the new DR. Each time the DR is changed, a new set of LSPs must be flooded. However, given that adding a new router to a broadcast link is a relatively uncommon occurrence in the operational day of a network, and that DR election happens very quickly in modern routerson the order of microsecondsthe lack of a BDR and the fact that a DR can be preempted should not be a cause for concern.

## Subnetwork Independent Functions

The functions of the subnetwork-independent sublayer define how CLNS delivers packets throughout the CLNP network and how these services are offered to the transport layer. The routing function itself is divided into four processes: the Update process, the Decision process, the Forwarding process, and the Receive process. As the names of the last two processes suggest, the Forwarding process is responsible for transmitting PDUs, and the Receive process is responsible for the reception of PDUs. These two processes, as described in ISO 10589, are more relevant to CLNS NPDUs than to IP packets, and so are not described further.

## Update Process

The Update process is responsible for constructing the L1 and L2 link-state databases. To do so, L1 LSPs are flooded throughout the area in which they are originated, and L2 LSPs are flooded over all L2 adjacencies. The specific fields of the LSPs are described in the section "IS-IS PDU Formats."

Each LSP contains a Remaining Lifetime, a Sequence Number, and a Checksum. The Remaining Lifetime is an age. The difference between an IS-IS LSP's Remaining Lifetime and an OSPF LSA's Age parameter is that the LSA Age increments from zero to MaxAge, whereas the LSP Remaining Lifetime begins at MaxAge and decrements to zero. The IS-IS MaxAge is 1200 seconds (20 minutes). Like OSPF, IS-IS ages each LSP as it resides in the link-state database, and the originator must periodically refresh its LSAs to prevent the Remaining Lifetime from reaching zero. The IS-IS refresh interval is 15 minutes minus a random jitter of up to 25 percent. If the Remaining Lifetime reaches zero, the expired LSP will be kept in the link-state database for 60 seconds, known as the *ZeroAgeLifetime*.

An interesting and important difference from the OSPF LSA refresh process is that because the IS-IS Remaining Lifetime begins as a non-zero number and is decremented down to zero, the beginning number can be changed from its default of 1200 seconds. In fact, it can be increased up to 65,535 secondsapproximately 18.2 hours. This is a major contributing factor to IS-IS scalability in very large areas. If the area links are reasonably stable, increasing the refresh time and the Remaining Lifetime can sharply reduce the flooding load caused by LSP refreshes. The initial Remaining Lifetime (MaxAge) value that is set in locally originated LSPs is changed with the command `max-lsp-lifetime`, and the period between refreshes of locally originated LSPs is set with the command `lsp-refresh-interval`. If you change the defaults, always set the refresh interval at least a few hundred seconds less than the Remaining Lifetime so that newly refreshed LSPs arrive at all routers in the area before the old instances of the LSPs age out.

Under older IS-IS procedures, if any router receives an LSP with an incorrect checksum, the router will purge the LSP by setting the LSP's Remaining Lifetime to zero and reflooding it. The purge causes the LSP's originator to send a new instance of the LSP. However, the second edition of ISO 10589 does away with purging LSPs with incorrect checksums because on error-prone subnetworks, allowing receiving routers to initiate purges can significantly increase the LSP traffic.

IOS implementations before 12.0 follow the older purging procedure, but you can override this behavior with the command ignore-lsp-errors. When a router with this option enabled receives a corrupted LSP, it drops it rather than purging it. The originator of the corrupted LSP will still know that the LSP was not received through the use of SNPs, described later in this section. As of 12.0, the newer IS-IS procedures are implemented and ignore-lsp-errors is enabled by default.

A significant point about the LSP purging, however, is yet another way that IS-IS differs from OSPF, where only the originator can purge an OSPF LSA.

The Sequence Number is an unsigned, 32-bit linear number. When a router first originates an LSP, it uses a Sequence Number of one, and each subsequent instance of the LSP has its Sequence Number incremented by one. If the Sequence Number increments to the maximum (0xFFFFFFFF), the IS-IS process must shut down for at least 21 minutes (MaxAge + ZeroAgeLifetime) to allow the old LSPs to age out of all databases.

On point-to-point subnetworks, routers send L1 and L2 LSPs directly to the neighboring router. On broadcast subnetworks, the LSPs are multicast to all neighbors. Frames carrying L1 LSPs have a destination MAC identifier of 0180.c200.0014, known as AllL1ISs. Frames carrying L2 LSPs have a destination MAC identifier of 0180.c200.0015, known as AllL2ISs.

IS-IS uses SNPs both to acknowledge the receipt of LSPs, to request LSPs, and to help maintain link-state database synchronization. There are two types of SNPs: *Partial SNPs* (PSNP) and *Complete SNPs* (CSNP). On a point-to-point subnetwork, a router uses a PSNP to explicitly acknowledge each LSP it receives.[16] The PSNP describes the LSP it is acknowledging by including the following information:

[16] The exception is if a router receives an instance of an LSP that is older than an instance of the same LSP in its database. In that case, the router will reply with the newer LSP.

- The LSP ID
- The LSP's Sequence Number
- The LSP's checksum
- The LSP's Remaining Lifetime

When a router sends an LSP on a point-to-point subnetwork, it sets a timer for a period known as *minimumLSPTransmissionInterval*. If the timer expires before the router receives a PSNP acknowledging receipt of the LSP, a new LSP will be sent. On Cisco routers, the default minimumLSPTransmissionInterval is five seconds. It can be changed on a per interface basis with the command isis retransmit-interval. The default value is almost always the right value; the occasional exception is if a neighbor regularly struggles to process a large number of LSPs. In this case the router might not acknowledge an LSP fast enough, triggering a retransmit and just adding to its problems. In this case, increasing the minimumLSPTransmissionInterval can help; but in the long run, the real solution to such a problem is upgrading the low-powered neighbor.

On broadcast subnetworks, LSPs are not acknowledged by each receiving router. Instead, the DR periodically multicasts a CSNP that describes every LSP in the link-state database. The default CSNP period is 10 seconds, which can be changed with the command isis csnp-interval. L1 CSNPs are multicast to AllL1ISs (0180.c200.0014), and L2 CSNPs are multicast to AllL2ISs (0180.c200.0015).

When a router receives a CSNP, it compares the LSPs summarized in the PDU with the LSPs in its database. If the router has an LSP that is missing from the CSNP or a newer instance of an LSP, the router multicasts the LSP onto the network. If another router sends the updated LSP first, however, the router will not send another copy of the same LSP. If the router's database does not contain a copy of every LSP listed in the CSNP or if the database contains an older instance of an LSP, the router multicasts a PSNP listing the LSPs it needs. Although the PSNP is multicast,

only the DR responds with the appropriate LSPs.

An interesting feature of IS-IS is its ability to signal other routers if it runs out of memory and cannot record the complete link-state database. The memory overload might be due to an area that has been allowed to grow too large, a router with insufficient memory, or a transitory condition such as the failure of a DR. If a router cannot store the complete link-state database, it will set a bit in its LSP called the *Overload* (OL) bit.

The OL bit signals that the router might not be able to make correct routing decisions because of its incomplete database. The other routers still route packets to the overloaded router's directly connected networks, but do not use the router for transit traffic until it has issued an LSP with the OL bit cleared. Because a set OL bit prevents the router from being used as a hop along a route, the bit is frequently called the *hippity* bit (as in hippity-hop, believe it or not). Memory should be allocated equally to the L1 and the L2 database, but a router can be in an overload condition at one level and normal in the other.

This overload feature is an artifact of the days when routers did not have much memory. Modern routers are unlikely to become overloaded. However, the OL bit serves another very useful purpose in modern networks, particularly in BGP networks. To understand the problem, it is necessary to understand a few basic concepts of BGP. While link-state protocols such as IS-IS converge very quickly, BGP converges slowlysometimes taking several minutes. In the interior of a BGP network, routes have next-hop addresses that can be several router hops away. When a packet arrives, its destination is looked up in the BGP routes. The next hop of the BGP route must then be looked up in the IGP routes before the packet can be forwarded.

Now suppose that a new router is added to the network, and the IGP has converged but BGP has not yet finished. If another router determines that this new router is on the best path to a BGP next-hop, based on the converged IGP routes, it will forward packets to the router to be forwarded along to the next-hop address. But if BGP has not yet converged in the new router, the router might not have a BGP route for the packet and might drop it, resulting in black-holed traffic.

By setting the IS-IS OL bit until BGP has converged, you can avoid this problem. Other routers, if the OL bit is set, will route around the new router. Once BGP is converged, the OL bit can be cleared and packets can be forwarded through the new router.

Using the command `set-overload-bit on-startup`, you can specify a number of seconds by which the OL bit is set whenever IS-IS starts. When the specified seconds expire, the OL bit is automatically cleared. By setting the OL bit at around 300 to 500 seconds, you ensure that BGP will converge before the OL bit is cleared. There is an alternative command, `set-overload-bit on-startup wait-for-bgp`, which always clears the OL bit as soon as BGP converges. While more dynamic than using a static number of seconds, this option can have a vulnerability: If one BGP session does not come up for some reason, the OL bit will never be cleared. So it is usually best to specify seconds rather than the `wait-for-bgp` option.

You can also set the OL bit manually and indefinitely by using the `set-overload-bit` command with no other options specified. This can be useful in situations where you want the router to be attached to an IS-IS network and receive a full database, but you do not want the router to be used in any forwarding path. A lab router attached to a production network is an example of such an application.

The command `show isis database` displays a summary of the IS-IS link state database, as shown in Example 10-3. The router Brussels in this display is an L1/L2 router, so it has both L1 and L2 databases. The LSP ID shown in the first column consists of the System ID of the originating router, concatenated with two more octets. The first octet following the System ID is the Pseudonode ID. If this octet is nonzero, the LSP was originated by a DR. The System ID and the nonzero Pseudonode ID together make the LAN ID of a broadcast subnetwork.

### Example 10-3. The IS-IS database can be observed with the command *show isis database*.

```
Brussels#show isis database
IS-IS Level-1 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.0C04.DCC0.00-00  0x00000036   0x78AE        1152          0/0/0
0000.0C0A.2AA9.00-00  0x0000011B   0x057B        416           0/0/0
0000.0C76.5B7C.00-00* 0x00000150   0xD5D4        961           1/0/0
0000.0C76.5B7C.02-00* 0x00000119   0xD9C3        407           0/0/0
0000.0C76.5B7C.03-00* 0x000000FA   0x896E        847           0/0/0
IS-IS Level-2 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.0C0A.2AA9.00-00  0x0000013E   0x319A        666           0/0/0
0000.0C0A.2C51.00-00  0x00000133   0x762D        654           0/0/0
0000.0C76.5B7C.00-00* 0x0000014C    0x4E91       886           0/0/0
0000.0C76.5B7C.02-00* 0x0000011F   0x3CC3        1174          0/0/0
0000.3090.C7DF.00-00  0x0000011A   0xDDF0        858           0/0/0
Brussels#
```

The last octet is the LSP Number. Occasionally, an LSP can be so large that it exceeds the MTU that can be supported by the router buffers or the data link. In this case, the LSP is *fragmented*that is, the information is conveyed in multiple LSPs. The LSP IDs of these multiple LSPs consist of identical System IDs and Pseudonode IDs, and distinct LSP Numbers.

An asterisk following the LSP ID indicates that the LSP is originated by the router on which the database is being observed. For example, the database shown in Example 10-3 is taken from a router named Brussels. The L1 LSP with an ID of 0000.0c76.5bB7c.00-00 is originated by Brussels, as are four other LSPs.

The second and third columns of the database display show the Sequence Number and checksum of each LSP. The fourth column, LSP Holdtime, is the Remaining Lifetime of the LSP in seconds. If you enter the show isis database command repeatedly, you can observe the numbers decrementing. When the LSP is refreshed, the Remaining Lifetime is reset to 1200 seconds and the Sequence Number is incremented by one.

The last column indicates the state of the Attached (ATT) bit, Partition (P) bit, and Overload (OL) bit of each LSP. L2 and L1/L2 routers will set the ATT bit to one to indicate that they have a route to another area. The P bit indicates that the originating router has partition repair capability. Cisco (and most other vendors) does not support this function, so the bit is always set to zero. The OL bit is set to one if the originating router is experiencing a memory overload and has an incomplete link-state database, or if the OL bit has been manually set.

A complete LSP can be observed by using the command show isis database detail along with the level and the LSP ID, as shown in Example 10-4. The meanings of the individual fields of the LSP are given in the section "IS-IS PDU Formats."

## Example 10-4. The command *show isis database detail* is used to observe complete LSPs in the database.

```
London#show isis database detail level-2 0000.0C0A.2C51.00-00
IS-IS Level-2 LSP 0000.0C0A.2C51.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.0C0A.2C51.00-00* 0x0000013B   0x6635        815           0/0/0
  Area Address: 47.0001
  NLPID:      0x81 0xCC
  IP Address: 10.1.3.2
  Metric: 10 IS 0000.0C76.5B7C.02
  Metric: 10 IP 10.1.3.0 255.255.255.0
  Metric: 20 IP 10.1.2.0 255.255.255.0
  Metric: 10 IP 10.1.255.4 255.255.255.252
```

```
  Metric: 20 IP 10.1.255.0 255.255.255.0
  Metric: 30 IP 10.1.255.8 255.255.255.252
London#
```

## Decision Process

Once the Update process has built the link-state database, the Decision process uses the information in the database to calculate a shortest-path tree. The process then uses the shortest-path tree to construct a forwarding database (route table). Separate SPF calculations are run for the L1 database and the L2 database.

ISO 10589 specifies the following metrics (one required and three optional) to be used by IS-IS to calculate the shortest path:

- Default This metric must be supported and understood by every IS-IS router.
- Delay This optional metric reflects the transit delay of a subnetwork.
- Expense This optional metric reflects the monetary cost of using the subnetwork.
- Error This optional metric reflects the residual error probability of the subnetwork, similar to the IGRP/EIGRP reliability metric.

Each metric is expressed as an integer between 0 and 63, and a separate route is calculated for each metric. Therefore, if a system supports all four metrics, the SPF calculation must be run four times for both the L1 database and the L2 database. Because of the potential for many iterations of the SPF calculation for every destination, resulting in many different route tables, and because the optional metrics provide rudimentary Type of Service routing at best, Cisco supports only the default metric.

Cisco assigns a default metric of 10 to every interface, regardless of the interface type. The command isis metric changes the value of the default metric, and the default can be changed separately for level 1 and level 2. If the metrics are left at 10 for every interface, every subnetwork is considered equal and the IS-IS metric becomes a simple measure of hop count, where each hop has a cost of 10.

The total cost of a route is a simple sum of the individual metrics at each outgoing interface, and the maximum metric value that can be assigned to any route is 1023. This small maximum is frequently pointed out as a limitation of IS-IS because it leaves little room for metric granularity in large networks.

However, newer extensions to IS-IS allow for a much larger metric spacecalled wide metricsof 32 bits. To set wide metrics, the command metric-style wide is used.

IS-IS not only classifies routes as L1 or L2 but also classifies routes as *internal* or *external*. Internal routes are paths to destinations within the IS-IS routing domain, and external routes are paths to destinations external to the routing domain. Whereas L2 routes may be either internal or external, L1 routes are normally internal; using a routing policy you can make an L1 route external, but this should only be done with good justification and with special care.

Given multiple possible routes to a particular destination, an L1 path is preferred over an L2 path. Within each level, a path that supports the optional metrics is preferred over a path that supports only the default metric. (Again, Cisco supports only the default metric, so the second order of preference is not relevant to Cisco routers.) Within each level of metric support, the path with the lowest metric is preferred. If multiple equal-cost, equal-level paths are found by the Decision process, they are all entered into the route table. The Cisco IS-IS implementation usually performs equal-cost load balancing on up to six paths.

The previous section, "Update Process," discussed the last octet of the LSP ID, known as the LSP Number, used to track fragmented LSPs. The Decision process pays attention to the LSP Number

for several reasons. First, if an LSP with an LSP Number of zero and a nonzero Remaining Lifetime is not present in the database, the Decision process will not process any LSPs from the same system that have nonzero LSP Numbers. For example, if the LSP IDs 0000.0c76.5b7c.00-01 and 0000.0c76.5b7c.00-02 exist in the database, but the database contains no LSP with an LSP ID of 0000.0c76.5b7c.00-00, the first two LSPs are not processed. This approach ensures that incomplete LSPs do not cause inaccurate routing decisions.

Also, the Decision process will accept the following information only from LSPs whose LSP Number is zero:

- The setting of the Database Overload bit
- The setting of the IS Type field
- The setting of the Area Addresses option field

The Decision process ignores these settings in LSPs whose LSP Number is nonzero. In other words, the first LSP in a series of fragments speaks for all the fragments for these three settings.

Example 10-5 shows a route table from a Cisco IS-IS router. Notice that the L1 and L2 routes are indicated and that three destinations have multiple paths. A mask is associated with each route, indicating support for VLSM. Finally, the route table indicates that the administrative distance of IS-IS routes is 115.

## Example 10-5. This route table shows both level 1 and level 2 IS-IS routes.

```
Brussels#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
     10.0.0.0 is variably subnetted, 8 subnets, 3 masks
C       10.1.3.0 255.255.255.0 is directly connected, Ethernet0
i L2    10.1.2.0 255.255.255.0 [115/30] via 10.1.3.2, Ethernet0
                               [115/30] via 10.1.3.3, Ethernet0
i L1    10.1.5.0 255.255.255.0 [115/20] via 0.0.0.0, Serial0
                               [115/20] via 10.1.4.2, Ethernet1
C       10.1.4.0 255.255.255.0 is directly connected, Ethernet1
i L2    10.1.255.4 255.255.255.252 [115/20] via 10.1.3.2, Ethernet0
i L2    10.1.255.0 255.255.255.0 [115/30] via 10.1.3.2, Ethernet0
i L1    10.1.255.8 255.255.255.252 [115/20] via 10.1.3.3, Ethernet0
i L1    10.1.6.240 255.255.255.240 [115/20] via 0.0.0.0, Serial0
                                   [115/20] via 10.1.4.2, Ethernet1
Brussels#
```

Another duty of the Decision process in L1 routers is to calculate the path to the nearest L2 router, for inter-area routing. As previously discussed, when an L2 or L1/L2 router is attached to another area, the router will advertise the fact by setting the ATT bit in its LSP to one. The Decision process in L1 routers will choose the metrically closest L1/L2 router as the default inter-area router. When IS-IS is used to route IP, a default IP route to the L1/L2 is entered into the route table. For example, Example 10-6 shows a link-state database for an L1 router and its associated route table. LSP 0000.0c0a.2c51.00-00 has an ATT = 1. Based upon this information, the Decision process has chosen the router with System ID 0000.0c0a.2c51 as the default inter-area router. The route table shows a default route (0.0.0.0) via 10.1.255.6, with a metric of 10. Although it is not readily apparent from the information shown in Example 10-6, 10.1.255.6 and System ID 0000.0c0a.2c51 refer to the same router.

### Example 10-6. Integrated IS-IS adds a default IP route to the nearest L1/L2 router whose ATT bit is set to one.

```
Paris#show isis database
IS-IS Level-1 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.0C0A.2C51.00-00   0x0000016D   0xA093        730           1/0/0
0000.3090.6756.00-00*  0x00000167   0xC103        813           0/0/0
0000.3090.6756.04-00*  0x0000014E   0x227F        801           0/0/0
0000.3090.C7DF.00-00   0x00000158   0x78A6        442           0/0/0
Paris#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is 10.1.255.6 to network 0.0.0.0
     10.0.0.0 is variably subnetted, 5 subnets, 2 masks
i L1    10.1.3.0 255.255.255.0 [115/20] via 10.1.255.6, Serial0
C       10.1.2.0 255.255.255.0 is directly connected, TokenRing0
i L1    10.1.255.4 255.255.255.252 [115/20] via 10.1.255.6, Serial0
C       10.1.255.0 255.255.255.0 is directly connected, Serial0
i L1    10.1.255.8 255.255.255.252 [115/20] via 10.1.2.2, TokenRing0
i*L1 0.0.0.0 0.0.0.0 [115/10] via 10.1.255.6, Serial0
Paris#
```

The information in Example 10-6 illustrates an interesting characteristic of working with Integrated IS-IS, particularly when troubleshooting. Although TCP/IP is the protocol being routed, the protocol determining the routes, including all the route control packets and addresses, is a CLNS protocol. Sometimes correlating the CLNS information with the IP information can be difficult. One command that can be useful is which-route.

The command is used primarily to determine the route table in which a particular CLNS destination is located. However, which-route can also yield useful information about the IP addresses associated with a particular CLNS address. In Example 10-7, the System ID/Circuit ID 0000.0c0a.2c51.00, shown in the database of Example 10-6 as having ATT = 1, is used as the argument to which-route. The result shows, among other things, that the next-hop address for the queried System ID is 10.1.255.6.

### Example 10-7. The command *which-route* can reveal some information tying CLNS addresses to IP addresses.

```
Paris#which-route 0000.0C0A.2C51.00
Route look-up for destination 00.000c.0a2c.5100
  Using route to closest IS-IS level-2 router
Adjacency entry used:
System Id        SNPA      Interface   State   Holdtime   Type   Protocol
0000.0C0A.2C51   *HDLC*    Se0         Up      26         L1     IS-IS
  Area Address(es): 47.0001
  IP Address(es): 10.1.255.6
  Uptime: 22:08:52
Paris#
```

## IS-IS PDU Formats

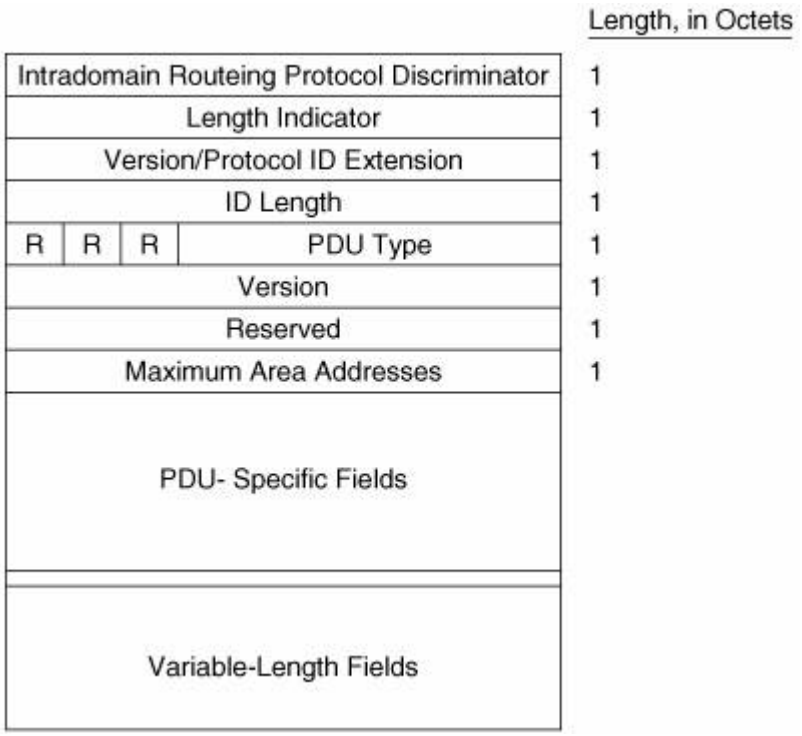
IS-IS uses nine PDU types in its processes, and each PDU is identified by a five-bit type number. The PDUs fall into three categories, as shown in Table 10-1.

Table 10-1. IS-IS PDU types.

| IS-IS PDU | Type Number |
|---|---|
| Hello PDUs | |
| Level 1 LAN IS-IS Hello PDU | 15 |
| Level 2 LAN IS-IS Hello PDU | 16 |
| Point-to-point IS-IS Hello PDU | 17 |
| Link State PDUs | |
| Level 1 LSP | 18 |
| Level 2 LSP | 20 |
| Sequence Numbers PDUs | |
| Level 1 CSNP | 24 |
| Level 2 CNSP | 25 |
| Level 1 PSNP | 26 |
| Level 2 PSNP | 27 |

The first eight octets of all of the IS-IS PDUs are header fields that are common to all PDU types, as shown in Figure 10-8. These first fields are described here, and the PDU-specific fields are described in subsequent sections.

Figure 10-8. The first eight octets of the IS-IS PDUs.




*Intradomain Routeing Protocol Discriminator* is a constant assigned by ISO 9577[17] to identify NPDUs. All IS-IS PDUs have a value of 0x83 in this field.

[17] International Organization for Standardization, "Protocol Identification in the Network Layer," ISO/IEC TR 9577, 1990.

*Length Indicator* specifies the length of the fixed header in octets.

*Version/Protocol ID Extension* is always set to one.

*ID Length* describes the length of the System ID field of NSAP addresses and NETs used in this routing domain. This field is set to one of the following values:

- An integer between 1 and 8 inclusive, indicating a System ID field of the same length in octets
- 0, indicating a System ID field of six octets
- 255, indicating a null System ID field (zero octets)

The System ID of Cisco routers must be six octets, so the ID Length field of a Cisco-originated PDU will always be zero.

*PDU Type* is a five-bit field containing one of the PDU type numbers shown in Table 10-1. The preceding three bits (R) are reserved and are always set to zero.

*Version* is always set to one, just like the Version/Protocol ID Extension in the third octet.

*Reserved* is always set to all zeroes.

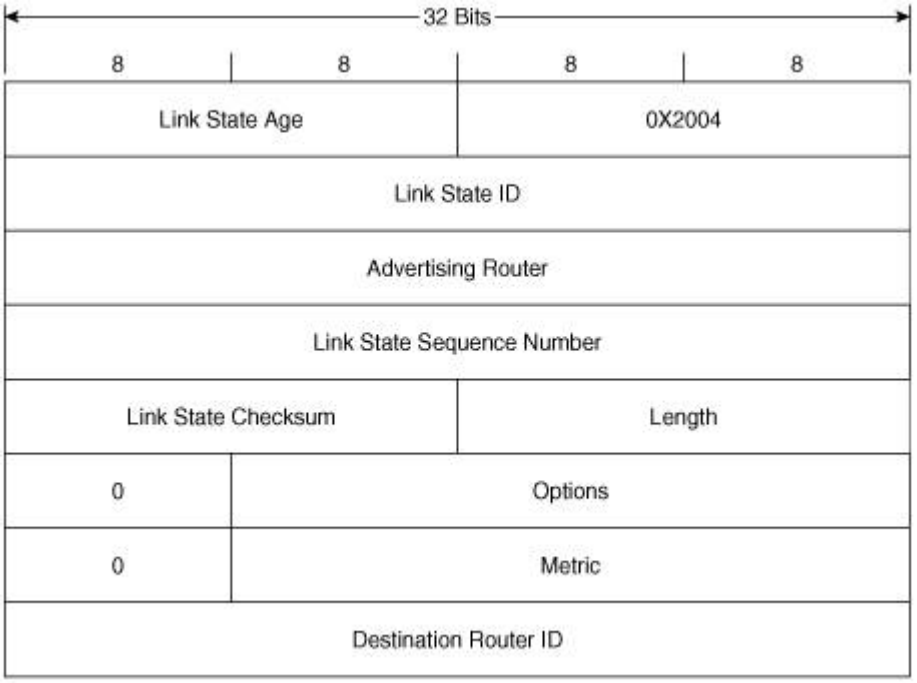
*Maximum Area Addresses* describes the number of area addresses permitted for this IS area. The number is set to one of the following values:

- An integer between 1 and 254 inclusive, indicating the number of areas allowed
- 0, indicating that the IS only supports a maximum of three addresses

Cisco IOS supports a maximum of three areas by default, so the Maximum Area Addresses field in IS-IS PDUs originated by Cisco routers will always be zero unless the default has been changed with the max-area-addresses command.

In Figure 10-9, an analyzer capture of an IS-IS PDU shows the first eight octets of the PDU.

Figure 10-9. This analyzer capture shows the first eight fields of an IS-IS PDU, along with its data-link header.




The PDU-specific fields following the common header fields are also part of the header; they vary according to PDU type and are discussed in the sections dealing with specific PDU types.

## TLV Fields

The variable-length fields following the PDU-specific fields are *Type/Length/Value* (TLV)[18] triplets, as shown in Figure 10-10. The Type (or Code)[19] is a number specifying the information content of the value field, the Length specifies the length of the Value field, and the Value field is the information itself. As the one-octet size of the Length field implies, the maximum size of the Value field is 255 octets.

[18] You are already familiar with the concept of TLVs from the coverage of EIGRP TLVs in Chapter 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)." In fact, RFC 1195 refers to Integrated IS-IS TLVs as TLVs.

[19] ISO uses the term "Code," while IETF uses the term "Type." Because "TLV" has come into more common usage, it is generally used in this chapter. This is a change from the first edition of this book, in which I chose to use the term "Code."

Figure 10-10. IS-IS TLV triplets perform the same function for IS-IS as TLV triplets perform for EIGRP.

| | Length, in Octets |
|---|---|
| Code | 1 |
| Length | 1 |
| Value | Length |

Table 10-2 lists the basic IS-IS TLV codes. The table also indicates whether the TLV is specified in ISO 10589 or in RFC 1195. The ISO-specified TLVs are designed for use with CLNP, although most of them are also used with IP. The RFC-specified TLVs are designed only for IP. If a router does not recognize a particular TLV code, it will ignore the TLV. This arrangement allows TLVs for CLNP, IP, or both to be carried in the same PDU.

Table 10-2. TLV codes used with IS-IS.

| Type | TLV | ISO 10589 | RFC 1195 |
|---|---|---|---|
| 1 | Area Addresses | X | |
| 2 | IS Neighbors (LSPs) | X | |
| 3 | ES Neighbors[*] | X | |
| 4 | Partition Designated level 2 IS[**] | X | |
| 5 | Prefix Neighbors | X | |
| 6 | IS Neighbors (Hellos) | X | |
| 8 | Padding | X | |
| 9 | LSP Entries | X | |
| 10 | Authentication Information | X | |
| 14 | LSP Buffersize | X | |
| 128 | IP Internal Reachability Information | | X |
| 129 | Protocols Supported | | X |
| 130 | IP External Reachability Information | | X |
| 131 | Inter-Domain Routing Protocol Information | | X |
| 132 | IP Interface Address | | X |
| 133 | Authentication Information[***] | | X |

[*] The ES-Neighbors and Prefix Neighbors TLVs are not relevant to IP routing and are not covered in this book.

[**] This TLV is used for partition repair, which Cisco does not support.

[***] RFC 1195 specifies this code for IP authentication, but most IS-IS implementations, including IOS, use the ISO code of 10.

Although many of the TLVs are used by more than one IS-IS PDU type, only one (Authentication) is used by all PDUs. As the formats of the IS-IS PDUs are described in the following sections, the

TLVs used by each PDU are listed. The format of each TLV is described only once, the first time it is listed. Table 10-3 summarizes which TLVs are used with which PDUs.

Table 10-3. The TLVs used by each IS-IS PDU.

| | PDU Type | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| TLV Type | 15 | 16 | 17 | 18 | 20 | 24 | 25 | 26 | 27 |
| Area addresses | X | X | X | X | X | | | | |
| IS Neighbors (LSPs) | | | | X | X | | | | |
| ES Neighbors | | | | X | | | | | |
| Partition Designated Level 2 IS | | | | | X | | | | |
| Prefix Neighbors | | | | | X | | | | |
| IS Neighbors (Hellos) | X | X | | | | | | | |
| Padding | X | X | X | | | | | | |
| LSP Entries | | | | | | X | X | X | X |
| Authentication Information | X | X | X | X | X | X | X | X | X |
| IP Internal Reachability Information | | | | X | X | | | | |
| Protocols Supported | X | X | X | X | X | | | | |
| IP External Reachability Information | | | | X | X | | | | |
| Inter-Domain Routing Protocol Information | | | | | X | | | | |
| IP Interface Address | X | X | X | X | X | | | | |

The great advantage of using TLVs is that to add new capabilities to IS-IS, you need only add new TLV types. Since it was first specified, there have been many enhancements to IS-IS. Table 10-4 lists many of the TLVs that have been added since ISO 10589 and RFC 1195 first specified IS-IS. Also listed are the RFCs specifying the new TLVs and what they are used for. In some cases, the extensions are new enough that at this writing they are still in Internet-Draft form. By the time you are reading this chapter, they might be RFCs; check the IETF website (www.ietf.org) Linked Varified to find out. Some of the extensions listed in Table 10-4 are detailed in subsequent sections of this chapter, and some are not.

Table 10-4. TLVs used for extending IS-IS capabilities.

| Type | TLV | RFC | Description |
|---|---|---|---|
| 12 | Optional Checksum | 3358 | Adds checksum capability to SNPs |
| 22 | Extended IS Reachability | 3784 | Adds Traffic Engineering capabilities, replaces type 2 TLVs |
| 134 | Traffic Engineering Router ID | 3784 | Adds Traffic Engineering capability |

| Type | TLV | RFC | Description |
|---|---|---|---|
| 135 | Extended IP Reachability | 3784 | Adds Traffic Engineering and wide metrics capability, replaces types 128 and 130 TLVs |
| 137 | Dynamic Hostname | 2763 | Adds the ability for nodes to be identified by hostname rather than SysID in commands such as show isis database |
| 211 | Restart | 3847 | Adds Graceful Restart capability |
| 222 | MT Intermediate Systems | Draft | Used with type 22 TLVs for multitopology support |
| 229 | Multi-Topology | Draft | Adds multitopology support |
| 232 | IPv6 Interface Address | Draft | Equivalent to type 132 TLVs, for IPv6 support |
| 235 | MT Reachable IPv4 Prefixes | Draft | Used with type 135 TLV for multitopology support |
| 236 | IPv6 Reachability | Draft | Equivalent to types 128 and 130, for IPv6 support |
| 237 | MT Reachable IPv6 Prefixes | Draft | Used with type 236 TLVs for multitopology support |
| 240 | Point-to-Point Three-Way Adjacency | 3373 | Adds 3-way handshaking capability |
| 250 | Experimental | Draft | Used for experimental extensions |

## IS-IS Hello PDU Format

The purpose of the IS-IS Hello PDU is to allow IS-IS routers to discover IS-IS neighbors on a link. Once the neighbors have been discovered and have become adjacent, the job of the Hello PDU is to act as a keepalive to maintain the adjacency and to inform the neighbors of any changes in the adjacency parameters.

The upper limit on the size of an IS-IS PDU is defined by either the buffer size of the originating router or the MTU of the data link on which the PDU is transmitted. ISO 10589 specifies that IS-IS Hellos must be padded, using type 8 Padding TLVs, to within one octet less than this maximum, partly to allow a router to implicitly communicate its MTU to its neighbors. More important, sending Hellos at or near the link MTU is intended to help detect link failure modes in which small PDUs pass but larger PDUs are dropped. If a neighboring interface does not support the minimum required MTU, the padded Hellos are dropped and no adjacency is established.

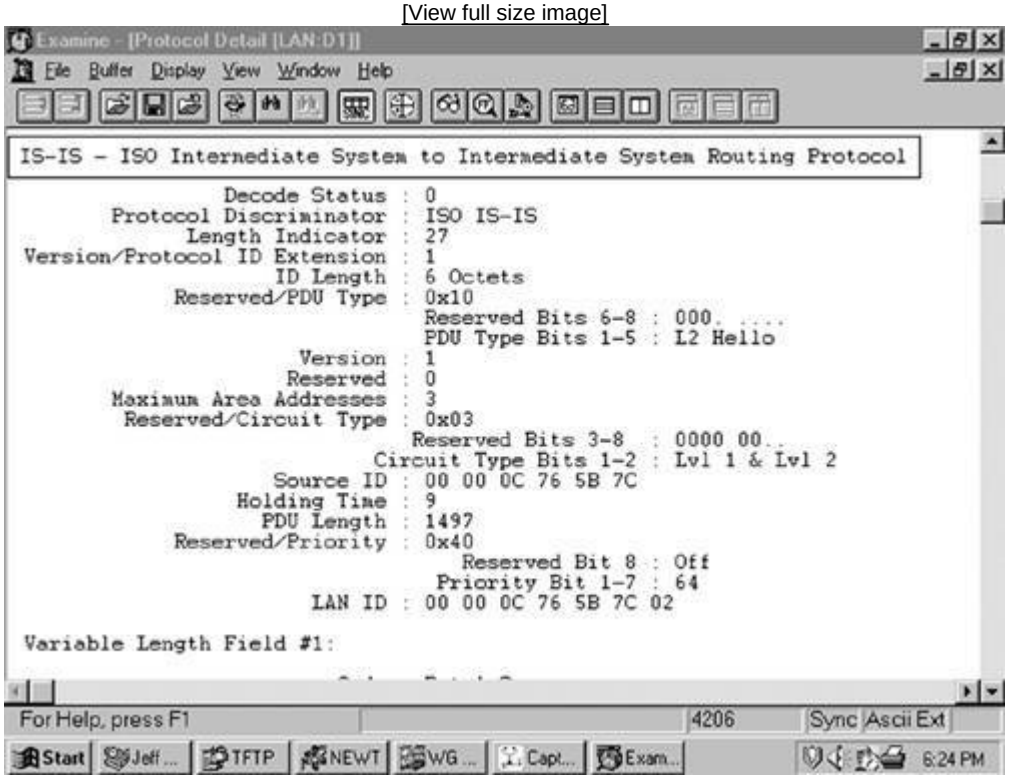
There are two kinds of IS-IS Hellos: LAN Hellos and point-to-point Hellos. The LAN Hellos can be further divided into L1 and L2 LAN Hellos. The format of the two types of LAN Hellos is identical, as shown in Figure 10-11. Figure 10-12 shows an analyzer capture of a level 2 LAN Hello.

Figure 10-11. The IS-IS LAN Hello PDU format.

| | | | | | | | Length, in Octets |
|---|---|---|---|---|---|---|---|
| Intradomain Routeing Protocol Discriminator | | | | | | | 1 |
| Length Indicator | | | | | | | 1 |
| Version/Protocol ID Extension | | | | | | | 1 |
| ID Length | | | | | | | 1 |
| R | R | R | PDU Type | | | | 1 |
| Version | | | | | | | 1 |
| Reserved | | | | | | | 1 |
| Maximum Area Addresses | | | | | | | 1 |
| R | R | R | R | R | R | Circuit Type | 1 |
| Source ID | | | | | | | ID Length |
| Holding Time | | | | | | | 2 |
| PDU Length | | | | | | | 2 |
| R | Priority | | | | | | 2 |
| LAN ID | | | | | | | ID Length + 1 |
| Variable-Length Fields | | | | | | | |

Figure 10-12. This analyzer capture of a LAN Hello shows the fields unique to a Hello PDU.

[View full size image]




*Circuit Type* is a two-bit field (the preceding six bits are reserved and are always zero) specifying

whether the router is an L1 (01), L2 (10), or L1/L2 (11). If both bits are zero (00), the entire PDU is ignored.

*Source ID* is the System ID of the router that originated the Hello.

*Holding Time* is the period a neighbor should wait to hear the next Hello before declaring the originating router dead.

*PDU Length* is the length of the entire PDU in octets.

*Priority* is a seven-bit field used for the election of a DR. The field carries a value between 0 and 127 with the higher number having the higher priority. L1 DRs are elected by the priority in L1 LAN Hellos, and L2 DRs are elected by the priority in L2 LAN Hellos.

*LAN ID* is the System ID of the DR plus one more octet (the Pseudonode ID) to differentiate this LAN ID from another LAN ID that might have the same DR.

The following TLVs can be used by an IS-IS LAN Hello:[20]

[20] As a reminder, RFC 1195 also specifies an Authentication Information TLV with a type number of 133. Cisco uses the ISO-specified type number of 10 to identify its Authentication Information TLVs.

- Area Addresses (type 1)
- Intermediate System Neighbors (type 6)
- Padding (type 8)
- Authentication Information (type 10)
- Optional Checksum (type 12)
- Protocols Supported (type 129)
- IP Interface Address (type 132)
- Restart (type 211)
- Multi-Topology (type 229)
- IPv6 Interface Address (type 232)
- Experimental (type 250)

Figure 10-13 shows the format of the IS-IS point-to-point Hello PDU. It is almost identical to the LAN Hello, except that there is no Priority field and there is a Local Circuit ID field instead of a LAN ID field. Unlike LAN Hellos, L1 and L2 information is carried in the same point-to-point Hello PDU. The point-to-point Hello carries all of the same TLVs, with the exception of the Intermediate System Neighbors TLV. Instead, if the IS-IS implementation supports the associated extension, the point-to-point Hello carries a P2P Three-Way Adjacency (type 240) TLV. The use of this TLV is described in the later section "3-Way Handshaking."

Figure 10-13. The IS-IS point-to-point Hello PDU.

| | Length, in Octets |
|---|---|
| Intradomain Routeing Protocol Discriminator | 1 |
| Length Indicator | 1 |
| Version/Protocol ID Extension | 1 |
| ID Length | 1 |
| R R R PDU Type | 1 |
| Version | 1 |
| Reserved | 1 |
| Maximum Area Addresses | 1 |
| R R R R R R Circuit Type | 1 |
| Source ID | ID Length |
| Holding Time | 2 |
| PDU Length | 2 |
| Local Circuit ID | 1 |
| | |
| Variable-Length Fields | |

*Local Circuit ID* is a one-octet ID assigned to this circuit by the router originating the Hello and is unique among the router's interfaces. The Local Circuit ID in the Hellos at the other end of the point-to-point link might or might not contain the same value.

## Area Addresses TLV

The Area Addresses TLV (Figure 10-14) is used to advertise the area addresses configured on the originating router. As the multiple Address Length/Area Address fields imply, a router can be configured with multiple area addresses.

Figure 10-14. The Area Addresses TLV.

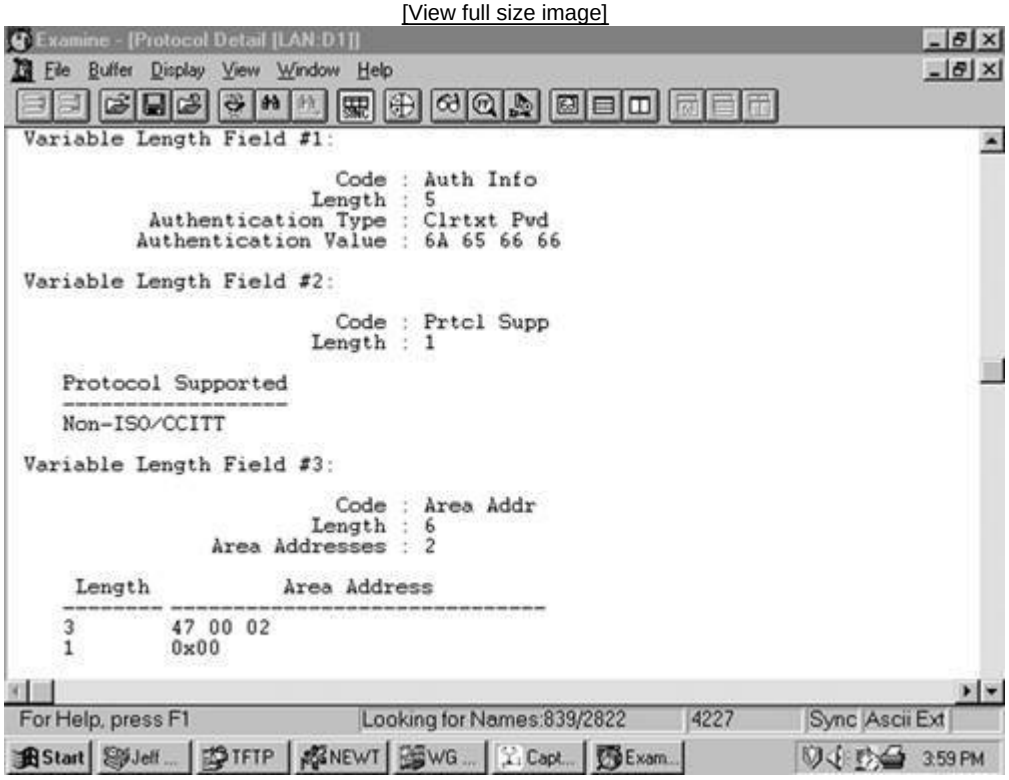
| | Length, in Octets |
|---|---|
| Code = 1 | 1 |
| Length | 1 |
| Address Length | 1 |
| Area Address | Address Length |
| Multiple Fields | |
| Address Length | 1 |
| Area Address | Address Length |

Figure 10-15 shows part of an IS-IS Hello PDU; Variable Length Field #3 is an Area Addresses

TLV with a total length of six octets. There are two area addresses: 47.0002 (three octets), and 0 (one octet).

Figure 10-15. An Area Addresses TLV is shown as Variable Length Field #3. The analyzer also indicates the total addresses listed in the TLV (2) for convenience, but this information is not one of the TLV fields.

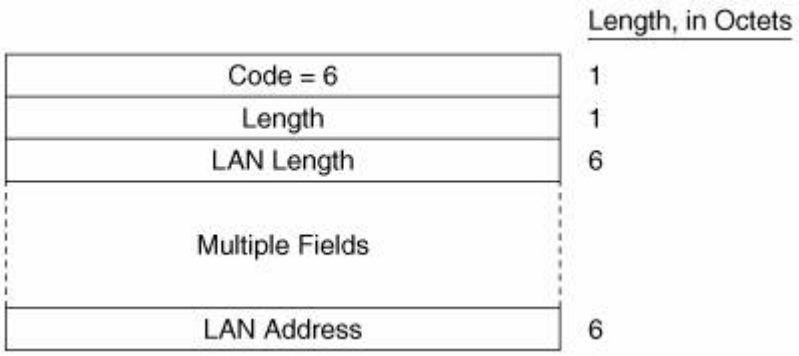
[View full size image]




## Intermediate System Neighbors TLV (Hello)

The IS Neighbors TLV (Figure 10-16) lists the System IDs of all neighbors from whom a Hello has been heard within the last Hold Time. Note that this TLV gives the IS-IS LAN Hello a function similar to the OSPF function of listing all recently-heard-from neighbors, to verify two-way communication.

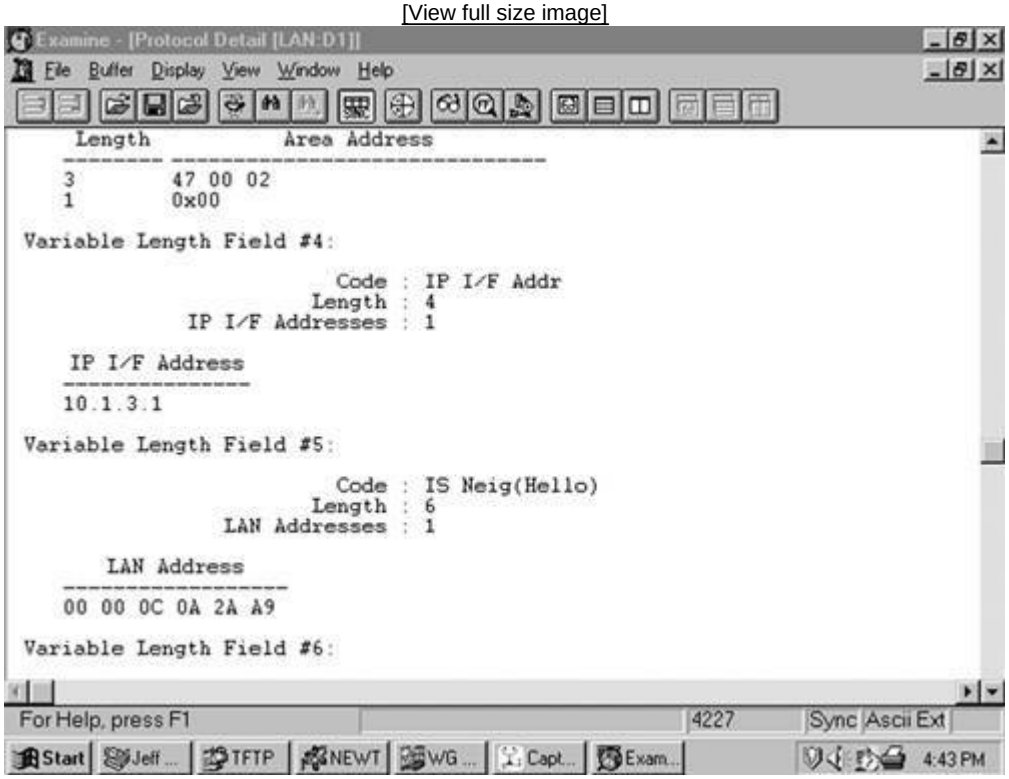
Figure 10-16. The IS Neighbors TLV for Hello PDUs.



This TLV is used only in LAN Hellos; it is not carried in point-to-point Hellos, where no DR election is performed. It is also different from the IS Neighbors TLV used by LSPs, and the two are distinguished by their type numbers. L1 LAN Hellos list L1 neighbors only, and L2 LAN Hellos list L2 neighbors. The neighbors listed in this TLV are identified by the MAC address of their interfaces on the LAN. Variable Length Field #5 in Figure 10-17 shows an IS Neighbors TLV in which a single neighbor, 0000.0c0a.2aa9, is listed.

Figure 10-17. An IS Neighbors TLV is shown as Variable Length Field #5.

[View full size image]

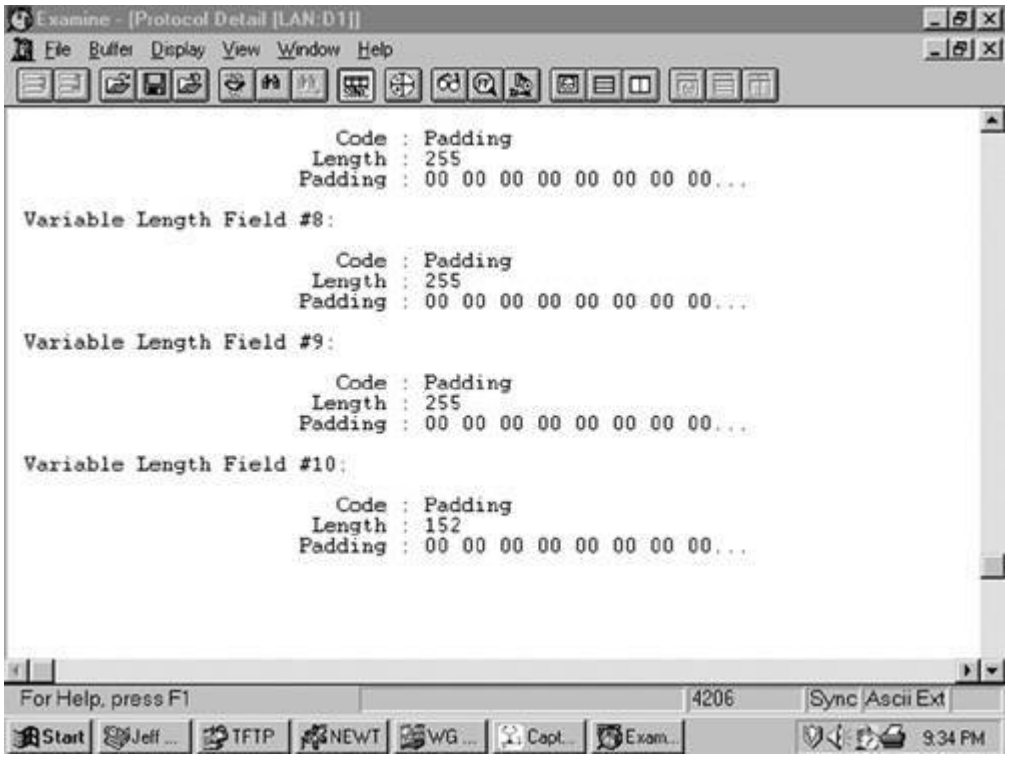



## Padding TLV

The Padding TLV is used to pad a Hello PDU out to either the minimum allowed IS-IS size of 1492 bytes or the MTU of the link. Since the maximum size of a Value field is 255 octets, multiple Padding TLVs are often used. The bits in the Value field can be arbitrary because they are ignored; Cisco sets all these bits to zero (Figure 10-18).

Figure 10-18. The last Variable Length Fields of the Hello PDU shown in Figure 10-12 are Padding TLVs, which bring the size of the PDU to the 1497-octet length shown in Figure 10-12. With the addition of the 3-octet LLC header and the 18-octet Ethernet header, the entire frame size is the 1518-octet Ethernet MTU.

[View full size image]




## Authentication Information TLV

The Authentication Information TLV (Figure 10-19) is used when authentication is configured. The Authentication Type field contains a number between 0 and 255 that specifies the type of authentication used and hence the type of information contained in the Authentication Value field. IOS supports either clear-text password authentication or, HMAC-MD5 encrypted hash authentication. Clear-text passwords are Authentication Type 1, and HMAC-MD5 is Authentication Type 54.

Figure 10-19. The Authentication Information TLV.

| | Length, in Octets |
|---|---|
| Code = 10 | 1 |
| Length | 1 |
| Authentication Type | 1 |
| Authentication Value | Length -1 |

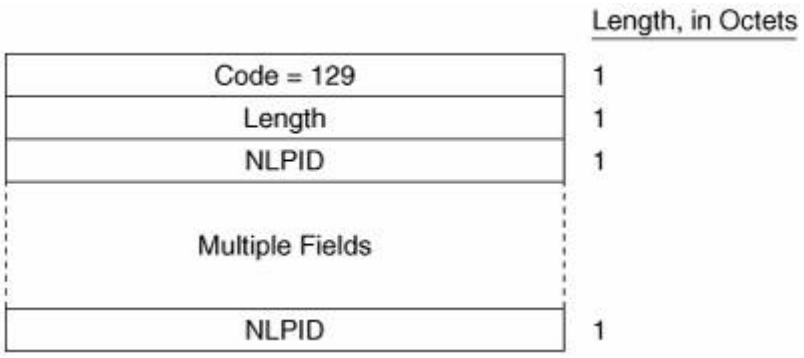
Variable Length Field #1 in Figure 10-15 is an Authentication Information TLV. The password is "jeff," which is displayed as the hex representation of its ASCII characters.

## Protocols Supported TLV

The Protocols Supported TLV (Figure 10-20) is specified in RFC 1195, and its original purpose was to indicate whether the originator of the PDU supports CLNP only, IP only, or both. Since

that time, the TLV is also used to indicate support for IPv6. The IPv4 NLPID is 204 (0xCC) and the IPv6 NLPID is 142 (0x8E). For each protocol supported, the corresponding one-octet Network Layer Protocol Identifier (NLPID), specified in ISO/TR 9577, is listed in the TLV. IP is 0x81. Variable Length Field #2 in Figure 10-15 is a Protocols Supported TLV.

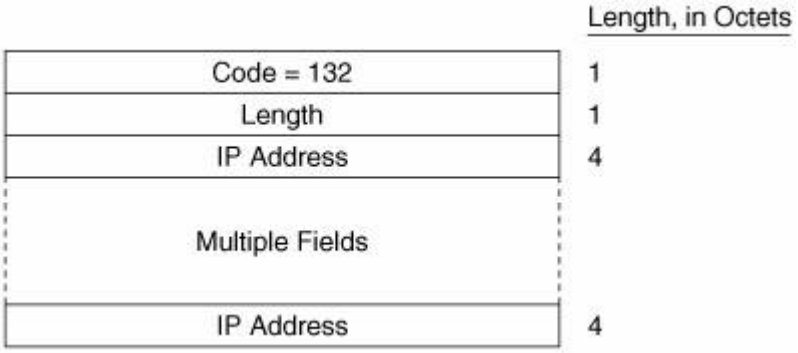
Figure 10-20. The Protocols Supported TLV.




## IP Interface Address TLV

The IP Interface Address TLV (Figure 10-21) contains the IP address or addresses of the interface out which the PDU was sent. Because the Length field is one octet, an interface of an IS-IS router can theoretically have up to 63 IP addresses. Variable Length Field #4 in Figure 10-17 is an IP Interface Address TLV, indicating that the captured Hello PDU was transmitted from an interface with an address of 10.1.3.1.

Figure 10-21. The IP Interface Address TLV.

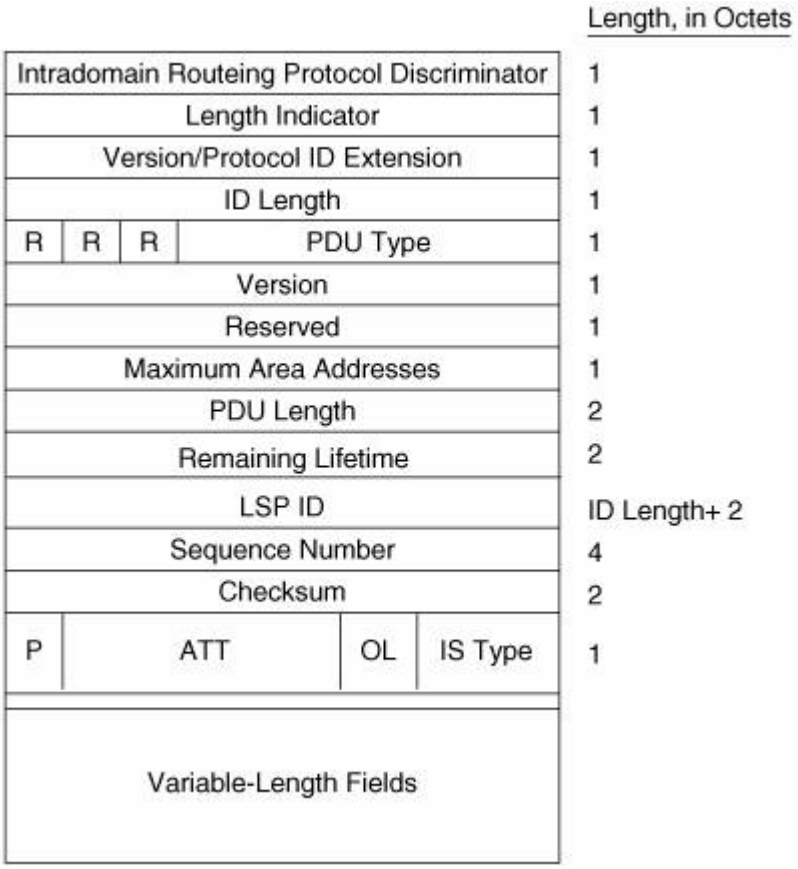



## IS-IS Link State PDU Format

The IS-IS LSP performs essentially the same function as the OSPF LSA. An L1 router floods an L1 LSP throughout an area to identify its adjacencies and the state of those adjacencies. An L2 router floods L2 LSPs throughout the level 2 domain, identifying adjacencies to other L2 routers and address prefixes that the advertising L2 router can reach.

Figure 10-22 shows the format of the IS-IS LSP. The format is the same for both L1 LSPs and L2 LSPs.

Figure 10-22. The IS-IS LSP format.




*PDU Length* is the length of the entire PDU in octets.

*Remaining Lifetime* is the number of seconds before the LSP is considered to be expired.

*LSP ID* is the System ID, the Pseudonode ID, and the LSP Number of the LSP. The LSP ID is described in more detail in section "Update Process."

*Sequence Number* is a 32-bit unsigned integer.

*Checksum* is the checksum of the contents of the LSP.

*P* is the Partition Repair bit. Although the bit exists in both L1 and L2 LSPs, it is relevant only in L2 LSPs. When this bit is set, it indicates that the originating router supports the automatic repair of area partitions. Cisco IOS does not support this function, so the P bit of LSPs originated by Cisco routers will always be zero.

*ATT* is a four-bit field indicating that the originating router is attached to one or more other areas. Although the bits exist in both L1 and L2 LSPs, they are relevant only in L1 LSPs that have been originated by L1/L2 routers. The four bits indicate which metrics are supported by the attachment. Reading from left to right, the bits indicate

- Bit 7: The Error metric
- Bit 6: The Expense metric
- Bit 5: The Delay metric

- Bit 4: The Default metric

Cisco IOS supports only the default metric, so bits 5 through 7 will always be zero.

*OL* is the Link State Database Overload bit. Under normal circumstances, this bit will be zero. Routers receiving an LSP with the OL bit set will not use the originating router as a transit router, although they will still route to destinations on the originator's directly connected links.

*IS Type* is a two-bit field indicating whether the originating router is an L1 or an L2:

- 00 = Unused value
- 01 = L1
- 10 = Unused value
- 11 = L2

An L1/L2 router sets the bits according to whether the LSP is an L1 or an L2 LSP.

The following TLVs can be used by an L1 LSP:

- Area Addresses (type 1)
- IS Neighbors (type 2)
- ES Neighbors (type 3)
- Authentication Information (type 10)
- LSP Buffer Size (type 14)
- Extended IS Reachability (type 22)
- IP Internal Reachability Information (type 128)
- Protocols Supported (type 129)
- IP External Reachability Information (type 130)
- Traffic Engineering Router ID (type 134)
- Extended IP Reachability (type 135)
- Dynamic Hostname (type 137)
- MT Intermediate Systems (type 222)
- Multi-Topology (type 229)
- IPv6 Interface Address (type 232)
- MT Reachable IPv4 Prefixes (type 235)
- IPv6 IP Reachability (type 236)
- MT Reachable IPv6 Prefixes (type 237)
- Experimental (type 250)

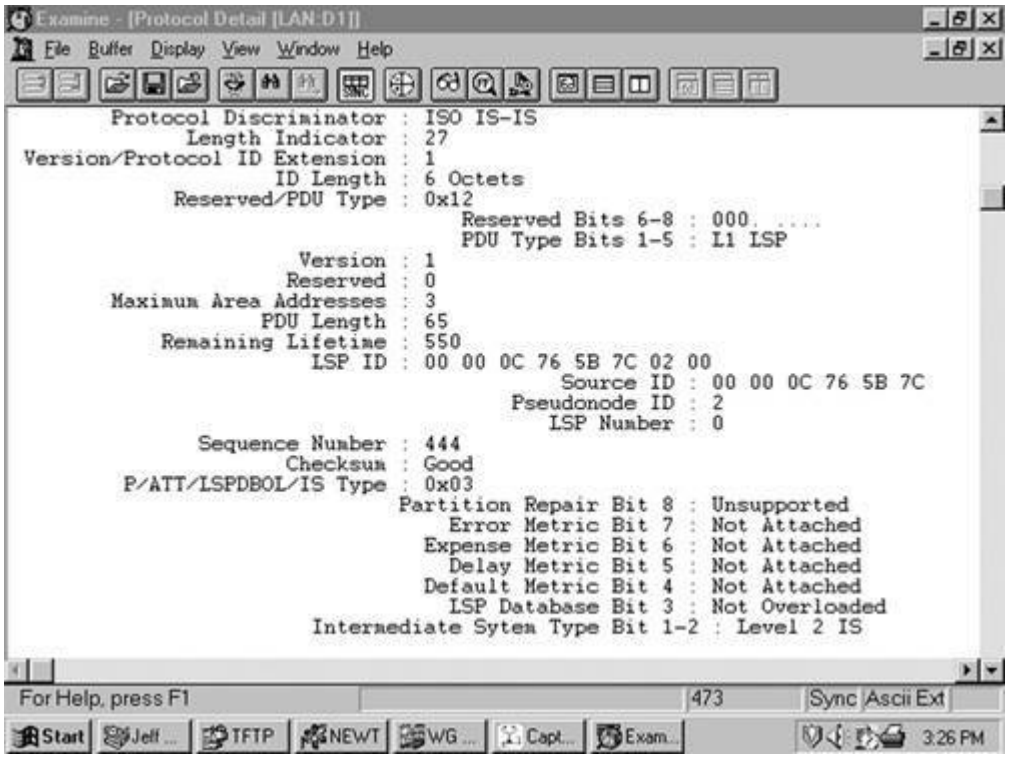
The following TLVs can be used by an L2 LSP:

- Area Addresses (type 1)
- IS Neighbors (type 2)
- Partition Designated Level 2 IS (type 4)
- Prefix Neighbors (type 5)
- Authentication Information (type 10)
- LSP Buffer Size (type 14)
- Extended IS Reachability (type 22)
- IP Internal Reachability Information (type 128)
- IP External Reachability Information (type 130)
- Inter-Domain Routing Protocol Information (type 131)
- Protocols Supported (type 129)
- IP Interface Address (type 132)
- Traffic Engineering Router ID (type 134)
- Extended IP Reachability (type 135)
- Dynamic Hostname (type 137)
- MT Intermediate Systems (type 222)
- Multi-Topology (type 229)
- IPv6 Interface Address (type 232)
- MT Reachable IPv4 Prefixes (type 235)
- IPv6 IP Reachability (type 236)
- MT Reachable IPv6 Prefixes (type 237)
- Experimental (type 250)

Figure 10-23 shows an L1 LSP that was originated by an L1/L2 router.

Figure 10-23. Analyzer capture of an LSP.

[View full size image]




## Intermediate System Neighbors TLV (LSP)

The Intermediate System Neighbors TLV used by LSPs (Figure 10-24) lists the originating router's IS-IS neighbors (including pseudonodes) and the metrics of the router's link to each of its neighbors.

Figure 10-24. Intermediate System Neighbors TLV for LSPs.

| | | | Length, in Octets |
|---|---|---|---|
| Code = 2 | | | 1 |
| Length | | | 1 |
| Virtual Flag | | | 1 |
| R | I/E | Default Metric | 1 |
| S | I/E | Delay Metric | 1 |
| S | I/E | Expense Metric | 1 |
| S | I/E | Error Metric | 1 |
| Neighbor ID | | | ID Length + 1 |
| Multiple Fields | | | |
| R | I/E | Default Metric | 1 |
| S | I/E | Delay Metric | 1 |
| S | I/E | Expense Metric | 1 |
| S | I/E | Error Metric | 1 |
| Neighbor ID | | | ID Length + 1 |

*Virtual Flag*, although eight bits long, has a value of either 0x01 or 0x00. A 0x01 in this field indicates that the link is a level 2 virtual link to repair an area partition. The field is relevant only to L2 routers that support area partition repair; Cisco does not, so the field will always be 0x00 in Cisco-originated LSPs.

*R* is a reserved bit and is always zero.

*I/E*, associated with each of the metrics, indicates whether the associated metric is internal or external. The bit has no meaning in IS Neighbors TLVs because all neighbors are by definition internal to the IS-IS domain. Therefore, this bit is always zero in IS Neighbors TLVs.

*Default Metric* is the six-bit default metric for the originating router's link to the listed neighbor and contains a value between 0 and 63.

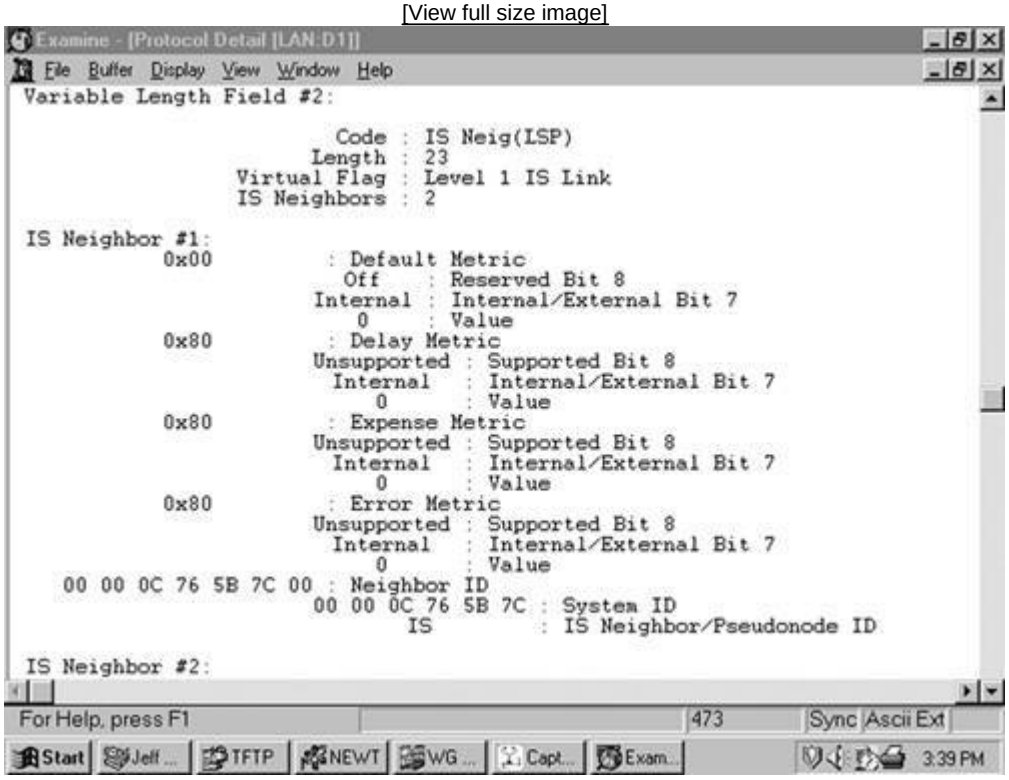
*S*, associated with each of the optional metrics, indicates whether the metric is supported (zero) or unsupported (one). Cisco does not support any of the three optional metrics, so the bit is always set to one and the associated six-bit metric fields are all zeroes.

*Neighbor ID* is the System ID of the neighbor, plus one more octet. If the neighbor is a router, the last octet is 0x00. If the neighbor is a pseudonode, the System ID is that of the DR and the last octet is the Pseudonode ID.

Figure 10-25 shows part of an IS Neighbors TLV.

Figure 10-25. Part of an IS Neighbors TLV in an LSP.

[View full size image]




## IP Internal Reachability Information TLV

The IP Internal Reachability Information TLV (Figure 10-26) lists IP addresses and associated

masks within the routing domain that are directly connected to the advertising router. The TLV is used by both L1 and L2 LSPs, but never appears in an LSP describing a pseudonode. The metric fields are identical to the IS Neighbors TLV, except that no I/E bit is associated with the optional metrics. Instead, the bit is reserved and is always zero. Like the IS Neighbors TLV, the I/E bit in this TLV is always zero because the addresses advertised in the TLV are always internal. Figure 10-27 shows an analyzer capture of an IP Internal Reachability Information TLV.

Figure 10-26. IP Internal Reachability Information TLV.

| | | | Length, in Octets |
|---|---|---|---|
| Code = 128 | | | 1 |
| Length | | | 1 |
| R | I/E | Default Metric | 1 |
| S | R | Delay Metric | 1 |
| S | R | Expense Metric | 1 |
| S | R | Error Metric | 1 |
| IP Address | | | 4 |
| Subnet Mask | | | 4 |
| Multiple Fields | | | |
| R | I/E | Default Metric | 1 |
| S | R | Delay Metric | 1 |
| S | R | Expense Metric | 1 |
| S | R | Error Metric | 1 |
| IP Address | | | 4 |
| Subnet Mask | | | 4 |

Figure 10-27. Analyzer capture of an IP Internal Reachability Information TLV.

[View full size image]

```
Variable Length Field #4:

                        Code : Intn Rch Info
                      Length : 24
Internal Reachability Information : 2

Internal Reachability Info #1:
        0x0A      : Default Metric
                  Off     : Reserved Bit 8
                Internal  : Internal/External Bit 7
                  10      : Value
        0x80      : Delay Metric
                Unsupported : Supported Bit 8
                    Off     : Reserved Bit 7
                     0      : Value
        0x80      : Expense Metric
                Unsupported : Supported Bit 8
                    Off     : Reserved Bit 7
                     0      : Value
        0x80      : Error Metric
                Unsupported : Supported Bit 8
                    Off     : Reserved Bit 7
                     0      : Value
      10.1.3.0    : IP Address
   255.255.255.0  : Subnet Mask

Internal Reachability Info #2:
        0x0A      : Default Metric
                  Off     : Reserved Bit 8
```

## IP External Reachability Information TLV

The IP External Reachability Information TLV lists IP addresses and associated masks external to the routing domain, which can be reached via one of the originating router's interfaces. Its format is identical to the Internal Reachability Information TLV shown in Figure 10-26 except for the code, which is 130. The I/E bit determines the metric type for all four metricsI/E = 0 for internal metrics and I/E = 1 for external metrics.

## Inter-Domain Routing Protocol Information TLV

The Inter-Domain Routing Protocol Information TLV (Figure 10-28) allows L2 LSPs to transparently carry information from external routing protocols through the IS-IS domain. The TLV serves the same purpose as the Route Tag fields of RIPv2, EIGRP, and OSPF packets. Route tagging is covered in Chapter 14, "Route Maps."

Figure 10-28. The Inter-Domain Routing Protocol Information TLV.

| | Length, in Octets |
|---|---|
| Code = 131 | 1 |
| Length | 1 |
| Inter-Domain Information Type | 1 |
| External Information | Variable |

*Inter-Domain Information Type* specifies the type of information contained in the variable-length External Information field. If the type field is 0x01, the External Information is of a format used by the local interdomain routing protocol. Chapter 14 includes examples of using route maps to set such local information. If the type field is 0x02, the External Information is a 16-bit

autonomous system number, which is used to tag all subsequent External IP Reachability entries until either the end of the LSP or the next occurrence of the Inter-Domain Routing Protocol Information TLV.

## IS-IS Sequence Numbers PDU Format

SNPs are used to maintain the IS-IS link-state database by describing some or all of the LSPs in the database. They are also used, in some cases, to request LSPs and to implicitly or explicitly acknowledge received LSPs. So SNPs have a function similar to OSPF Link State Request, Database Description, and Link State Acknowledgment messages.

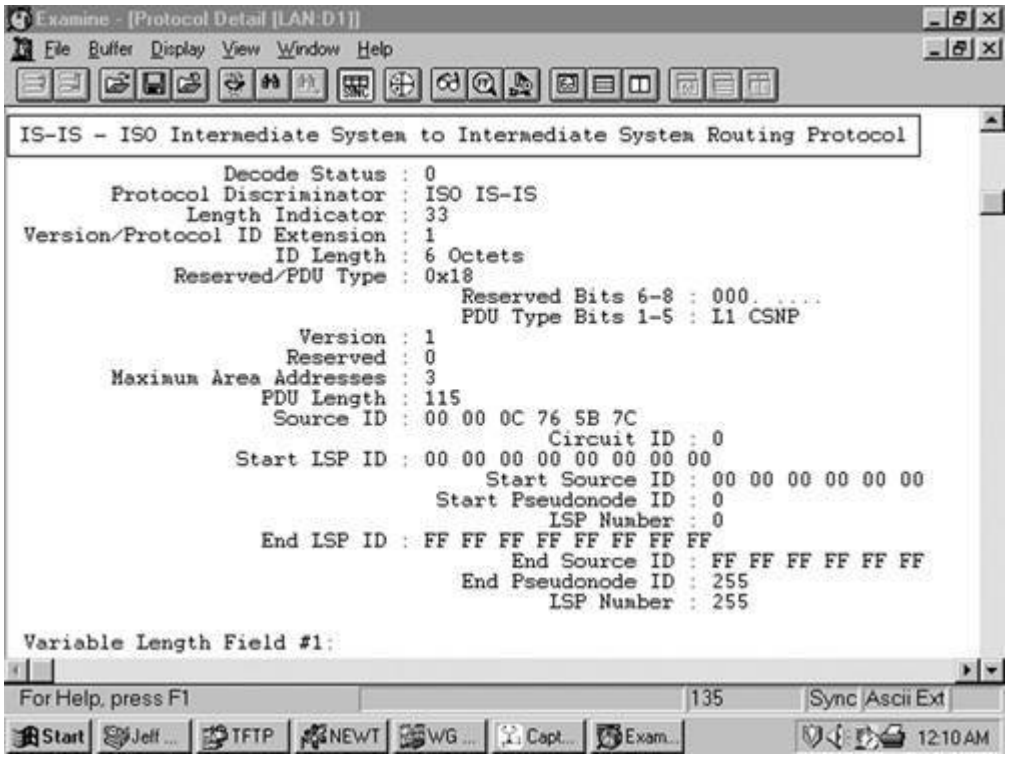
A DR periodically multicasts a CSNP (Figure 10-29) to describe all the LSPs in the pseudonode's database. Because there is an L1 database and an L2 database, CSNPs are also either L1 or L2. Some link-state databases can be so large that the LSPs cannot all be described in a single CSNP. For this reason, the last two fields of the CSNP header are the *Start LSP ID* field and the *End LSP ID* field, which together describe the range of LSPs described in the CSNP. Figure 10-30 shows how these two fields are used. In this CSNP, the full database is described; therefore, the LSP ID range starts with 0000.0000.0000.00.00 and ends with ffff.ffff.ffff.ff.ff. If two CSNPs were required to describe the database, the range of the first CSNP might be something like 0000.0000.0000.00 to 0000.0c0a.1234.00.00 and the range of the second CSNP might be 0000.0c0a.1234.00.01 to ffff.ffff.ffff.ff.ff.

Figure 10-29. The IS-IS CSNP format.

| | | | | Length, in Octets |
|---|---|---|---|---|
| Intradomain Routeing Protocol Discriminator | | | | 1 |
| Length Indicator | | | | 1 |
| Version/Protocol ID Extension | | | | 1 |
| ID Length | | | | 1 |
| R | R | R | PDU Type | 1 |
| Version | | | | 1 |
| Reserved | | | | 1 |
| Maximum Area Addresses | | | | 1 |
| PDU Length | | | | 2 |
| Source ID | | | | ID Length + 1 |
| Start LSP ID | | | | ID Length + 2 |
| End LSP ID | | | | ID Length + 2 |
| Variable-Length Fields | | | | |

Figure 10-30. This analyzer capture shows the header of an L1 CSNP.

[View full size image]




A PSNP (Figure 10-31) is similar to a CSNP, except that the former describes only some LSPs rather than the entire database. Therefore, no Start and End fields are necessary as they are with CSNPs. A router sends a PSNP on a point-to-point subnetwork to acknowledge received LSPs. On a broadcast subnetwork, PSNPs request missing or more recent LSPs. Like CSNPs, there are both L1 and L2 PSNPs.

Figure 10-31. IS-IS PSNP format.

| | | | | Length, in Octets |
|---|---|---|---|---|
| Intradomain Routeing Protocol Discriminator | | | | 1 |
| Length Indicator | | | | 1 |
| Version/Protocol ID Extension | | | | 1 |
| ID Length | | | | 1 |
| R | R | R | PDU Type | 1 |
| Version | | | | 1 |
| Reserved | | | | 1 |
| Maximum Area Addresses | | | | 1 |
| PDU Length | | | | 2 |
| Source ID | | | | ID Length + 1 |
| Variable-Length Fields | | | | |

Only four TLVs are used by SNPs, whether they are CSNP or PSNP and whether they are L1 or L2:

- LSP Entries (type 9)
- Authentication Information (type 10)
- Optional Checksum (type 12)
- Experimental (type 250)

## LSP Entries TLV

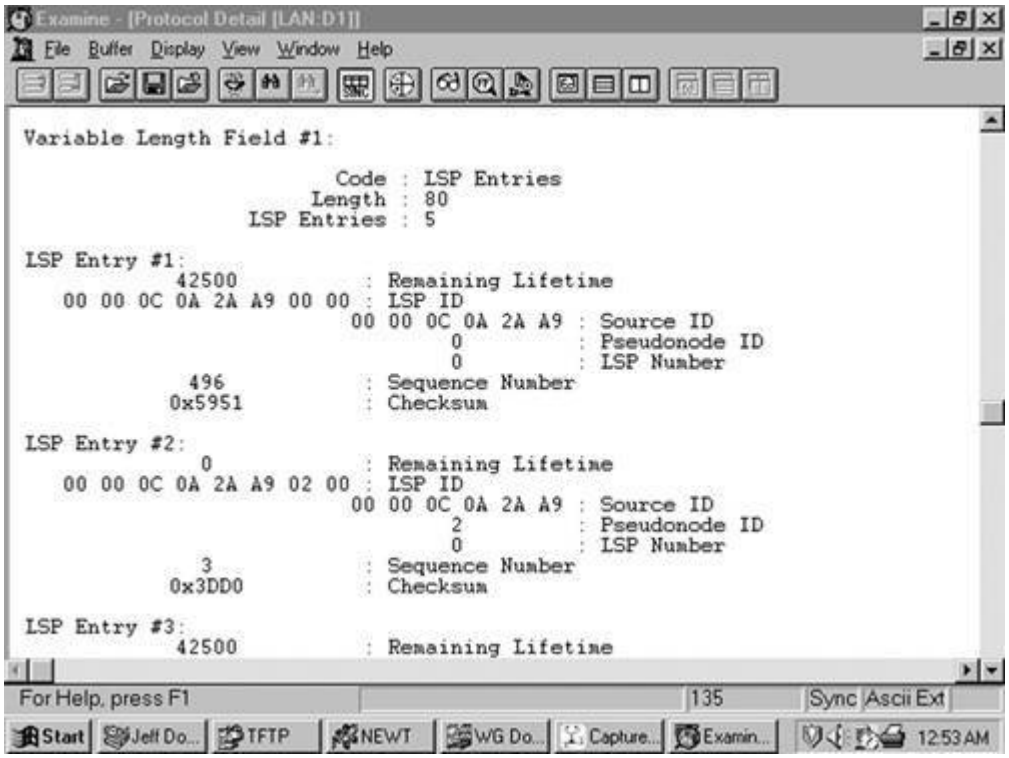
The LSP Entries TLV (Figure 10-32) summarizes an LSP by listing its Remaining Lifetime, LSP ID, Sequence Number, and Checksum. These fields not only identify the LSP but also completely identify a particular instance of an LSP. Figure 10-33 shows part of an LSP Entries TLV.

Figure 10-32. LSP Entries TLV.

| | Length, in Octets |
|---|---|
| Code = 9 | 1 |
| Length | 1 |
| Remaining Lifetime | 2 |
| LSP ID | ID Length + 2 |
| LSP Sequence Number | 4 |
| Checksum | 2 |
| Multiple Fields | |
| Remaining Lifetime | 2 |
| LSP ID | ID Length + 2 |
| LSP Sequence Number | 4 |
| Checksum | 2 |

Figure 10-33. Part of the LSP Entries TLV of the CSNP in Figure 10-30.

[View full size image]




## Extensions to IS-IS

As you have already seen in Table 10-4 and the associated discussion, a number of extensionseither to support optional capabilities or to improve the basic mechanisms of IS-IShave been added since the protocol was first extended to support IPv4. In this section we explore the most important of those extensions.

One of the nice things about basic IS-IS behavior is that unlike OSPFv2's LSAs, if IS-IS encounters a TLV that it does not understand, it just ignores it. This makes extendability and backward-compatibility much easier in IS-IS. If you are migrating your network to support a new feature, or simply have one router that understands an extension and another that does not, there is much less worry that adjacencies would be effected than with OSPFv2. You also saw in Chapter 9, "OSPFv3," that OSPFv3 has adopted at least some of this same behavior, making the protocol more forgiving than OSPFv2.

### 3-Way Handshaking

Before neighbors become adjacent, they must ensure that there is two-way communication between them. The process for ensuring this is called *handshaking*. It is not enough to simply send and receive Hellos, which is two-way handshaking; it is possible that your neighbor is not receiving the Hellos you send. So *three-way handshaking*, in which there is some explicit acknowledgment that your neighbor has received your Hellos, is required. OSPF always uses three-way handshaking, by listing in transmitted Hellos the neighbors that it has received Hellos from. So if an OSPF router sees its RID listed in the Neighbor field of a received Hello, it knows that the originator of the Hello has received this router's Hellos. Bidirectional communication is confirmed.

IS-IS also uses three-way handshaking across broadcast networks. The LAN Hello uses the IS Neighbors TLV to list all neighbors from which a Hello has been received. The TLV then serves the same purpose as the Neighbors list in OSPF Hellos: An IS-IS router that sees its own SysID in the IS Neighbors TLV of a received LAN Hello knows that bidirectional communication is confirmed.

But as has already been mentioned, Point-to-Point Hellos do not carry IS Neighbors TLVs. ISO 10589 prescribes only two-way handshaking across point-to-point links, with the admonition that the point-to-point medium should be reliable. In reality, point-to-point links can often be unreliable; certainly we do not want to reject IS-IS as a potential IGP choice just because we might have a link in our network that does not meet some vague reliability requirement.

To remedy this problem, RFC 3373 specifies a Point-to-Point Three-Way Adjacency TLV. This type 240 TLV, shown in Figure 10-34, lists the SysIDs of all neighbors known by the originator, and also indicates the originator's perception of the state of its adjacency on the link: Up, Initializing, or Down.

Figure 10-34. Point-to-Point Three-Way Adjacency TLV.

| | Length, in Octets |
|---|---|
| Type = 240 (0 X F0) | 1 |
| Length | 1 |
| Adjacency Three-Way State { Up = 0, Initializing = 1, Down = 2 } | 1 |
| Extended Local Circuit ID | 4 |
| Neighbor System ID | ID Length of Point-to-Point Hello |
| Neighbor Extended Local Circuit ID | 4 |

## Domain-Wide Prefix Distribution

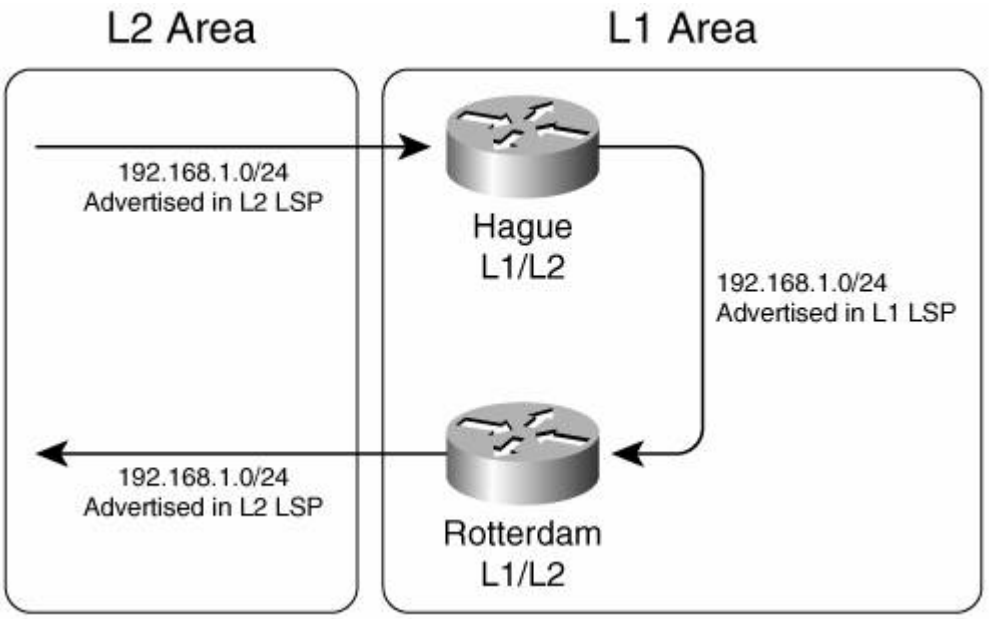
The default characteristic of L1 areas, as you read earlier in this chapter, is very much like an OSPF totally stubby area. That is, no prefixes are advertised from L2 to L1; instead, L1/L2 routers set the ATT bit and L1 routers then install a default router to the nearest L1/L2 router. In fact, RFC 1195 specifically prohibits advertising prefixes from L2 to L1.

But this is not always acceptable. If there are multiple L1/L2 routers, sometimes you would prefer to pick the one closest to the destination rather than just the closest L1/L2 router exiting the area. For this to happen, the L1 routers must have knowledge of specific prefixes and their costs outside of the local area; and that, in turn, means that prefixes must be advertised from L2 to L1. In IS-IS lingo this is called *route leaking*.

The potential difficulty, and the reason RFC 1195 prohibits such "downward" route leaking, is that if you have more that one L1/L2 router in the L1 areawhich is most likely why you want to leak routes from L2 to L1 in the first placeyou have the potential for inter-area routing loops. Figure 10-35 illustrates the problem. The L1/L2 router Hague learns prefix 192.168.0/24 from some L2 LSP. If it is to advertise the prefix into its L1 area, it must advertise the prefix in its L1 LSP using an IP Internal Reachability TLV. When that L1 TLV is flooded, it is received by Rotterdam, another L1/L2 router. But because the prefix is in an L1 LSP, Rotterdam has no way of knowing that it originated outside of the L1 area. So it advertises the prefix back out to the L2 area in an L2 LSP. A routing loop can now exist in which L2 routers think that a path to 192.168.1.0/24 exists within the L1 area.

Figure 10-35. Under basic RFC 1195 rules, a prefix leaked from L2 to

L1 is at risk of being advertised back to L2, creating a routing loop.




OSPF does not have this problem because prefixes external to a non-backbone area are advertised in type 3 LSAs. Other ABRs do not advertise prefixes learned from type 3 LSAs in a non-backbone area into area 0.

RFC 2666 provides a workaround to the problem by defining a bit in the IP Internal Reachability and IP External Reachability TLVs called the *Up/Down* (U/D) bit. Looking at the format of the IP Internal Reachability TLV in Figure 10-26 (and remembering that the format of the IP External Reachability bit is identical), notice that the octet containing the I/E bit and the 6-bit Default Metric field begins with a reserved bit. This bit becomes the U/D bit, as shown in Figure 10-36. As the type field indicates, the bit is defined for both IP Internal Reachability and IP External Reachability TLVs.

Figure 10-36. The eighth bit of the default metric field, previously reserved, is redefined as the Up/Down bit.

| | | | Length, in Octets |
|---|---|---|---|
| Type = 128 or 130 | | | 1 |
| Length | | | 1 |
| U/D | I/E | Default Metric | 1 |
| S | R | Delay Metric | 1 |
| S | R | Expense Metric | 1 |
| S | R | Error Metric | 1 |
| IP Address 1 | | | 4 |
| Subnet Mask 1 | | | 4 |
| ⋮ | | | |
| U/D | I/E | Default Metric | 1 |
| S | R | Delay Metric | 1 |
| S | R | Expense Metric | 1 |
| S | R | Error Metric | 1 |
| IP Address *n* | | | 4 |
| Subnet Mask *n* | | | 4 |

When an L1/L2 router advertises a route from L2 to L1, it sets the U/D bit. Any other L1/L2 router that receives a prefix with its U/D bit set in an L1 LSP does not advertise the prefix in an L2 LSP. If an L1/L2 router does not understand the U/L bit, it ignores the bit. So it is important, if you are using L2 to L1 route leaking, to ensure that all of your L1/L2 routers understand this extension.

The Case Study "Route Leaking Between Levels" later in this chapter demonstrates how to set up L2 to L1 route leaking.

## Wide Metrics

Traffic Engineering (TE) is a function primarily associated with Multiprotocol Label Switching (MPLS) networks in which some subset of packets can be forwarded differently depending on a user-specified set of constraints. In other words, rather than always choosing the single shortest path through a network as an IGP does, traffic flows can be spread out over different paths. This can both help overall bandwidth utilization in a network and enable service differentiationby, for example, ensuring that delay-sensitive flows take the shortest path while other traffic takes longer paths.

One of the keys to traffic engineering is the communication of much more detailed interface parameters than just metrics; it makes sense to use an IGP that is already designed for sharing path and interface information to share these TE interface parameters. Both OSPF and IS-IS have been extended to carry TE interface parameters; for IS-IS, two new TLVs are used:

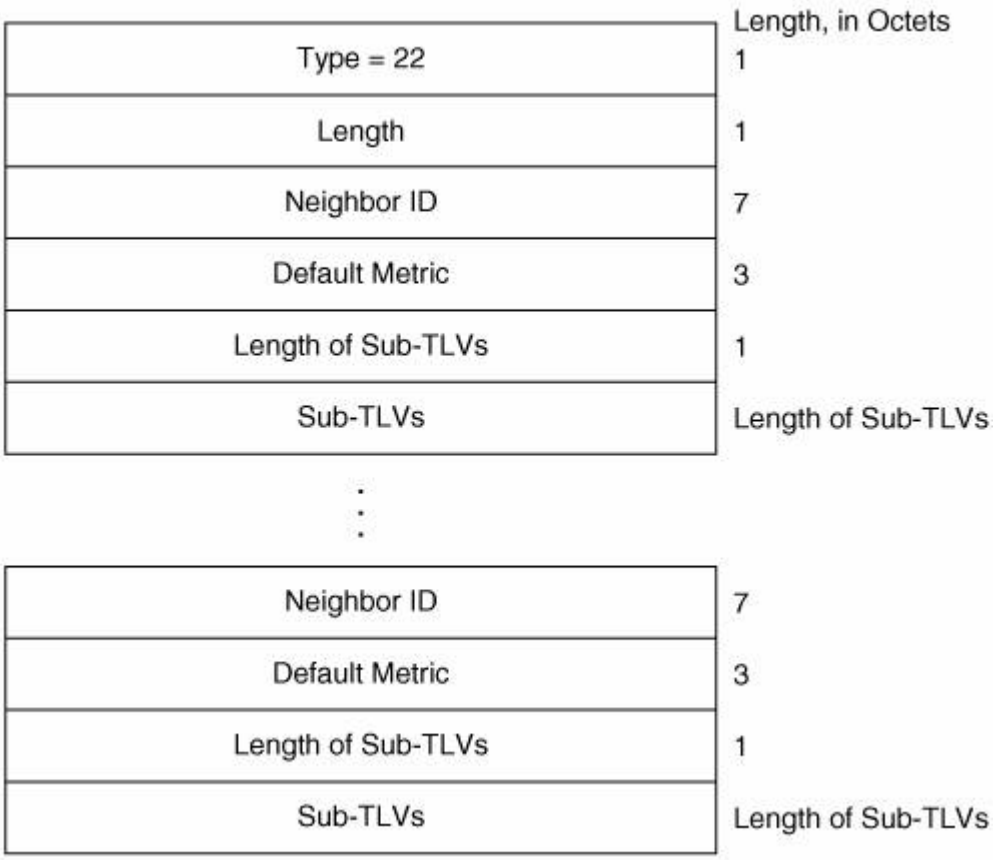
- Extended IS Reachability (type 22)
- Extended IP Reachability (type 135)

These TLVs, and the TE parameters they support, are specified in RFC 3784. However, MPLS Traffic Engineering is beyond the scope of this book. Nevertheless, these two new TLVs are of interest to us because they provide an important new capability that can be used outside of just Traffic Engineering.

As mentioned earlier in this chapter, a concern about IS-IS in its original form is that the six-bit metric field does not allow enough metric granularity for a large network. These new TLVs support a much larger metric space, called wide metrics. The Extended IS Reachability TLV uses a 24-bit metric field, and the Extended IP Reachability TLV uses a 32-bit metric field.

The format of the Extended IS Reachability TLV is shown in Figure 10-37. When wide metrics are enabled, this TLV is used in place of the type 2 IS Neighbors TLV in LSPs. Of interest to us is that like the type 2 TLV, it lists neighbors and their metrics for the SPF calculations; notice the 24-bit metric field. The sub-TLV field is for TE parameters, and has no relevance to this discussion. However, it is useful to note that TLVs such as this one do allow nested TLVsthat is, sub-TLVs or TLVs within other TLVs. That's just one more example of the flexibility that TLVs allow developers.

Figure 10-37. Extended IS Reachability TLV.

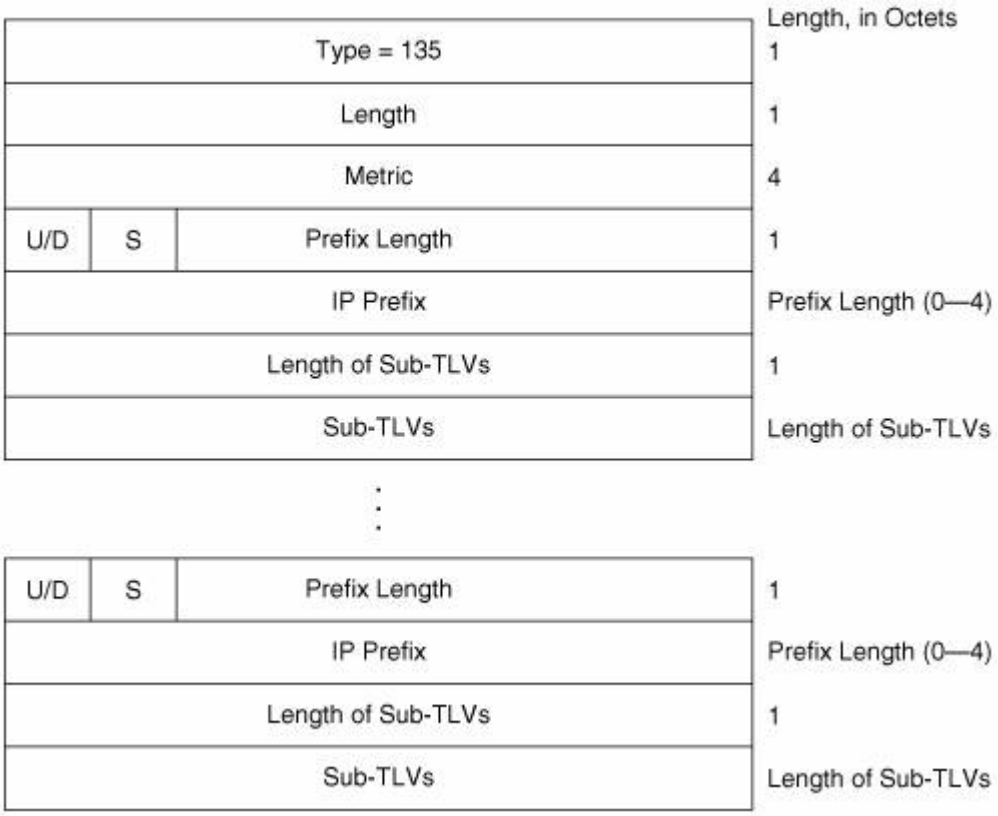



The format of the Extended IP Reachability TLV is shown in Figure 10-38. When wide metrics are enabled, this TLV is used in place of both IP Internal Reachability Information and IP External Reachability Information TLVs. Accordingly, this TLV can appear in both L1 and L2 LSPs. You can see in the illustration that prefixes advertised by this TLV can have a metric of up to 32 bits.

There is also an Up/Down flag for L2 to L1 route leaking. As with the Extended IS Reachability TLV, the sub-TLV fields only have relevance to Traffic Engineering (the S bit indicates the presence of sub-TLVs).

Figure 10-38. Extended IP Reachability TLV.

[View full size image]




Wide metrics are enabled in IOS with the command metric-style wide. Examples of using wide metrics are shown in this chapter in the Case Studies "A Basic Integrated IS-IS Configuration for IPv6" and "Transition to Multiple Topology Mode."

## Routing IPv6 with IS-IS

In Chapter 9, you saw that OSPF support for IPv6 requires an entirely new version of OSPF. IS-IS, on the other hand, is easily extended to support IPv6 through the addition of two new TLVs. As of this writing, the IPv6 extension of IS-IS is still in the Internet-Draft stage, but it is likely to be an RFC very soonprobably by the time you are reading this chapter.

IS-IS indicates its support of IPv6 by including the IPv6 NLPID 142 (0x8E) in its Protocols Supported TLV. The two TLVs for IPv6 support are

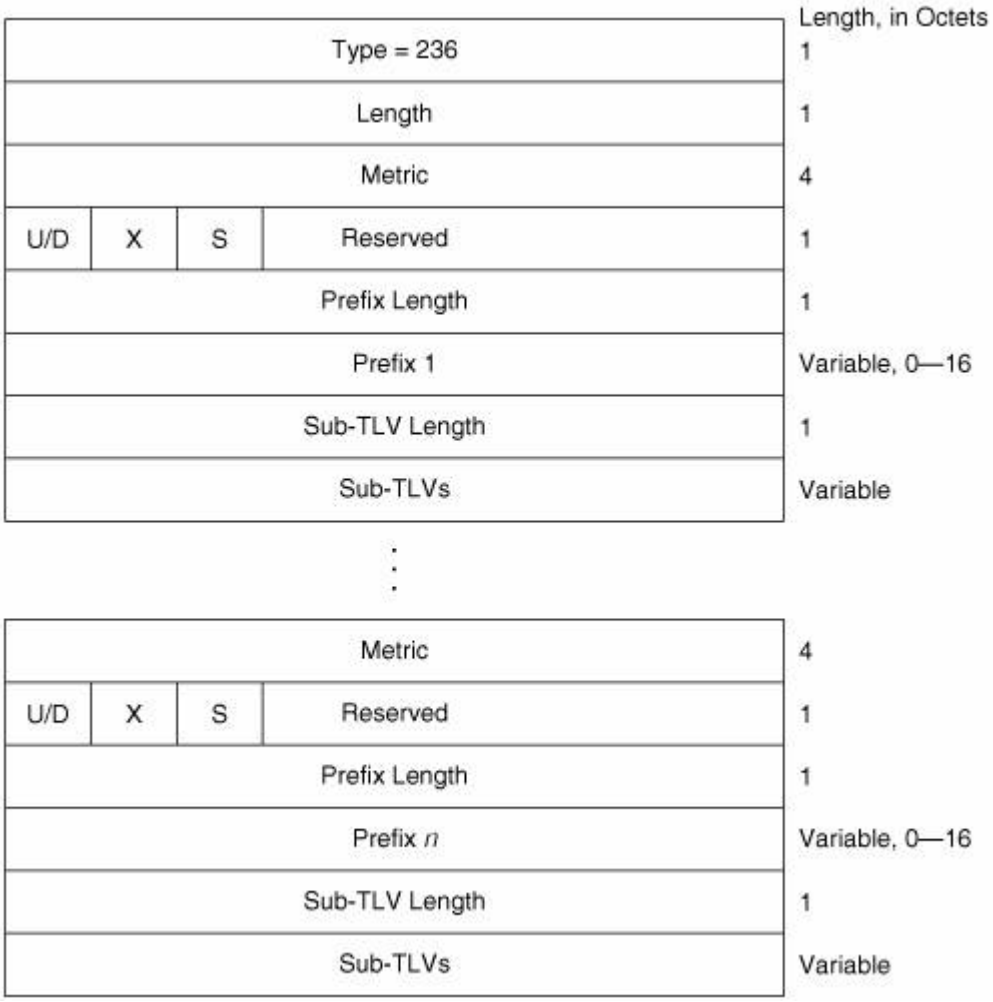
- IPv6 Reachability (type 236)
- IPv6 Interface Address (type 232)

The IPv6 Reachability TLV (Figure 10-39) can be said to serve the same purpose as the IP Internal Reachability Information and IP External Reachability Information TLVs for IPv4. However, the parallel function is actually much closer to the type 135 Extended IP Reachability

TLV for two reasons:

- The one TLV is used to advertise both internal and external prefixes.
- The TLV includes a 32-bit metric field, so wide metrics are supported.

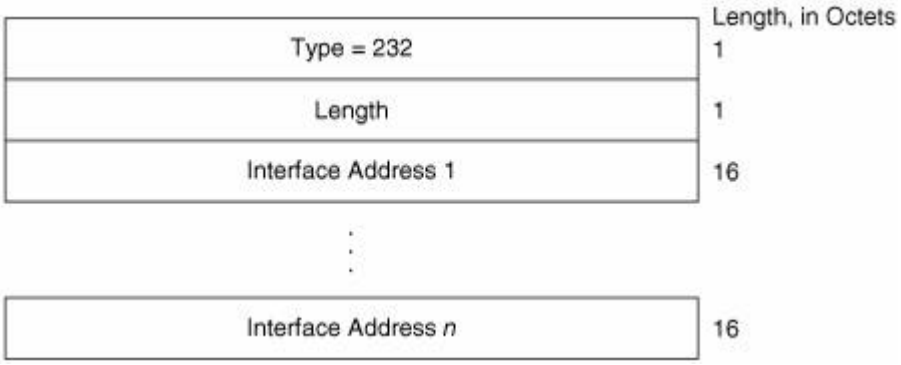
Figure 10-39. IPv6 Reachability TLV.




You can see in the figure the 32-bit metric field and an Up/Down bit for L2 to L1 route leaking for each prefix. The X bit indicates whether the prefix is internally originated (X = 0) or externally originated (X = 1). The S bit indicates whether sub-TLVs exist.

The IPv6 Interface Address TLV, shown in Figure 10-40, is the functional equivalent of the type 132 IP Interface Address TLV for IPv4: It advertises the address of the interface from which the TLV originates. And like its IPv4 counterpart, it can be carried in both Hellos and LSPs. When the TLV appears in Hellos, the addresses it advertises are the link-local addresses of the originating interface. If the TLV appears in LSPs, the addresses it advertises are the non-link-local (site or global scope) addresses of the originating interface.

Figure 10-40. IPv6 Interface Address TLV.




Even though the IP Interface Address TLV can carry up to 63 32-bit IPv4 addresses, IPv6 addresses are four times as long. That restricts the IPv6 Interface Address TLV to carrying a maximum of 15 IPv6 addresses.

An example of configuring IS-IS for IPv6 is provided in the Case Study "A Basic Integrated IS-IS Configuration for IPv6."

## Dynamic Hostname Exchange

One operational difficulty when working with IS-IS is that it can be hard to associate the SysIDs in various router displays with the correct router. Identifying IPv4 addresses is not easy, but we tend to be more comfortable with them just through long familiarity. But SysIDs, such as in the display of Example 10-1, are a real challenge.

RFC 2763 provides a convenient solution to this challenge by specifying a Dynamic Hostname TLV (type 137). This simple TLV provides, in its value field, up to 255 bits for carrying an ASCII router name. Normally, implementations supporting this extension insert the configured router hostname into the field; the TLV is then carried in any non-pseudonode LSP. When a router display is called up, as in Example 10-8, the ASCII value of the Dynamic Hostname TLV is used instead of the SysID, for easier interpretation.

### Example 10-8. This re-display of the IS-IS neighbor table of Example 10-1 shows the advantage of the Dynamic Hostname Exchange mechanism.

```
Brussels#show clns is-neighbors

System Id      Interface   State  Type Priority  Circuit Id        Format
Dublin         Se0         Up     L1   0         06                Phase V
Amsterdam      Et1         Up     L1   64        Brussels.03       Phase V
Rome           Et0         Up     L2   64        Brussels.02       Phase V
London         Et0         Up     L1L2 64/64     Brussels.02       Phase V

Brussels#
```

## Multiple Topologies

Modern IP networks often provide multiple services over a common IP infrastructure, using basic service building blocks such as IPv4, IPv6, and multicast. The financial advantages of using a common infrastructure for all services, as opposed to building separate networks for each service, are enormous: Far less capital expenditure, less equipment to maintain, simpler operational procedures, and fewer operations personnel. But often the requirements of the

separate basic service building blocks require different routing topologies. Perhaps IPv4 is run everywhere, but IPv6 should be limited to just some parts of the IPv4 domain. Multicast might also need to be limited to some subset of the IPv4 domain, but different from the IPv6 topology.

Multi Topology (MT) routing allows you to build these routing subsets on a common infrastructure. The individual topologies are identified by Multi Topology Identifiers (MT IDs). The currently defined MT IDs are listed in Table 10-5. These MT IDs are assigned to router interfaces to indicate what topologies the interface belongs to; each interface can have one or more MT IDs. Adjacencies are not MT specific, but are established between all IS-IS neighbors as usual. But the LSPs are tagged with the appropriate MT IDs, and a separate SPF calculation is run for each topology. In each router, a separate route table is maintained for each topology, and entries are made in them based on the separate PSF calculations.

### Table 10-5. Multi Topology Identifiers used by MT IS-IS

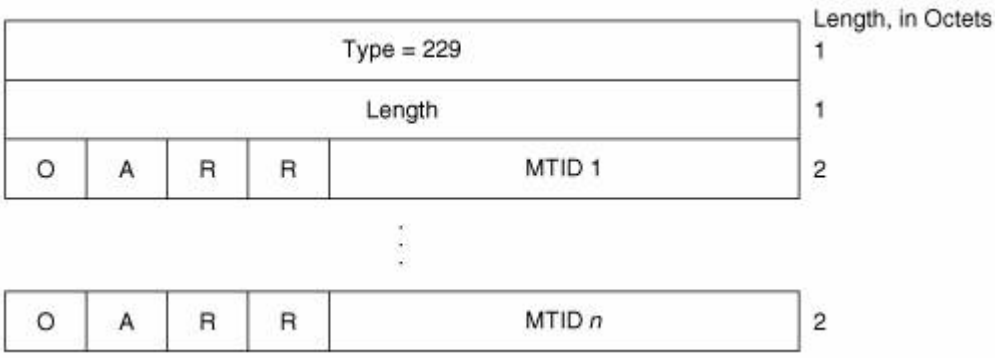
| MT ID | Topology |
|---|---|
| 0 | Standard (default) topology (IPv4 unicast routing topology) |
| 1 | IPv4 in-band management |
| 2 | IPv6 unicast routing topology |
| 3 | IPv4 multicast routing topology |
| 4 | IPv6 multicast routing topology |
| 53995 | Reserved for IETF consensus |
| 39964095 | Reserved for development, experimental, and proprietary features |

IS-IS support for MT routing is currently in the Internet-Draft stage. The draft specifies several TLVs for MT support:

- MT Intermediate Systems (type 222)
- Multi Topology (type 229)
- MT Reachable IPv4 Prefixes (type 235)
- MT Reachable IPv6 Prefixes (type 237)

When an MT IS-IS router sends Hellos, it includes one or more Multi Topology TLVs (Figure 10-41), one for each topology to which the originating interface belongs. If a neighbor does not include any Multi Topology TLVs in its Hellos, the neighbor is considered to belong only to the default IPv4 topology (MT ID 0). On a point-to-point link, if the two neighbors do not have any MT IDs in common, no adjacency is formed. However, behavior is different on a broadcast link; neighbors form an adjacency even if they have no MT IDs in common. This is because the Designated Router election is independent of the MT IS-IS extensions; a router that does not support the extensions can be elected DR, so all routers at the same level must be adjacent.

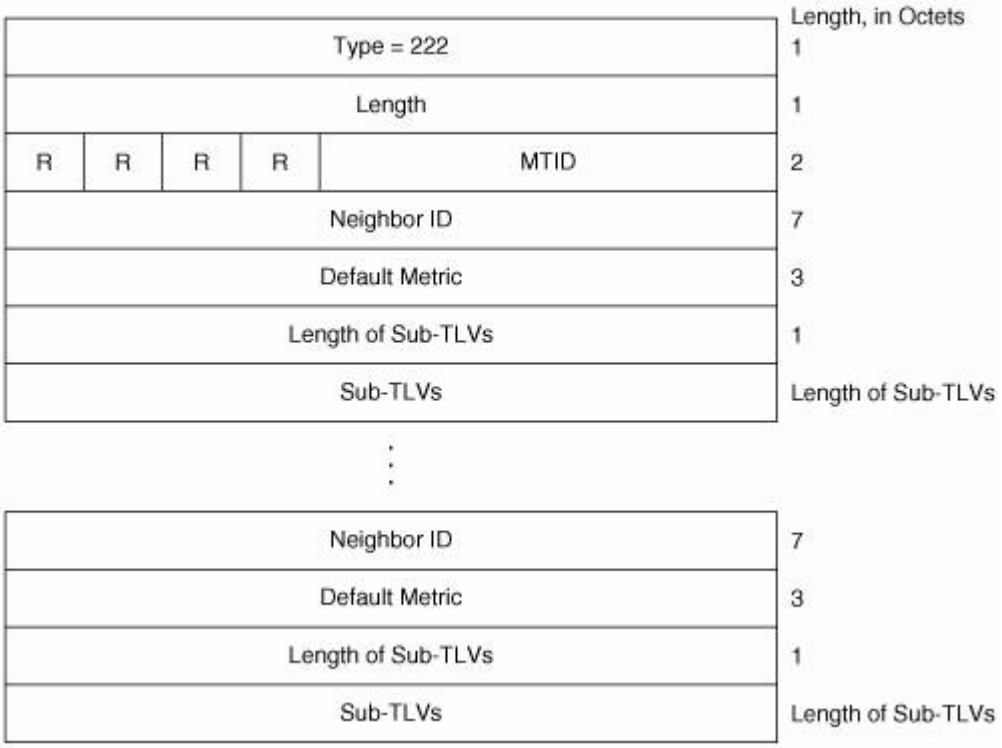
### Figure 10-41. Multi Topology TLV.




The Multi Topology TLV is carried not only in Hellos but also in LSPs. As Figure 10-41 shows, the TLV can signal Overloading (with the O bit) and L2 attachment (with the A bit) separately for each topology.

The format of the MT Intermediate Systems TLV (Figure 10-42) is exactly the same as that of the Extended IS Reachability TLV, and serves the same purpose, except that there is a separate MT Intermediate Systems TLV for each of the topologies to which the originator belongs.

Figure 10-42. MT Intermediate Systems TLV.




The MT Reachable IPv4 Prefixes TLV (Figure 10-43) and the MT Reachable IPv6 Prefixes TLV (Figure 10-44) are the functional equivalents of the Extended IP Reachability and IPv6 Reachability TLVs, respectively. They advertise prefixes, but associated with a given topology.

Figure 10-43. MT Reachable IPv6 Prefixes TLV.

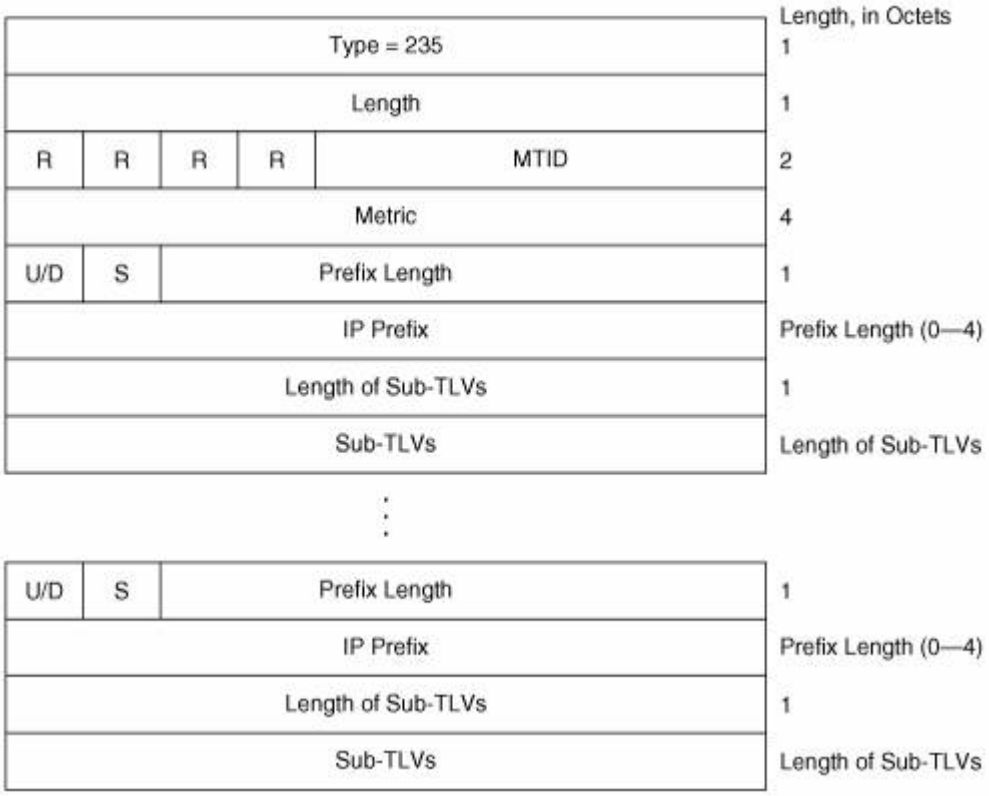



Figure 10-44. MT Reachable IPv6 Prefixes TLV.

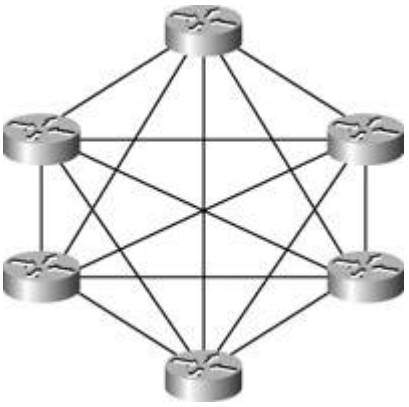
| | | | | | Length, in Octets |
|---|---|---|---|---|---|
| Type = 237 | | | | | 1 |
| Length | | | | | 1 |
| R | R | R | R | MTID | 2 |
| Metric | | | | | 4 |
| U | X | S | Reserved | | 1 |
| Prefix Length | | | | | 1 |
| Prefix 1 | | | | | Variable, 0—16 |
| Sub-TLV Length | | | | | 1 |
| Sub-TLVs | | | | | Variable |
| ⋮ | | | | | |
| Metric | | | | | 4 |
| U | X | S | Reserved | | 1 |
| Prefix Length | | | | | 1 |
| Prefix *n* | | | | | Variable, 0—16 |
| Sub-TLV Length | | | | | 1 |
| Sub-TLVs | | | | | Variable |

An example of configuring multiple IS-IS topologies is provided later in this chapter in the Case Study "Transition to Multiple Topology Mode."

## Mesh Groups

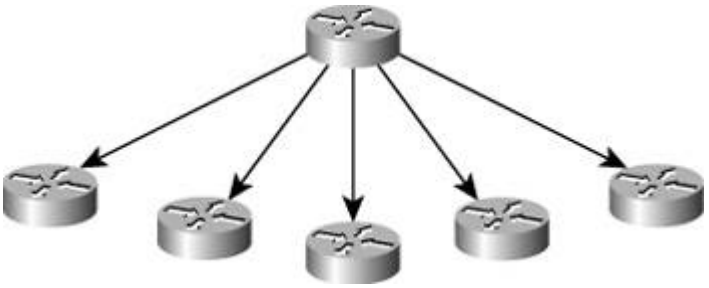
Although the technologies are quickly vanishing in favor of SONET and MPLS, Frame Relay and ATM networks are often still used as the core transport media of many large networks. And a topology frequently used in Frame Relay and ATM networks is that the virtual circuits (VCs) used for connectivity are fully meshed as shown in Figure 10-45. But well-meshed or fully meshed networkswhere all routers have connections to all other routersare susceptible to abnormally heavy flooding loads.

Figure 10-45. ATM and Frame Relay infrastructures underlying an IP network often are configured so that every node is connected to every other nodethat is, the nodes are fully meshed.



Because every router is connected to every other router, when a router floods an LSP every other router immediately receives it, as shown in Figure 10-46. This is as flooding needs to proceed.

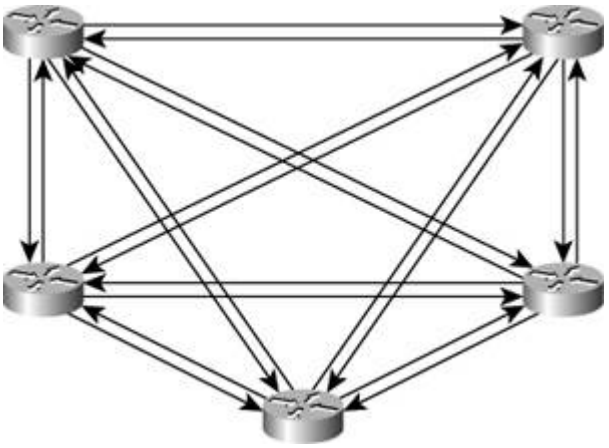
Figure 10-46. In a fully meshed network, a transmission from one router is immediately received by all other routers.



The problem is that the receiving routers have links to other routers, and they have no way of knowing that the flooded LSP has already been received by these other routers. So following the basic split horizon rule for flooding, these other routers flood the LSP out every interface except the one that the LSP was received on, as shown in Figure 10-47. This illustration shows that every router in the network floods $n$ 2 unnecessary LSPs, where $n$ is the number of routers in the network: It floods an LSP to every router except itself and the router from which the LSP was received. As quadratic equations demonstrate, this works out to ($n$ 1)($n$ 2) or $n^2$ 3$n$ + 2 unnecessarily flooded LSPs.[21]

[21] Elementary school math teachers love to torture their students with "language problems," so here's the equation expressed linguistically: Every router except the originator floods $n$ 2 unnecessary LSPs.

Figure 10-47. Even though every router has received the flooded LSP, they still must follow the rules of flooding and send the LSP to every neighbor except the one from which they received the LSP.



In the network we are examining here, the extra flooding load is not that bad: With six routers, $n^2$ $3n$ + 2 = 20 unnecessarily flooded LSPs. But suppose there are 100 fully meshed routers. Now you have $n^2$ $3n$ + 2 = 9,702 unnecessarily flooded LSPs. And this is just from one router refreshing its LSP. Consider every router refreshing its LSPs, possibly multiple LSPs per router, and other activity besides refresh timers triggering LSP floods, and you can see that the unnecessary LSPs at every flood are tremendous. Of course, in a modern network these numbers might not be unreasonable given the overall volume of packets being handled, but to many network architects any inefficiency offends their sense of aesthetics.

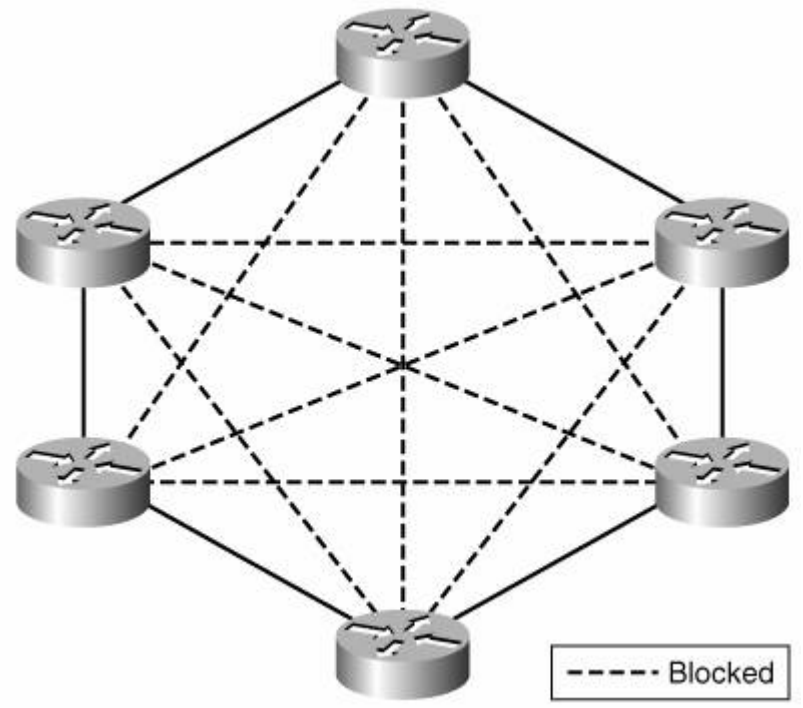
For these individuals who find beauty in simplicity, RFC 2973 offers *mesh groups* for IS-IS. The mesh group mechanism allows you to block flooding of an LSP when you are reasonably certain that a router's neighbor has already received the LSP. Mesh groups place the interface in one of three modes:

- Inactive
- Blocked
- Set

Inactive mode simply means that although the router supports mesh groups, no mesh groups are enabled for that interface and LSPs are flooded normally.

When an interface is in blocked mode, it does not flood any LSPs. Figure 10-48 shows how blocked interfaces might reduce the flooding load of the network of Figure 10-45. Here, the fully meshed topology of Figure 10-45 has been reduced to a "ring" flooding topology; of course, regular packet forwarding can still use the full mesh. In this flooding topology, every router has two neighbors rather than $n$-1 neighbors. There is still a bit of unnecessary flooding, but it is greatly reduced from the level shown in Figure 10-47.

Figure 10-48. Blocking flooding on some interfaces reduces overall flooding load.

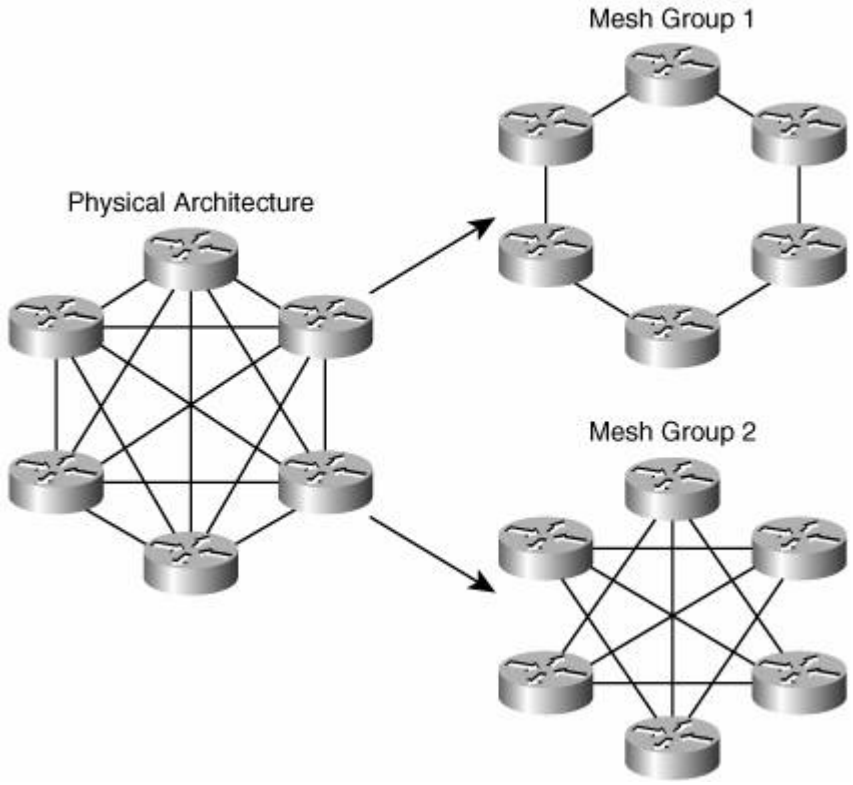



There is a tradeoff for the partially blocked topology in Figure 10-48there is always a tradeoff. First, while every router has an unblocked flooding link to two other routers, if both of those links fail flooding will be corrupted even though the router has other perfectly good links to neighbors. Second, convergence time can be effected. Although full-mesh flooding as shown in Figure 10-46 ensures that every router receives a flooded LSP as soon as it is sent, the partially blocked flooding topology of Figure 10-48 means that a flooded LSP in some cases must transit one or several routers before all routers have received it.

Set mode gives you a bit more flexibility than blocked mode, providing a compromise between the reduced redundancy and increased convergence time of blocked mode while still reducing the flooding loadalthough not as much as blocked mode. In set mode, you define numbered mesh groups and then assign interfaces to the groups. When an LSP is received, it is received on an interface belonging to some numbered group. The LSP is then flooded out all interfaces except interfaces belonging to the same group as the interface on which the LSP was received.

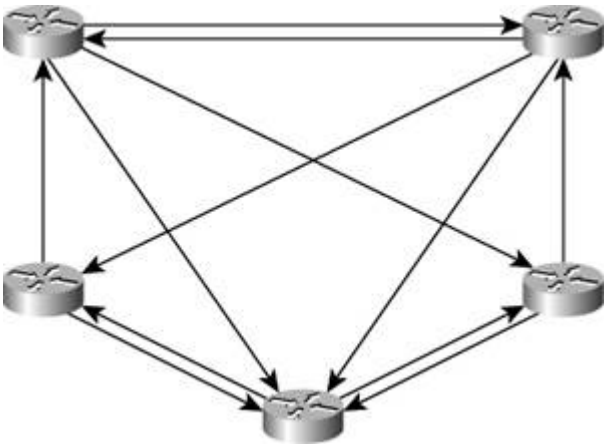
Figure 10-49 shows the topology of Figure 10-45 configured into two mesh groups: Group 1 and Group 2. If a router is originating an LSP, it floods the LSP out every interface no matter what group the interface belongs to; so the first flooding looks just like Figure 10-46. Some of the neighbors receive the LSP on an interface belonging to group 1, and some receive it on interfaces belonging to group 2.

Figure 10-49. Set mode allows you to create numbered mesh groups by assigning interfaces to groups.




The receiving routers then flood the LSP out interfaces belonging to mesh groups other than the one it was received on. If the LSP was received on an interface belonging to group 1, it is only flooded out interfaces belonging to group 2, and vice versa, as shown in Figure 10-50. All of these flooded LSPs are of course unnecessary; every router received a copy of the LSP when the originator initially flooded it. Comparing the flooding pattern to Figure 10-48, you can see that although the unnecessary flooding is heavier than with blocked mode, it is still less than the load depicted in Figure 10-47. And, because the initial flood reached all of the routers, convergence time is not affected.

Figure 10-50. A received LSP is flooded on all interfaces except those belonging to the same mesh group as the interface on which it was received.



Inactive, blocked, and set mode can be mixed in a complex topology to attain the flooding

patterns you want. In practice, blocked mode is used far more often than set mode. But if you choose to use mesh groups at all, be aware that you are in all cases trading some of the robustness of full-mesh flooding for a reduced flooding load.

Using IOS, you can block flooding on an interface with the command `isis mesh-group blocked` or assign the interface to a numbered mesh group with the command `isis mesh-group` *number*.

## Flooding Delays

While mesh groups can control the amount of unnecessary LSPs sent during flooding in densely meshed networks, flooding can also become a problem if the network becomes unstable. For example, a quickly and constantly flapping link can cause tremendous flooding of LSPs, each triggering SPF calculations in all area routers, eating up network resources.

IOS offers several tools for reducing the impact of flooding in such situations. The first is a means of reducing flooding of locally originated LSPs when the router experiences an instability such as a flapping link. In such a situation, imposing a delay before flooding an LSP might mean that not every flap causes a flood. For example, if a link flaps every 20 seconds but flooding is delayed for one minute, only every fifth flap will cause a flood.

But we do not want to always delay flooding; when the network is stable, waiting one minute before flooding a new LSP can cause a serious increase in convergence time. So IOS uses an exponential backoff algorithm in which a very small delay of 50 milliseconds is imposed before the first generation of an LSP. If an event should cause the generation of another LSP within five seconds of the previous generation, the router waits five seconds before generating the new LSP. This delay continues until the network has been calm for 10 seconds, and then the 50 ms generation interval is restored.

The default behavior of this exponential backoff can be changed with the command `lsp-gen-interval`, and the following command options:

- `lsp-initial-wait` is the normal time the router waits before generating an LSP. The default is 50 ms and can be changed between 1 and 120,000 ms.
- `lsp-second-wait` is the delay between the first and second generation of an LSP. The default is 5000 ms (five seconds) and can be changed between 1 and 120,000 ms. After the second LSP generation, the previous delay is automatically doubled for each subsequent generation. So if the second wait is 5 seconds, there will be a wait of 10 seconds between the second and third generation, 20 seconds between the third and fourth generation, 40 seconds between the fourth and fifth, and so on, up to the `lsp-max-wait`. It is the function of this parameter that gives the exponential backoff algorithm its name.
- `lsp-max-wait` is the longest delay allowed between LSP generations. This parameter is necessary so that in the event of a continuing instability the delay will not increase so high that no LSP is ever generated. The default is 5 seconds, and it can be changed to between 1 and 120 seconds.

While `lsp-gen-interval` throttles the generation of LSPs during local instabilities, there might also be a problem in which a particular router is underpowered and can be overwhelmed by heavy flooding. The only real solution to this problem is to upgrade the router. But there are options for controlling the flooding to the neighbor in the interim. The first option is the command `isis lsp-interval`, which changes the default delay between transmissions of LSPs on an interface (and hence to a specific neighbor). The default delay is 33 ms. So, for example, if the delay is set to 100 ms, LSPs cannot be transmitted faster than one every .1 seconds or 10 LSPs per second.

There is also the problem of LSP retransmissions, which again applies to underpowered neighbors. If a router is struggling to process received LSPs, it might be delayed in acknowledging the LSPs, causing its neighbors to retransmit the LSPs. If the router was

struggling to begin with, the retransmissions would make matters worse. The default wait time before retransmitting an LSP is 5 seconds. An underpowered neighbor can be protected somewhat by increasing the wait time, up to 65,535 seconds, with the command isis retransmit-interval.

There might still be a problem in which, even if you had increased the wait time before retransmitting an LSP with the isis retransmit-interval command, the wait time would expire on multiple LSPs and they all would need to be retransmitted. You can regulate the time between the transmission of each of these retransmitted LSPs, in milliseconds, with the command isis retransmit-throttle-interval. So while isis retransmit-interval increases the time to wait for an acknowledgment before retransmitting, isis retransmit-throttle-interval increases the interval between each retransmitted LSP if and when the wait time does expire.

Although these commands give you options for controlling flooding, they should be used only in extreme cases. In the great majority of cases the defaults should not be changed; and where tinkering with these values becomes necessary, you should look to the cause. Usually a router upgrade or an improvement in link reliability is the better solution.

## Improving SPF Efficiency

IOS uses two mechanisms for improving the efficiency of its SPF algorithm:

- Incremental SPF (iSPF)
- Partial Route Calculations (PRC)

When a stub router is added to the networkthat is, when a router is added in which there is only one link to itall routers in the area do not need to recalculate the SPF. Instead, it is enough to just add the router to the tree. And if a link that is not already on the SPF tree changes in some way that does not affect the tree, it is unnecessary for routers to run SPF at all. iSPF takes such cases into account, and only runs SPF to the extent that the topological change warrants. iSPF can also limit the scope of an SPF calculation. That is, if a change affects only a limited part of the topology, iSPF can confine the SPF recalculation to that portion of the topology.

Another change that does not need to trigger an SPF calculation is the addition, deletion, or metric change of an IP prefix. A router detecting such a change floods an LSP with an IP Reachability TLV (or one of its functional equivalents) to advertise the change. But this LSP does not need to trigger an SPF calculation. This is Partial Route Calculation: Received LSPs are examined, and if there is no topological change requiring an SPF calculation, such as a new IP Reachability TLV or a new IS Neighbors TLV, no SPF calculation is run.

In times of network instability, imposing longer delays between SPF runs might span the reception of multiple LSPs. That is, if many LSPs are being flooded, waiting between SPF runs might mean that rather than running SPF every time an LSP is received, the router might receive many LSPs and account for them all with a single SPF run. IOS uses an exponential backoff algorithm like the one used for throttling LSP generation, described in the previous section, for SPF runs. The command spf-interval applies to full SPF runs, and prc-interval for partial route calculations. In both cases, like the LSP generation throttling, you can specify an initial wait, a second wait that is doubled for each subsequent run, and a maximum wait time that cannot be exceeded. The defaults for the SPF exponential backoff are 5500 ms for the initial wait, 5500 ms for the second wait, and 10 seconds for the maximum wait; the defaults for PRC are 2000 ms, 5000 ms, and 5 seconds respectively. However, as with the flooding delays, it is seldom a good idea to change the default values of the SPF delays. The real solutions to overly frequent SPF

PREV NEXT

# Configuring Integrated IS-IS

Integrated IS-IS is unique among the IP routing protocols covered in this book for two reasons. First, it is the only IP routing protocol that must be enabled both as a process and on individual interfaces. Second, it is the only IP routing protocol that was not originally designed for IP. Because IS-IS uses CLNS PDUs rather than IP packets, the configuration is not always as obvious as that of the other protocols.

## Case Study: A Basic IPv4 Integrated IS-IS Configuration

A basic Integrated IS-IS process is configured on a Cisco router in four steps:

Step 1. Determine the area in which the router is to be located and the interfaces on which IS-IS is to be enabled.

Step 2. Enable IS-IS with the command `router isis`.[22]

[22] The `router isis` command can also be given a name, such as `router isis Warsaw`. If IS-IS and ISO-IGRP are configured on the same router, one or both of the processes must be named. If ISO-IGRP is not configured, naming is unnecessary.

Step 3. Configure the NET with the `net` command.

Step 4. Enable Integrated IS-IS on the proper interfaces with the command `ip router isis`. This command must be added not only to transit interfaces (interfaces connected to IS-IS neighbors) but also to interfaces connected to stub networks whose IP addresses should be advertised by IS-IS.

Figure 10-51 shows a small six-router network divided into two areas. In the NETs, areas 1 and 2 will be encoded as 00.0001 and 00.0002, respectively, and the System IDs will be the MAC identifiers of the E0 or FastEthernet0/0 interfaces of each router. Table 10-6 shows the NETs encoded with this information.

Figure 10-51. Area 1 is encoded as 00.0001 in the NET, and area 2 is encoded as 00.0002. The System ID of each NET is the E0 or TO0 MAC identifier.

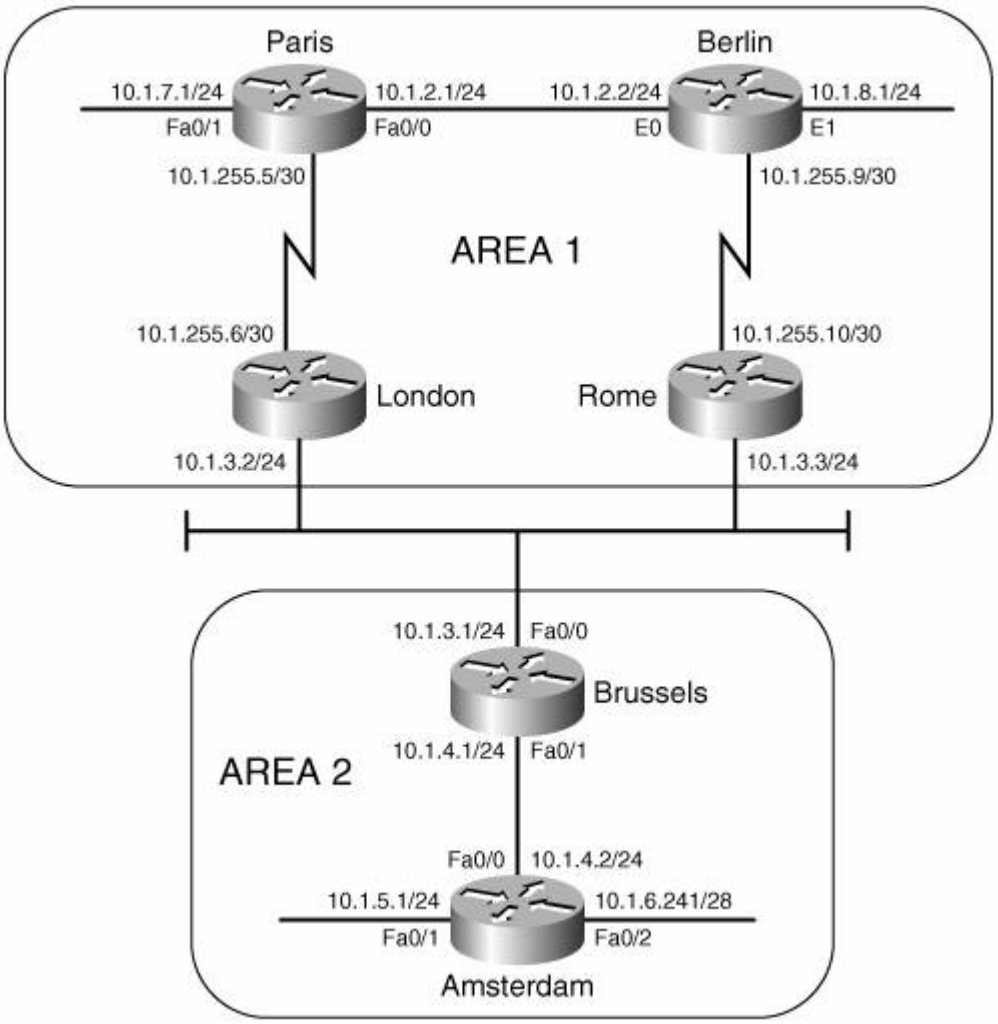



Table 10-6. The NETs used for the IS-IS configurations of the routers in Figure 10-51.

| Router | Area | MAC | Net |
|---|---|---|---|
| Paris | 00.0001 | 0004.c150.e700 | 00.0001.0004.c150.e700.00 |
| Berlin | 00.0001 | 0010.7b3c.6bd3 | 00.0001.0010.7b3c.6bd3.00 |
| London | 00.0001 | 00b0.6430.1de0 | 00.0001.00b0.6430.1de0.00 |
| Rome | 00.0001 | 0004.c150.f1c0 | 00.0001.0004.c150.f1c0.00 |
| Brussels | 00.0002 | 0005.5e6b.50a0 | 00.0002.0005.5e6b.50a0.00 |
| Amsterdam | 00.0002 | 0000.0c8d.34f1 | 00.0002.0000.0c8d.34f1.00 |

The configurations of Paris, London, Brussels, and Amsterdam are displayed in Example 10-9 through Example 10-12.

## Example 10-9. Paris.

```
interface Serial0/0.1 point-to-point
```

```
 ip address 10.1.255.5 255.255.255.252
 ip router isis
!
interface FastEthernet0/0
 ip address 10.1.2.1 255.255.255.0
 ip router isis
!
interface FastEthernet0/1
 ip address 10.1.7.1 255.255.255.0
 ip router isis
!
router isis
 net 00.0001.0004.c150.e700.00
```

Example 10-10. London.

```
interface Ethernet0/0
 ip address 10.1.3.2 255.255.255.0
 ip router isis
!
interface Serial0/0.1 point-to-point
 ip address 10.1.255.6 255.255.255.252
 ip router isis
!
router isis
 net 00.0001.00b0.6430.1de0.00
```

Example 10-11. Brussels.

```
interface FastEthernet0/0
 ip address 10.1.3.1 255.255.255.0
 ip router isis
!
interface FastEthernet0/1
 ip address 10.1.4.1 255.255.255.0
 ip router isis
!
router isis
 net 00.0002.0005.5e6b.50a0.00
```

Example 10-12. Amsterdam.

```
clns routing
!
interface FastEthernet0/0
 ip address 10.1.4.2 255.255.255.0
 ip router isis
!
interface FastEthernet0/1
 ip address 10.1.5.1 255.255.255.0
 ip router isis
!
interface FastEthernet0/2
 ip address 10.1.6.241 255.255.255.240
 ip router isis
!
```

```
router isis
 net 00.0002.0000.0c8d.34f1.00
```

The configurations of Berlin and Rome are similar. A detail that you should notice is that CLNS routing is enabled in Amsterdam's configuration. The CLNS routing is necessary to handle the CLNS PDUs used by IS-IS. CLNS routing is enabled automatically when you create an IS-IS process. In some versions of IOS, you might see the `clns routing` command in the configuration, even though it is not required to enter it as a configuration step.

Example 10-13 shows Paris's route table. Notice that the table contains both L1 and L2 routes. By default, Cisco routers are L1/L2 routers. This fact is also apparent by observing the routers' IS neighbor tables, as in Example 10-14.

## Example 10-13. Paris's route table shows both L1 and L2 routes, indicating that this router is an L1/L2 router.

```
Paris#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     10.0.0.0/8 is variably subnetted, 9 subnets, 3 masks
i L1    10.1.8.0/24 [115/20] via 10.1.2.2, FastEthernet0/0
i L1    10.1.3.0/24 [115/20] via 10.1.255.6, Serial0/0.1
C       10.1.2.0/24 is directly connected, FastEthernet0/0
C       10.1.7.0/24 is directly connected, FastEthernet0/1
i L2    10.1.5.0/24 [115/40] via 10.1.255.6, Serial0/0.1
i L2    10.1.4.0/24 [115/30] via 10.1.255.6, Serial0/0.1
C       10.1.255.4/30 is directly connected, Serial0/0.1
i L1    10.1.255.8/30 [115/20] via 10.1.2.2, FastEthernet0/0
i L2    10.1.6.240/28 [115/40] via 10.1.255.6, Serial0/0.1
Paris#
```

## Example 10-14. Berlin's IS neighbor table shows that both Paris and Rome are L1/L2 routers.

```
Berlin#show clns is-neighbors

System Id      Interface   State  Type Priority  Circuit Id       Format
Rome           Se0.1       Up     L1L2 0 /0      00               Phase V
Paris          Et0         Up     L1L2 64/64     Berlin.01        Phase V
Berlin#
```

Notice in Berlin's IS neighbor table that the name of each neighbor is listed. Hostnames are dynamically exchanged using the TLV type 137, as discussed in the section "Dynamic Hostname Exchange." To view the system identifiers associated with each hostname, use the command `show isis hostname`, as displayed in Example 10-15.

## Example 10-15. The system IDs associated with known hostnames are displayed using the command *show isis hostname*.

```
Berlin#show isis hostname
Level  System ID      Dynamic Hostname  (notag)
1      00B0.6430.1DE0 London
2      0000.0C8D.34F1 Amsterdam
1      0004.C150.E700 Paris
1      0004.C150.F1C0 Rome
     * 0010.7B3C.6BD3 Berlin
Berlin#
```

Because every router in the network of Figure 10-51 is an L1/L2, every router has formed both L1 adjacencies and L2 adjacencies. Consequently, every router has both an L1 and an L2 LS database. The L1 areas completely overlap with the L2 area. For example, Example 10-16 shows the LS databases of Amsterdam. The L1 database contains an LSP originated by Amsterdam[23] and an LSP originated by Brussels. It also contains a pseudonode LSP (Brussels.02-00) originated by Brussels, representing the Ethernet link between Brussels and Amsterdam. Remember that the LSP ID is recognizable as that of a pseudonode LSP because the next-to-last octet, the Pseudonode ID, is non-zero.

[23] As discussed earlier, the asterisk after the LSP ID indicates that the LSP was originated by this router.

### Example 10-16. Amsterdam has both a level 1 LS database and a level 2 LS database, indicating that the router is an L1/L2 router.

```
Amsterdam#show isis database

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime     ATT/P/OL
Amsterdam.00-00     * 0x00000015   0x642E        1112             1/0/0
Brussels.00-00        0x00000016   0x9B81        1187             1/0/0
Brussels.02-00        0x00000013   0x46BD        1139             0/0/0
IS-IS Level-2 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime     ATT/P/OL
Amsterdam.00-00     * 0x00000015   0xEAFF        1176             0/0/0
Paris.00-00           0x00000023   0x2B29        504              0/0/0
Rome.00-00            0x0000001D   0xD016        482              0/0/0
Brussels.00-00        0x00000019   0x45BB        1187             0/0/0
Brussels.02-00        0x00000013   0xD5B6        375              0/0/0
Berlin.00-00          0x0000001E   0x408B        506              0/0/0
Berlin.01-00          0x00000013   0xB10F        557              0/0/0
London.00-00          0x00000019   0xF1B1        463              0/0/0
London.01-00          0x00000013   0x9E92        844              0/0/0
Amsterdam#
```

The three level-1 LSPs indicate that, other then Amsterdam, the only L1 router known to Amsterdam is Brussels. This single node is expected because Brussels is the only other router in area 2. Amsterdam's L2 database shows that Amsterdam knows of every router in the IS-IS domain, which is also expected because every router is an L1/L2.

## Case Study: Changing the Router Types

In a small network like the one in Figure 10-51, leaving all routers at their default type is acceptable. As the network grows, these defaults become less and less acceptable. Not only is the processing and maintenance of two LS databases a burden on the router's CPU and memory, the L1 and L2 IS-IS PDUs being originated by every router become a burden on the buffers and bandwidth.

Paris, Berlin, and Amsterdam in Figure 10-51 can be configured as L1 routers, because they have no direct connection to another area. To change the default router type, use the command is-type. For example, to make Berlin an L1 router, its configuration is as displayed in Example 10-17.

## Example 10-17. Berlin is configured to be a L1 router.

```
router isis
 net 00.0001.0000.3090.c7df.00
 is-type level-1
```

Paris and Amsterdam are configured similarly. Comparing Paris's route table in Example 10-18 with its route table in Example 10-13 shows that the L2 routes have been deleted. Similarly, comparing Example 10-19 with Example 10-16 shows that Amsterdam now has only an L1 LS database.

## Example 10-18. After Paris is configured as an L1 router, its route table contains routes only to destinations within its own area.

```
Paris#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 10.1.255.6 to network 0.0.0.0
     10.0.0.0/8 is variably subnetted, 6 subnets, 2 masks
i L1    10.1.8.0/24 [115/20] via 10.1.2.2, FastEthernet0/0
i L1    10.1.3.0/24 [115/20] via 10.1.255.6, Serial0/0.1
C       10.1.2.0/24 is directly connected, FastEthernet0/0
C       10.1.7.0/24 is directly connected, FastEthernet0/1
C       10.1.255.4/30 is directly connected, Serial0/0.1
i L1    10.1.255.8/30 [115/20] via 10.1.2.2, FastEthernet0/0
i*L1 0.0.0.0/0 [115/10] via 10.1.255.6, Serial0/0.1
Paris#
```

## Example 10-19. After Amsterdam is configured as an L1 router, it has only a level 1 link-state database.

```
Amsterdam#show isis database

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Amsterdam.00-00     * 0x00000017   0x5644        1065              0/0/0
Brussels.00-00        0x00000018   0x9783        1063              1/0/0
Brussels.02-00        0x00000014   0x44BE        1063              0/0/0
Amsterdam#
```

Recall from the earlier discussion of the ATT bit in the LSPs that an L1/L2 router uses the ATT bit to tell L1 routers that it has an inter-area connection. Example 10-20 shows that both London's LSP and Rome's LSP have ATT = 1. So Paris should know to send inter-area traffic to either London or Rome. In other words, Paris should have a default route to London or Rome, with

London being preferred because it is metrically closer. Example 10-18 shows that there is a default route (0.0.0.0) in Paris's route table.

## Example 10-20. The L1 LSPs of London and Rome have ATT = 1, indicating a connection to another area.

```
Paris#show isis database

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum LSP Holdtime      ATT/P/OL
Paris.00-00         * 0x00000022   0xA4C8       857               0/0/0
Rome.00-00            0x00000018   0x139F       727               1/0/0
Berlin.00-00          0x0000001C   0x28B8       733               0/0/0
Berlin.01-00          0x00000018   0x161F       851               0/0/0
London.00-00          0x00000017   0xDD92       848               1/0/0
London.01-00          0x00000014   0x9F54       1080              0/0/0
Paris#
```

In older versions of IOS, the IP process does not directly interpret the ATT bit. The ATT bit is a CLNS function. When running older versions of IOS, if no default route is automatically created in the L1 routers, there are two solutions to the problem. The first solution is to enable IS-IS for CLNS on the interfaces in addition to IS-IS for IP. For example, the serial interface configurations for London and Paris are displayed in Example 10-21 and Example 10-22.

## Example 10-21. London is configured with IS-IS for CLNS on the interface to enable processing of the ATT bit.

```
interface Serial0/0.1
 ip address 10.1.255.6 255.255.255.252
 ip router isis
 clns router isis
```

## Example 10-22. Paris is configured with IS-IS for CLNS on the interface to enable processing of the ATT bit.

```
interface Serial0/0.1
 ip address 10.1.255.5 255.255.255.252
 ip router isis
 clns router isis
```

This first solution works in a dual CLNP/IP environment, but if IS-IS is being used as an IP-only routing protocol, enabling CLNS routing just for default IP routes might be undesirable. A second solution to the default route problem is to configure a static default route on the L1/L2 router and configure IS-IS to advertise the default with the command default-information originate. Using this method in area 2 of Figure 10-51, the Brussels' configuration is displayed in Example 10-23.

## Example 10-23. Brussels is configured to originate a default route.

```
router isis
 net 00.0002.0000.0c76.5b7c.00
 default-information originate
```

```
!
ip route 0.0.0.0 0.0.0.0 Null0
```

Default routes and the default-information originate command are discussed in more detail in Chapter 12, "Default Routes and On-Demand Routing."

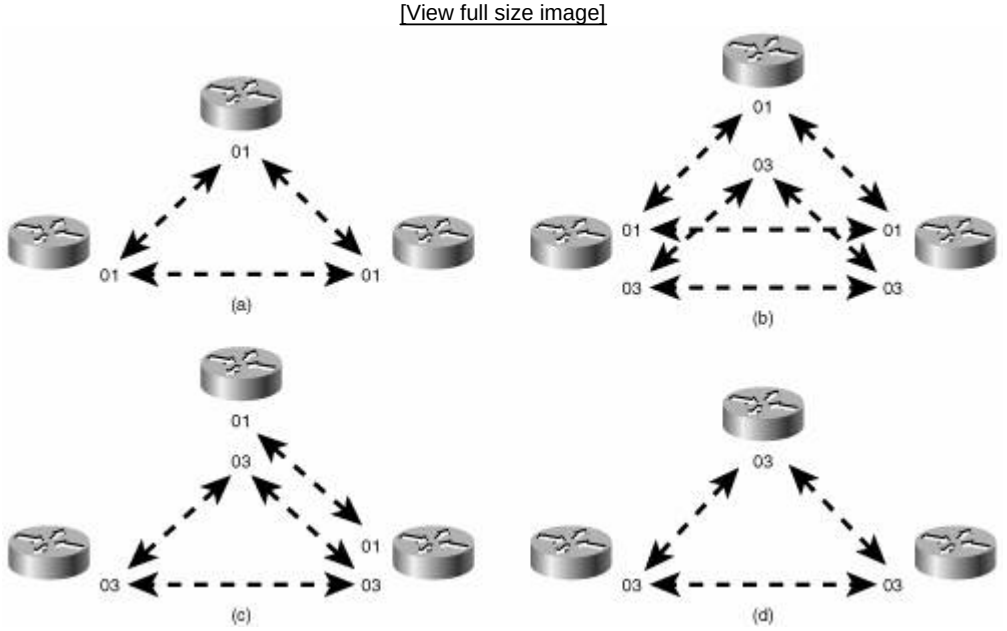
## Case Study: An Area Migration

To change area addresses in an OSPF domain, downtime must be scheduled. However, IS-IS is designed to allow areas to be changed nondisruptively. As discussed in "Operation of Integrated IS-IS," Cisco routers can be configured with up to 254 L1 area addresses. For two routers to form an L1 adjacency, they must have at least one area address in common. With multiple area addresses allowed, a new adjacency can take over while an old adjacency is being broken. This approach is useful when areas are being consolidated or split, when an area is being renumbered, or when area addresses assigned by multiple addressing authorities are being used in the same IS-IS domain.

For example, the routers in Figure 10-52(a) all have an area address of 01. (A NET of one of these routers would look like 01.0000.0c12.3456.00.) In Figure 10-52(b), the routers have been assigned an additional area address of 03. Although multiple adjacencies are not actually formed, the routers do recognize that they have multiple area addresses in common. In Figure 10-52(c), area 01 has been removed from one of the routers. All three routers remain adjacent, because they all have at least one area address in common. Finally, in Figure 10-52(d), all of the 01 area addresses have been removed and the routers are all in area 03. At no time during the area migration was an adjacency lost.

Figure 10-52. The support of multiple area addresses per router eases area changes.

[View full size image]




Suppose that the "powers that be" over the network in Figure 10-51 decree that the area addressing scheme being used is inappropriate and should become GOSIP compliant. After registering with the U.S. GSA, the following components are to be used to construct the NETs:

| | |
|---|---|
| AFI: | 47 |
| IDI: | 0005 |
| DFI: | 80 |
| AAI: | 00ab7c |
| Reserved: | 0000 |
| RDI: | fffe9 |
| Areas: | 0001 (area 1),<br>0002 (area 2) |

The new NETs are shown in Table 10-7.

### Table 10-7. The new GOSIP-format NETs to be assigned to the routers in Figure 10-51.

| Router | NET |
|---|---|
| Paris | 47.0005.80.00ab7c.0000.ffe9.0001.0004.c150.e700.00 |
| Berlin | 47.0005.80.00ab7c.0000.ffe9.0001.0010.7b3c.6bd3.00 |
| London | 47.0005.80.00ab7c.0000.ffe9.0001.00b0.6430.1de0.00 |
| Rome | 47.0005.80.00ab7c.0000.ffe9.0001.0004.c150.f1c0.00 |
| Brussels | 47.0005.80.00ab7c.0000.ffe9.0002.0005.5e6b.50a0.00 |
| Amsterdam | 47.0005.80.00ab7c.0000.ffe9.0002.0000.0c8d.34f1.00 |

The first step in changing the area addresses is to add the new NETs to the routers without changing the old NETs. Rome's IS-IS configuration is shown in Example 10-24.

### Example 10-24. Rome is configured with two NETs during transition.

```
router isis
 net 00.0001.0000.0c0a.2aa9.00
 net 47.0005.8000.ab7c.0000.ffe9.0001.0004.c150.f1c0.00
```

The other five routers are configured similarly. The results can be observed by using the detail keyword with either the show isis database command (Example 10-25) or the show clns is-neighbors command (Example 10-26). In both databases, multiple areas are associated with each router in the network.

### Example 10-25. The LSPs in Rome's link-state database show that all routers in the network of Figure 10-41 are advertising two area addresses.

```
Rome#show isis database detail
IS-IS Level-1 Link State Database:
```

```
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Paris.00-00           0x00000026   0xC3AA        886               0/0/0
  Area Address: 00.0001
  Area Address: 47.0005.8000.ab7c.0000.ffe9.0001
  NLPID:        0xCC
  Hostname: Paris
  IP Address:   10.1.7.1
  Metric: 10         IP 10.1.2.0 255.255.255.0
  Metric: 10         IP 10.1.255.4 255.255.255.252
  Metric: 10         IP 10.1.7.0 255.255.255.0
  Metric: 10         IS Berlin.01
  Metric: 10         IS London.00
Rome.00-00         *  0x0000001E   0x4D64        688               1/0/0
  Area Address: 00.0001
  Area Address: 47.0005.8000.ab7c.0000.ffe9.0001
  NLPID:        0xCC
  Hostname: Rome
  IP Address:   10.1.255.10
  Metric: 10         IP 10.1.3.0 255.255.255.0
  Metric: 10         IP 10.1.255.8 255.255.255.252
  Metric: 10         IS Berlin.00
  Metric: 10         IS London.01
Berlin.00-00          0x00000024   0xC419        716               0/0/0
  Area Address: 00.0001
  Area Address: 47.0005.8000.ab7c.0000.ffe9.0001
  NLPID:        0xCC
  Hostname: Berlin
  IP Address:   10.1.8.1
  Metric: 10         IP 10.1.2.0 255.255.255.0
  Metric: 10         IP 10.1.8.0 255.255.255.0
  Metric: 10         IP 10.1.255.8 255.255.255.252
  Metric: 10         IS Berlin.01
  Metric: 10         IS Berlin.02
  Metric: 10         IS Rome.00
Berlin.01-00          0x0000001B   0x1022        572               0/0/0
  Metric: 0          IS Berlin.00
  Metric: 0          IS Paris.00
London.00-00          0x0000001A   0x1959        1044              1/0/0
  Area Address: 00.0001
--More
```

Example 10-26. Rome's IS-IS neighbor table also shows multiple addresses associated with each neighbor.

```
Rome#show clns is-neighbors detail
System Id      Interface   State  Type Priority  Circuit Id        Format
Berlin         Se0/0.1     Up     L1   0         00                Phase V
  Area Address(es): 00.0001 47.0005.8000.ab7c.0000.ffe9.0001
  IP Address(es):  10.1.255.9*
  Uptime: 00:30:49
London         Fa0/0       Up     L1L2 64/64     London.01         Phase V
  Area Address(es): 00.0001 47.0005.8000.ab7c.0000.ffe9.0001
  IP Address(es):  10.1.3.2*
  Uptime: 04:49:01
  NSF capable
Brussels       Fa0/0       Up     L2   64        London.01         Phase V
  Area Address(es): 00.0002 47.0005.8000.ab7c.0000.ffe9.0002
  IP Address(es):  10.1.3.1*
  Uptime: 04:51:46
```

```
  NSF capable
Rome#
```

The last step of the migration is to delete the old NET statements from all routers. For example, under Rome's IS-IS configuration the command no net 00.0001.0004.c150.f1c0.00 is entered. Example 10-27 shows some of the LSPs in Rome's database after the old NET statements have been removed from the routers.

### Example 10-27. The LSPs in Rome's database show only a single area address.

```
Rome#show isis data detail
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Paris.00-00           0x0000002D   0x412E        1171              0/0/0
  Area Address: 47.0005.8000.ab7c.0000.ffe9.0001
  NLPID:        0xCC
  Hostname: Paris
  IP Address:   10.1.7.1
  Metric: 10         IP 10.1.2.0 255.255.255.0
  Metric: 10         IP 10.1.255.4 255.255.255.252
  Metric: 10         IP 10.1.7.0 255.255.255.0
  Metric: 10         IS Berlin.01
  Metric: 10         IS London.00
Rome.00-00          * 0x00000025   0x0BA7        1174              1/0/0
  Area Address: 47.0005.8000.ab7c.0000.ffe9.0001
  NLPID:        0xCC
  Hostname: Rome
  IP Address:   10.1.255.10
  Metric: 10         IP 10.1.3.0 255.255.255.0
  Metric: 10         IP 10.1.255.8 255.255.255.252
  Metric: 10         IS Berlin.00
  Metric: 10         IS London.01
Berlin.00-00          0x00000029   0x20C0        1097              0/0/0
  Area Address: 47.0005.8000.ab7c.0000.ffe9.0001
  NLPID:        0xCC
  Hostname: Berlin
  IP Address:   10.1.8.1
  Metric: 10         IP 10.1.2.0 255.255.255.0
  Metric: 10         IP 10.1.8.0 255.255.255.0
  Metric: 10         IP 10.1.255.8 255.255.255.252
  Metric: 10         IS Berlin.01
  Metric: 10         IS Berlin.02
  Metric: 10         IS Rome.00
Berlin.01-00          0x0000001D   0x0C24        1090              0/0/0
  Metric: 0          IS Berlin.00
  Metric: 0          IS Paris.00
London.00-00         0x0000001F    0xB7BD        1163              1/0/0
  Area Address: 47.0005.8000.ab7c.0000.ffe9.0001
  NLPID: 0xCC
 --More
```

## Case Study: Route Summarization

Route summarization between areas in a link-state protocol is introduced in Chapter 8. A more complete discussion of summarization, in the context of default routes, is presented in Chapter

12. Briefly, summary routes are useful because

- They reduce the size of LSPs, which reduces the size of the link-state database; consequently, memory and CPU are conserved.
- They hide instabilities within areas. If an address within a summary range changes or a link changes state, the change is not advertised outside of the summarized area.

The primary disadvantages of summary routes are

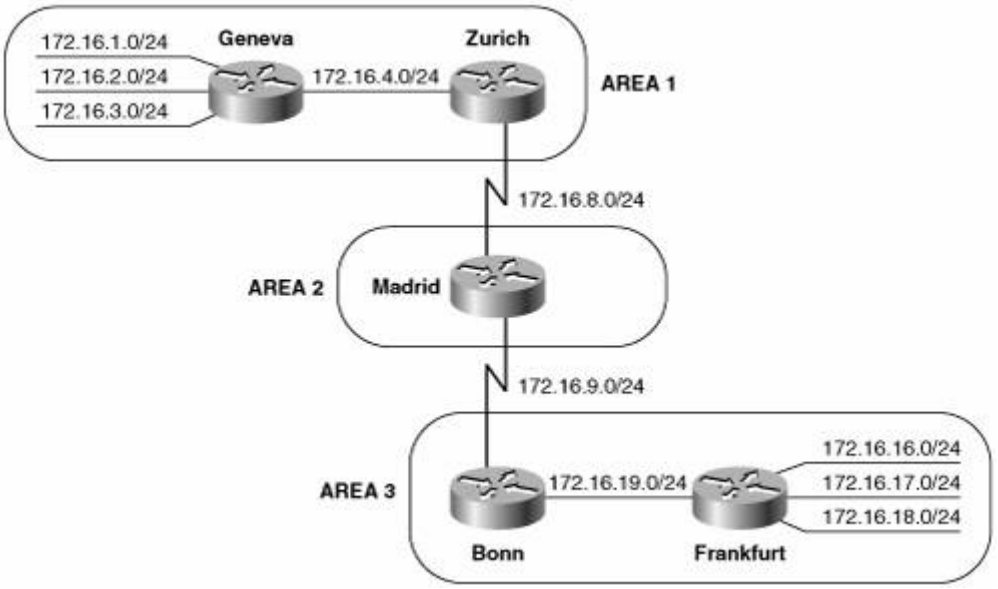
- Their effectiveness is dependent on being able to summarize a contiguous range of IP addresses, so addresses must be planned carefully.
- They can reduce route precision by hiding the details of the area. If there are multiple paths into a summarized area, the best path cannot be determined.

Summarization is enabled under the IS-IS configuration with the summary-address command. Any more-specific destination addresses that fall within the summarization range are suppressed, and the metric of the summary route is the smallest metric of all the more-specific addresses.

Figure 10-53 shows an IS-IS network with three areas. The addresses within area 1 can be summarized with 172.16.0.0/21, and the addresses within area 3 can be summarized with 172.16.16.0/21. The configurations of Zurich, Madrid, and Bonn are displayed in Example 10-28 through Example 10-30.[24]

[24] Notice that Madrid, which has no L1 neighbors, is configured as an L2 router.

Figure 10-53. Zurich and Bonn are summarizing areas 1 and 3 into area 2.




Example 10-28. Zurich is configured to perform route summarization.

```
router isis
 net 01.0000.0c76.5b7c.00
 summary-address 172.16.0.0 255.255.248.0
```

### Example 10-29. Madrid's configuration has no route summarization.

```
router isis
 net 02.0000.3090.6756.00
 is-type level-2-only
```

### Example 10-30. Bonn is configured to perform route summarization.

```
router isis
 net 03.0000.0c0a.2aa9.00
 summary-address 172.16.16.0 255.255.248.0
```

Notice that Madrid, which has no L1 neighbors, is configured as an L2 router. Zurich and Bonn are summarizing their areas into the level 2 backbone. The results of the summarization can be seen in Madrid's route table (Example 10-31).

### Example 10-31. Madrid's route table shows the summary addresses advertised by Bonn and Zurich.

```
Madrid#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
i L2    172.16.16.0/21 [115/20] via 172.16.9.2, Serial0/0.2
C       172.16.8.0/24 is directly connected, Serial0/0.1
C       172.16.9.0/24 is directly connected, Serial0/0.2
i L2    172.16.0.0/21 [115/20] via 172.16.8.2, Serial0/0.1
Madrid#
```

## Case Study: Authentication

IS-IS authentication can use cleartext passwords or HMAC-MD5. There are two methods of configuring cleartext password. Both provide weak security against a determined attack on the network but are effective for preventing service disruptions from misconfigured or unauthorized routers. One of the cleartext configuration modes uses key chains, which can be encrypted in the configuration file so that someone cannot inadvertently obtain the password by looking at the file. Cleartext authentication without key chains is considered the "old style" password.

Cisco IOS supports IS-IS authentication on three levels: between neighbors, area-wide, and domain-wide. The three authentication levels can be used by themselves or together. The rules for IS-IS authentication are

- When authenticating between neighbors, the same password must be configured on the

connecting interfaces.

- When authenticating between neighbors, authentication may be configured separately for L1 and L2 adjacencies.
- When authenticating between neighbors, either clear text or MD5 may be used.
- When performing area-wide authentication, every router in the area must use the same authentication mode and must have a common key-string.
- When performing domain-wide authentication, every L2 and L1/L2 router in the IS-IS domain must utilize the same mode of authentication and must use a common key-string.

The steps for configuring authentication are the same as for RIPv2 and EIGRP. These steps are repeated here:

Step 1. Define a key chain with a name.

Step 2. Define the key or keys on the key chain.

Step 3. Enable authentication on an interface or for the IS-IS instance at level-1 (area wide) or level-2 (domain wide), and specify the key chain to be used.

Step 4. Specify whether the interface or IS-IS instance level-1 or level-2 will use clear text or MD5 authentication.

Step 5. Optionally configure key management.

Authentication can be configured in the network without breaking adjacencies between routers. To accomplish this, the command isis authentication send-only must first be applied to all routers that will use authentication. This command is used on the interfaces to authenticate between neighbors and it is used under the ISIS routing process to authenticate area-wide or domain-wide. The command is removed after authentication is fully configured.

To authenticate between neighbors, the key chain is configured, the isis authentication key-chain command references the preconfigured key chain, then the isis authentication mode command is used to configure the type of authentication on the connected interfaces. L1 and L2 adjacencies may reference different key chains. Neighboring routers must share a common key-string or password. The key chains for either or both levels can be specified on an interface, and the key-strings for each level can be the same or different. When configured, the key-strings are carried in authentication information TLVs in the L1 or L2 Hellos between IS-IS neighbors.

For example, to authenticate between Geneva, Zurich, and Madrid in Figure 10-53, the configurations are as displayed in Example 10-32 through Example 10-34.

## Example 10-32. Geneva's authentication configuration.

```
interface FastEthernet0/0
 ip address 172.16.4.1 255.255.255.0
 ip router isis
 isis password magic
```

## Example 10-33. Zurich's authentication configuration.

```
key chain troll
 key 1
```

```
  key-string magic

key chain fairy
 key 1
  key-string dust

interface Ethernet0/0
 ip address 172.16.4.2 255.255.255.0
 ip router isis
 isis authentication mode text
 isis authentication key-chain troll level-1
!
interface Serial0/0.1 point-to-point
 ip address 172.16.8.2 255.255.255.0
 ip router isis
 isis authentication mode text
 isis authentication key-chain fairy level-2
```

## Example 10-34. Madrid's authentication configuration.

```
key chain fairy
 key 1
  key-string dust

interface Serial0/0.1 point-to-point
 ip address 172.16.8.1 255.255.255.0
 ip router isis
 isis authentication mode text
 isis authentication key-chain fairy level-2
```

Since the adjacency between Geneva and Zurich is L1, only a level 1 key-chain reference is specified. Between Zurich and Madrid, only an L2 adjacency exists, so a level 2 key-chain reference is used. Note that if no level-1 or level-2 keyword is specified, the isis password, isis authentication mode and isis authentication key-chain commands default to both L1 and L2.

Note Geneva's configuration. Geneva's interface connecting to Zurich is configured with the old style cleartext password. The password must be the same as the key-string defined on Zurich for the adjacencies to establish.[25]

[25] Passwords and/or key-strings must be identical between routers. They are case sensitive. Also, spaces and other characters are part of the passwords. If configuration files are created off-line, and pasted into the routers, be aware of trailing spaces.

To authenticate within an area, the isis authentication mode *mode* level-1 and isis authentication key-chain *name* level-1 commands are used to define the authentication mode and to reference a key chain under the IS-IS configuration. When the mode is text, the two commands together are used in place of the old style area-password command. Whereas the key-string specified at the interface level by the isis authentication command is carried in Hellos, the key-string specified by the level-1 authentication command for the IS-IS process is carried in all L1 LSPs, CSNPs, and PSNPs. So the neighbor-level authentication regulates the establishment of adjacencies, and the area-level authentication regulates the exchange of level 1 link-state information. If area authentication is not configured correctly, routers will still become adjacent but L1 LSPs will not be exchanged.

To configure area passwords in area 3 of Figure 10-53, the configurations of Bonn and Frankfurt are as displayed in Example 10-35 and Example 10-36.

## Example 10-35. Bonn's area password configuration.

```
Key chain river
 Key 1
  Key-string Rhine

router isis
 net 03.0000.0c0a.2aa9.00
 authentication key-chain river level-1
 authentication mode text level-1
 summary-address 172.16.16.0 255.255.248.0
```

## Example 10-36. Frankfurt's area password configuration.

```
router isis
 net 03.0000.0c04.dcc0.00
 is-type level-1
 area-password Rhine
```

Frankfurt's IOS does not support the new style authentication, so the area-password command is used. Note that the password is the same as the key-string defined on Bonn.

To authenticate domain-wide, the authentication key-chain and authentication mode commands are used with the level-2 keyword for the IS-IS process. These configuration commands are interoperable with the old style command, domain-password. Both styles of commands cannot be configured on the same router concurrently. The key-string defined in the key-chain referenced by these authentication commands is carried in L2 LSPs, CSNPs, and PSNPs. As a result, domain authentication regulates the exchange of L2 route information. Like area authentication, domain authentication does not authenticate L2 adjacencies, but does authenticate the exchange of L2 LSPs.

To configure domain authentication in the network of Figure 10-53, only Zurich, Madrid, and Bonn must be configured because Geneva and Frankfurt are L1 routers. The configurations are shown in Example 10-37 through Example 10-39.

## Example 10-37. Zurich's domain authentication configuration.

```
key chain troll
 key 1
  key-string magic

key chain fairy
 key 1
  key-string dust

key chain forest
 key 1
  key-string Blackforest

interface Ethernet0/0
 ip address 172.16.4.2 255.255.255.0
 ip router isis
 isis authentication mode text
 isis authentication key-chain troll level-1
!
interface Serial0/0.1 point-to-point
 ip address 172.16.8.2 255.255.255.0
```

```
 ip router isis
 isis authentication mode text
 isis authentication key-chain fairy level-2
!
router isis
 authentication key-chain forest level-2
 authentication mode md5 level-2
 net 01.0000.0c76.5b7c.00
 summary-address 172.16.0.0 255.255.248.0
```

Example 10-38. Madrid's domain authentication configuration.

```
key chain fairy
 key 1
  key-string dust
key chain forest
 key 1
  key-string Blackforest

interface Serial0/0.1 point-to-point
 ip address 172.16.8.1 255.255.255.0
 ip router isis
 isis authentication mode text
 isis authentication key-chain fairy level-2
!
router isis
 net 02.0000.3090.6756.00
 is-type level-2-only
 authentication key-chain forest level-2
 authentication mode md5 level-2
```

Example 10-39. Bonn's domain authentication configuration.

```
key chain river
 key 1
  key-string Rhine
key chain forest
 key 1
  key-string Blackforest
!
router isis
 net 03.0000.0c0a.2aa9.00
 authentication key-chain river level-1
 authentication key-chain forest level-2
 authentication mode text level-1
 authentication mode md5 level-2
 summary-address 172.16.16.0 255.255.248.0
```

## Case Study: A Basic Integrated IS-IS Configuration for IPv6

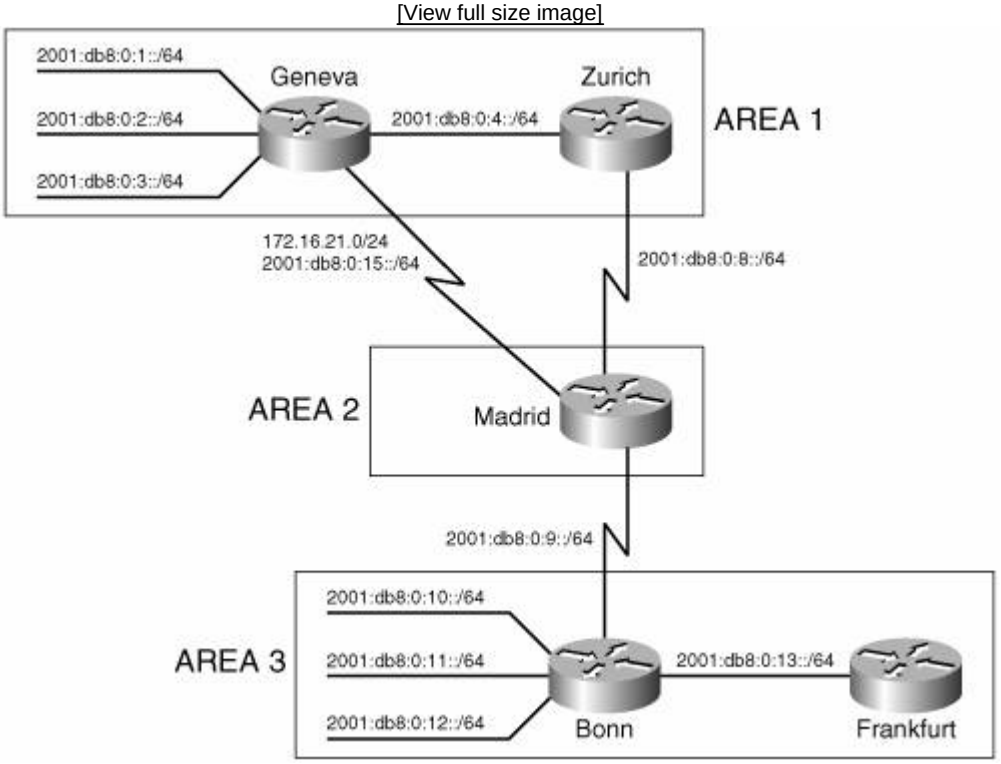
Integrated IS-IS computes a single SPF to create a single topology for both IPv4 and IPv6 (and CLNS). If both IPv4 and IPv6 are configured on the network, all interfaces and all routers must be configured with both protocols.

A link is added to the network in Figure 10-53, between Geneva and Madrid. Geneva is no longer

an L1-only router. IPv6 is added to the network also. The new addresses are shown in Figure 10-54. Integrated IS-IS is already configured on the routers. To enable IS-IS for IPv6, enable IPv6 routing globally with the command ipv6 unicast-routing, then enable IPv6 IS-IS on the interfaces. An IPv6 address and IPv6 IS-IS is added to every interface with an IPv4 address. Geneva's new configuration is displayed in Example 10-40.

Figure 10-54. IS-IS for IPv6 is added to the network.

[View full size image]




Example 10-40. Geneva's IPv6 IS-IS configuration.

```
Hostname Geneva
ipv6 unicast-routing
!
interface FastEthernet0/0
 ip address 172.16.4.1 255.255.255.0
 ip router isis
 isis password magic
 ipv6 address 2001:db8:0:4::1/64
 ipv6 router isis
!
interface s 0/0.1 point-to-point
 ip address 172.16.21.1 255.255.255.0
 ip router isis
 ipv6 address 2001:db8:0:15::1/64
 ipv6 router isis
!
interface FastEthernet0/1
 ip address 172.16.1.1 255.255.255.0
 ipv6 address 2001:db8:0:1::1/64
 ip router isis
 ipv6 router isis
```

```
!
interface FastEthernet0/2
 ip address 172.16.2.1 255.255.255.0
 ipv6 address 2001:db8:0:2::1/64
 ip router isis
 ipv6 router isis
!
interface FastEthernet0/3
 ip address 172.16.3.1 255.255.255.0
 ipv6 address 2001:db8:0:3::1/64
 ip router isis
 ipv6 router isis
!
router isis
 net 01.0004.c150.f1c0.00
 authentication mode md5
 authentication key-chain forest
```

The IS-IS routing process has not been changed. An IPv6 address has been added to each interface and IS-IS for IPv6 has been enabled on each interface. The other routers are configured similarly.

The IPv6 addresses in Figure 10-54 are summarized similarly to the IPv4 addresses. Zurich, Geneva, and Bonn summarize the routes as they are advertised to the L2 routers. IPv6 addresses are summarized under the global IS-IS routing process by specifying the IPv6 address family and configuring the address range to be summarized (see Example 10-41 through Example 10-43).

### Example 10-41. Zurich is summarizing IPv6 routes.

```
router isis
 address-family ipv6
 summary-prefix 2001:db8::/62
```

### Example 10-42. Geneva is summarizing IPv6 routes.

```
router isis
 address-family ipv6
 summary-prefix 2001:db8::/62
```

### Example 10-43. Bonn is summarizing IPv6 routes.

```
router isis
 address-family ipv6
 summary-prefix 2001:db8:0:10::/62
```

Example 10-44 shows Madrid's IPv6 route table, with Zurich's, Geneva's, and Bonn's summarized address ranges.

### Example 10-44. Madrid's IPv6 route table shows addresses summarized by Zurich, Geneva, and Bonn.

```
Madrid#show ipv6 route
IPv6 Routing Table - 12 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
I2  2001:DB8::/62 [115/20]
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.3
I2  2001:DB8:0:4::/64 [115/20]
     via FE80::2B0:64FF:FE30:1DE0, Serial0/0.1
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.3
C   2001:DB8:0:8::/64 [0/0]
     via ::, Serial0/0.1
L   2001:DB8:0:8::1/128 [0/0]
     via ::, Serial0/0.1
C   2001:DB8:0:9::/64 [0/0]
     via ::, Serial0/0.2
L   2001:DB8:0:9::1/128 [0/0]
     via ::, Serial0/0.2
C   2001:DB8:0:15::/64 [0/0]
     via ::, Serial0/0.3
L   2001:DB8:0:15::2/128 [0/0]
     via ::, Serial0/0.3
I2  2001:DB8:0:10::/62 [115/20]
     via FE80::205:5EFF:FE6B:50A0, Serial0/0.2
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Madrid#
```

IPv4 and IPv6 share the same topology and, therefore, the same metric values. IPv6 TLVs use extended metrics, but by default, IPv4 uses the narrow metric style with a maximum value of 63. The IPv6 metric is, therefore, also limited to 63. The default value for each interface is 10. The metric style can be changed using the `metric-style` command. The keywords `wide`, `transition`, or `wide transition` enable the router to send or accept narrow- and wide-style metrics during network reconfiguration. The type of metric used can be seen in the output of the command `show clns protocol`, displayed in Example 10-45.

## Example 10-45. Zurich is configured to use the default narrow metric style.

```
Zurich#show clns protocol
IS-IS Router: <Null Tag>
  System Id: 0000.0C76.5B7C.00 IS-Type: level-1-2
  Manual area address(es):
        01
  Routing for area address(es):
        01
  Interfaces supported by IS-IS:
        Serial0/0.1 - IP - IPv6
        Ethernet0/0 - IP - IPv6
  Redistribute:
    static (on by default)
  Distance for L2 CLNS routes: 110
  RRR level: none
  Generate narrow metrics: level-1-2
```

```
  Accept narrow metrics:   level-1-2
  Generate wide metrics:   none
  Accept wide metrics:     none
Zurich#
```

As seen in Example 10-45, Zurich sends and accepts narrow metrics only. Notice that the metric style is not specific for either IPv4 or IPv6.

Zurich's metric-style configuration is changed as in Example 10-46.

### Example 10-46. Zurich is configured with the metric style in transition mode, accepting and sending both wide and narrow metrics.

```
router isis
 metric-style transition
 address-family ipv6
  summary-prefix 2001:db8::/62
```

The keyword transition causes the router to send and accept wide metrics and narrow metrics. Example 10-47 shows the results on Zurich.

### Example 10-47. Both narrow and wide metrics are generated and accepted.

```
Zurich#show clns protocol

IS-IS Router: <Null Tag>
  System Id: 0000.0C76.5B7C.00 IS-Type: level-1-2
  Manual area address(es):
        01
  Routing for area address(es):
        01
  Interfaces supported by IS-IS:
        Serial0/0.1 - IP - IPv6
        Ethernet0/0 - IP - IPv6
  Redistribute:
    static (on by default)
  Distance for L2 CLNS routes: 110
  RRR level: none
  Generate narrow metrics: level-1-2
  Accept narrow metrics:   level-1-2
  Generate wide metrics:   level-1-2
  Accept wide metrics:     level-1-2
Zurich#
```

## Case Study: Transition to Multiple Topology Mode

A problem has arisen in the network shown in Figure 10-54. Frankfurt's IOS does not support IPv6. When IPv6 is configured on Bonn and the rest of the network, by default, IPv6 is added to the single IS-IS topology that is created for each level for IPv4. Because a single topology exists for each level, and the topology is shared by both IPv4 and IPv6, IPv4 and IPv6 must both be configured on each link and each router. If a single topology is used, and both IPv4 and IPv6 are configured, every link that has an IPv4 address must also have an IPv6 address. The problem can be seen on Bonn. When IPv6 was configured, the IS-IS adjacency with Frankfurt failed.

debug isis adj-packets, shown in Example 10-48, shows the problem with the IPv6 address.

## Example 10-48. *debug isis adj-packets* shows an error with the IPv6 addressing in single topology mode.

```
Bonn#debug is adj-packets fastethernet0/0
IS-IS Adjacency related packets debugging is on
Bonn#
ISIS-Adj: Sending L1 LAN IIH on FastEthernet0/0, length 1497
ISIS-Adj: Sending L2 LAN IIH on FastEthernet0/0, length 1497
ISIS-Adj: Rec L1 IIH from 0000.0c8d.34f1 (FastEthernet0/0), cir type L1, cir id
0000.0C0A.2AA9.01, length 1497
ISIS-Adj: No usable IPv6 linklocal addresses in LAN IIH from FastEthernet0/0
```

IOS supports multiple topology IS-IS in addition to the default single topology mode.[26] In single topology mode, IPv4, IPv6, and CLNS share the same IS-IS topology. When multi-topology is configured, IOS supports two topologies: one for IPv6, the other for IPv4 and CLNS. Individual SPF are computed for each topology.

[26] Multiple topology support is available in IOS 12.2(15)T, 12.2(18)S, 12.0(26)S, 12.3, 12.3(2)T, and later.

As explained earlier in the section "Multiple Topologies," single topology IS-IS and multiple topology IS-IS use different TLVs to exchange information. A router that does not support multi-topology will not interpret multi-topology TLVs. A multi-topology router will process single-topology TLVs: If no multi-topology TLVs are received over an adjacency, the receiving router assumes the neighbor to be part of the IPv4/CLNS topology. This could be problematic when single topology mode and IPv6 are already configured in the network and you want to migrate to multiple topologies. During the migration, areas of the IPv6 network could become unreachable.

IOS provides a transition mode, in which both single topology and multi-topology TLVs are sent, but the routers operate only in single topology mode. After all routers are configured, transition mode can be removed.

The use of extended metrics, which are used in multi-topology TLVs, must be configured on the router before multi-topology can be configured.

Multi-topology is configured in the network shown in Figure 10-54. All routers are first configured in transition mode, with the configuration in Example 10-49.

## Example 10-49. All routers in Figure 10-54 have this configuration.

```
router isis
 metric-style wide transition
 address-family ipv6
  multi-topology transition
```

Frankfurt does not support IPv6 or multi-topology mode. The only configuration change on Frankfurt is to change to the wide metric-style. Since no multi-topology TLVs are sent from Frankfurt, the Frankfurt router and its links remain part of the default IPv4 topology.

Two commands are needed to change from single topology mode to multi-topology mode: The first command changes the metric style from narrow to wide, and the second command enables multi-topology mode. The new topology shares the NET and the L1/L2 boundaries with the IPv4 topology.

The two different topologies can be seen on Bonn. Bonn's IS-IS IPv6 topology shows IPv6 paths to L1 routers and L2 routers. There are two entries for L1 routers. One is for Bonn itself. The

other is for Frankfurt. The asterisks mean that there is no IPv6 path to this router, but a link does exist. Compare the IPv6 topology with the IPv4 topology. Example 10-50 shows the output of show isis ipv6 topology and show isis topology. The command show isis topology displays the default, IPv4 topology.

## Example 10-50. The IPv4 topology and the IPv6 topology are displayed on the Bonn router.

```
Bonn#show isis ipv6 topology

IS-IS IPv6 paths to level-1 routers
System Id            Metric     Next-Hop             Interface   SNPA
Bonn                 --
Frankfurt            **
IS-IS IPv6 paths to level-2 routers
System Id            Metric     Next-Hop             Interface   SNPA
Bonn                 --
Zurich               20         Madrid               Se0/0.1     DLCI 171
Madrid               10         Madrid               Se0/0.1     DLCI 171
Geneva               20         Madrid               Se0/0.1     DLCI 171
Bonn#show isis topology
IS-IS paths to level-1 routers
System Id            Metric     Next-Hop             Interface   SNPA
Bonn                 --
Frankfurt            10         Frankfurt            Fa0/0       0000.0c8d.34f1
IS-IS paths to level-2 routers
System Id            Metric     Next-Hop             Interface   SNPA
Bonn                 --
Zurich               20         Madrid               Se0/0.1     DLCI 171
Madrid               10         Madrid               Se0/0.1     DLCI 171
Geneva               20         Madrid               Se0/0.1     DLCI 171
Bonn#
```

A single adjacency is formed over point-to-point links if at least one topology exists. The show clns is-neighbors detail command in Example 10-51 shows that Bonn has one adjacency on Serial0/0.1 to Madrid, but that two topologies share that adjacency, IPv4 and IPv6.

## Example 10-51. A single adjacency is formed between Bonn and Madrid, shared by both the IPv4 and the IPv6 topologies.

```
Bonn#show clns is-neighbors detail
System Id      Interface   State  Type Priority  Circuit Id         Format
Madrid         Se0/0.1     Up     L2   0         00                 Phase V
  Area Address(es): 03
  IP Address(es):  172.16.9.1*
  IPv6 Address(es): FE80::204:C1FF:FE50:E700
  Uptime: 00:23:14
  NSF capable
  Topology: IPv4, IPv6
Frankfurt      Fa0/0       Up     L1   64        Bonn.01            Phase V
  Area Address(es): 03
  IP Address(es): 172.16.19.2*
  Uptime: 00:23:17
Bonn#
```

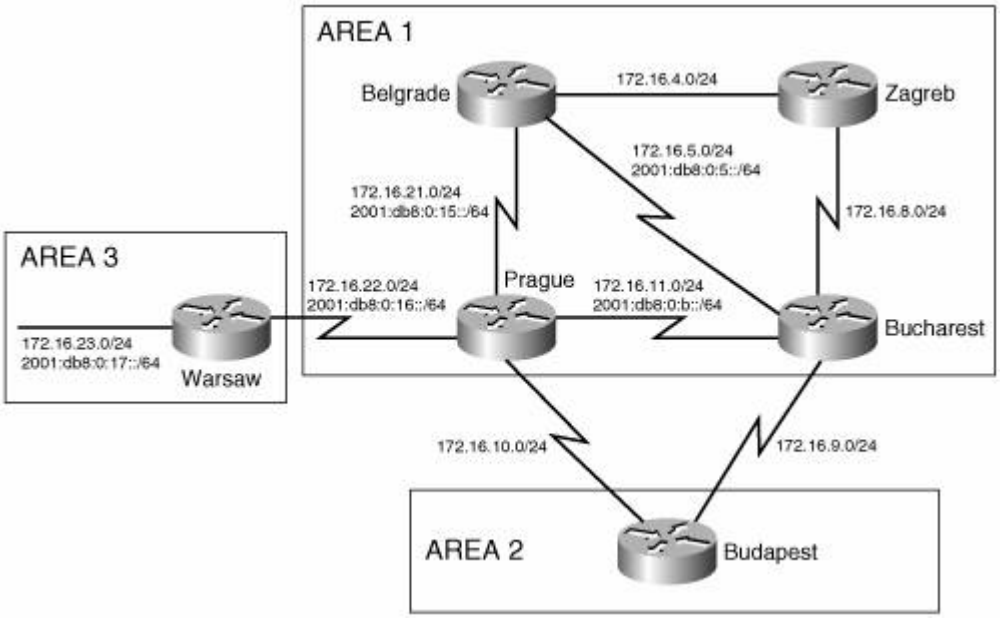
## Case Study: Route Leaking Between Levels

Recall that when an L1/L2 router sends its L1 LSP into an area, it signals other L1 routers that it can reach another area by setting the Attached (ATT) bit in the LSP. The L1 routers forward packets to the nearest L2 router (measured as the least cost path to a router with the ATT bit set) when forwarding packets out of the area.

Consider the network in Figure 10-55. In area 1, both Prague and Bucharest are L1/L2 routers. They both set the ATT bit on L1 updates sent to Belgrade and Zagreb, both L1 routers. Example 10-52 is a display of Belgrade's IS database.

Figure 10-55. A network with multiple L1/L2 routers in area 1 could result in inefficient routing from area 1 to other areas.

[View full size image]




Example 10-52. The IS-IS database of an L1 router in area 1 shows two L1/L2 routers with the ATT bit set.

```
Belgrade#show is database

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Bucharest.00-00       0x00000052   0xE802        1104              1/0/0
Prague.00-00          0x00000052   0xEDC1        1099              1/0/0
Zagreb.00-00          0x0000003B   0x13A5        1076              0/0/0
Belgrade.00-00      * 0x00000050   0x5EC1        485               0/0/0
Belgrade#
```

Belgrade's IS-IS database shows all routers in area 1. Prague and Bucharest are both L1/L2 routers and, therefore, have set the ATT bit. Recall that L1 routers use the closest L1/L2 router to forward traffic out of the area. Both L1/L2 routers are equal distance to Belgrade. Belgrade's IPv4 route table (Example 10-53) and IPv6 route table (Example 10-54) show that the both Bucharest and Prague will be used to forward traffic out of the area.

Example 10-53. The IPv4 route table on the L1 router Belgrade shows two paths out of the area, indicated by two default routes.

```
Belgrade#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 172.16.5.1 to network 0.0.0.0
     172.16.0.0/24 is subnetted, 8 subnets
C       172.16.21.0 is directly connected, Serial0/0.1
i L1    172.16.22.0 [115/20] via 172.16.21.2, Serial0/0.1
i L1    172.16.8.0 [115/20] via 172.16.4.2, Serial0/0.2
                   [115/20] via 172.16.5.1, Serial0/0.3
i L1    172.16.9.0 [115/20] via 172.16.5.1, Serial0/0.3
i L1    172.16.10.0 [115/20] via 172.16.21.2, Serial0/0.1
i L1    172.16.11.0 [115/20] via 172.16.21.2, Serial0/0.1
                    [115/20] via 172.16.5.1, Serial0/0.3
C       172.16.4.0 is directly connected, Serial0/0.2
C       172.16.5.0 is directly connected, Serial0/0.3
i*L1 0.0.0.0/0 [115/10] via 172.16.5.1, Serial0/0.3
               [115/10] via 172.16.21.2, Serial0/0.1
Belgrade#
```

Example 10-54. The IPv6 route table on the L1 router Belgrade shows two paths out of the area, indicated by two default routes.

```
Belgrade#show ipv6 route
IPv6 Routing Table - 9 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
I1  ::/0 [115/10]
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.3
C   2001:DB8:0:5::/64 [0/0]
     via ::, Serial0/0.3
L   2001:DB8:0:5::2/128 [0/0]
     via ::, Serial0/0.3
I1  2001:DB8:0:B::/64 [115/20]
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.3
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
C   2001:DB8:0:15::/64 [0/0]
     via ::, Serial0/0.1
L   2001:DB8:0:15::1/128 [0/0]
     via ::, Serial0/0.1
I1  2001:DB8:0:16::/64 [115/20]
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Belgrade#
```

As you can see in Example 10-53 and Example 10-54, only L1 routes are in Belgrade's route table. All addresses outside of the area follow the path indicated by the default route. The default routes (IPv4: 0.0.0.0/0, IPv6: ::/0) show two next-hop addresses, one via Bucharest (S0/0.3) and one via Prague (S0/0.1).

172.16.23.0/24 and 2001:db8:0:17::/64 are addresses connected to the Warsaw router, in area 3. The best path for Belgrade to take to these addresses is via Prague. However, because both Bucharest and Prague are equal distance L1/L2 routers, Belgrade will load share between the L1/L2 routers when sending traffic to 172.16.23.0/24 or 2001:db8:0:17::/64. Half of the traffic will take the optimal path, the other half will take a longer path. Example 10-55 shows Belgrade pinging 172.16.23.1 and recording the route. One route is via Bucharest (172.16.11.2), the other is via Prague (172.16.22.2).

## Example 10-55. Ping with the record route option shows the round trip path a packet takes from Belgrade to Warsaw.

```
Belgrade#ping
Protocol [ip]:
Target IP address: 172.16.23.1
Extended commands [n]: y
Loose, Strict, Record, Timestamp, Verbose[none]: r
Loose, Strict, Record, Timestamp, Verbose[RV]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.23.1, timeout is 2 seconds:
<...>Reply to request 0 (416 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
   (172.16.5.2)
   (172.16.11.2)
   (172.16.22.2)
   (172.16.23.1)
   (172.16.22.1)
   (172.16.21.2)
   (172.16.5.2) <*>
   (0.0.0.0)
   (0.0.0.0)
 End of list
Reply to request 1 (216 ms).  Received packet has options
 Total option bytes= 40, padded length=40
 Record route:
   (172.16.21.1)
   (172.16.22.2)
   (172.16.23.1)
   (172.16.22.1)
   (172.16.21.2)
   (172.16.21.1) <*>
   (0.0.0.0)
   (0.0.0.0)
   (0.0.0.0)
End of list
```

The path that the traffic takes from Belgrade to 172.16.23.0 can be controlled by advertising the address to the L1 routers in area 1. This advertising of L2 routes to L1 router is called route leaking. The downside of route leaking is that the L1 router needs to keep track of additional addresses and paths to those addresses.

Route leaking is configured by creating a list containing the addresses to be advertised into the area, and then configuring the distribution of this list to the L1 routers. Prague's configuration is displayed in Example 10-56.

## Example 10-56. Prague is configured to leak routes L2 routes into the L1 area.

```
access-list 100 permit ip 172.16.23.0 0.0.0.255 any
ipv6 prefix-list warsaw seq 5 permit 2001:DB8:0:17::/64
router isis
 redistribute isis ip level-2 into level-1 distribute-list 100
 address-family ipv6
  redistribute isis level-2 into level-1 distribute-list warsaw
 exit-address-family
```

The access list number 100 defines the IPv4 addresses to be advertised and the prefix-list named warsaw defines the IPv6 addresses to be advertised. The redistribute isis ip command looks for L2 IPv4 addresses that match the list 100 and advertises them to L1 routers within the area. The redistribute isis command listed under the address-family ipv6 command looks for L2 IPv6 addresses that match those defined in the prefix-list named warsaw, and advertises them to the L1 area. Access-lists are discussed in Appendix B, and route redistribution is discussed in more detail in Chapter 11.

Leaked routes are advertised the same way as other addresses: in type 128, 130, or 135 IPv4 Reachability TLVs, and type 236 and 237 IPv6 Reachability TLVs. TLV 128 and 130 are used with narrow-type metrics, and 135 is used with wide metrics. To distinguish leaked routes from other connected IP routes or external IP routes an Up/Down bit is defined, as discussed previously in the section "Domain-Wide Prefix Distribution." If the Up/Down bit is set to 0, the route originated in this L1 area. If the Up/Down bit is set to 1, the route was leaked from L2. Setting this bit helps keeps route loops from occurring. A L1/L2 router will not advertise an L1 address that has the Up/Down bit set to another L2 router.

A route that was leaked into L1 is identifiable in the route table and in the IS-IS database. Example 10-57 and Example 10-58 display Belgrade's IPv4 and IPv6 route tables, respectively, and Example 10-59 displays Belgrade's IS-IS database.

## Example 10-57. The leaked address in the route table is identified as an "ia" route, or inter-area route.

```
Belgrade#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 172.16.5.1 to network 0.0.0.0
     172.16.0.0/24 is subnetted, 9 subnets
C       172.16.21.0 is directly connected, Serial0/0.1
i L1    172.16.22.0 [115/20] via 172.16.21.2, Serial0/0.1
i ia    172.16.23.0 [115/30] via 172.16.21.2, Serial0/0.1
i L1    172.16.8.0 [115/20] via 172.16.4.2, Serial0/0.2
                   [115/20] via 172.16.5.1, Serial0/0.3
i L1    172.16.9.0 [115/20] via 172.16.5.1, Serial0/0.3
i L1    172.16.10.0 [115/20] via 172.16.21.2, Serial0/0.1
i L1    172.16.11.0 [115/20] via 172.16.21.2, Serial0/0.1
```

```
                    [115/20] via 172.16.5.1, Serial0/0.3
C       172.16.4.0 is directly connected, Serial0/0.2
C       172.16.5.0 is directly connected, Serial0/0.3
i*L1 0.0.0.0/0 [115/10] via 172.16.5.1, Serial0/0.3
               [115/10] via 172.16.21.2, Serial0/0.1
Belgrade#
```

Example 10-58. The leaked IPv6 address in the route table is identified as "IA," inter-area.

```
Belgrade#show ipv6 route
IPv6 Routing Table - 10 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
I1  ::/0 [115/10]
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.3
C   2001:DB8:0:5::/64 [0/0]
     via ::, Serial0/0.3
L   2001:DB8:0:5::2/128 [0/0]
     via ::, Serial0/0.3
I1  2001:DB8:0:B::/64 [115/20]
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.3
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
C   2001:DB8:0:15::/64 [0/0]
     via ::, Serial0/0.1
L   2001:DB8:0:15::1/128 [0/0]
     via ::, Serial0/0.1
I1  2001:DB8:0:16::/64 [115/20]
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
IA  2001:DB8:0:17::/64 [115/30]
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Belgrade#
```

Example 10-59. The leaked address in the IS-IS database is identified as an IP-inter-area address.

```
Belgrade#show is database Prague.00-00 detail l1
IS-IS Level-1 LSP Prague.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Prague.00-00          0x0000006D   0x7A37        655               1/0/0
  Area Address: 01
  Topology:     IPv4 (0x0) IPv6 (0x4002 ATT)
  NLPID:        0xCC 0x8E
  Hostname: Prague
  IP Address:   172.16.11.1
  Metric: 10         IP 172.16.22.0/24
  Metric: 10         IP 172.16.21.0/24
  Metric: 10         IP 172.16.10.0/24
  Metric: 10         IP 172.16.11.0/24
```

```
  IPv6 Address: 2001:DB8:0:15::2
  Metric: 10         IPv6 (MT-IPv6) 2001:DB8:0:16::/64
  Metric: 10         IPv6 (MT-IPv6) 2001:DB8:0:15::/64
  Metric: 10         IPv6 (MT-IPv6) 2001:DB8:0:B::/64
  Metric: 10         IS-Extended Belgrade.00
  Metric: 10         IS-Extended Bucharest.00
  Metric: 10         IS (MT-IPv6) Belgrade.00
  Metric: 10         IS (MT-IPv6) Bucharest.00
  Metric: 20         IP-Interarea 172.16.23.0/24
  Metric: 20         IPv6-Interarea (MT-IPv6) 2001:DB8:0:17::/64
Belgrade#
```

The route tables display the IPv4 and IPv6 routes as IS-IS inter-area routes, designated by the type "ia" for IPv4 and "IA" for IPv6. The database entry for the router that originated the leak shows the addresses as IP-Interarea and IPv6-Interarea.

## Case Study: Multiple L1 Areas Active on a Router

IOS supports the ability to have multiple, active, L1 areas on a single router: Up to 29 L1 areas can be configured, but only one L2 area. By default, the first configured IS-IS routing process defines the L2 area. It can also define a single L1 area. To enable additional L1 areas on the router, additional IS-IS routing processes are defined. Routers within each of the configured L1 areas can communicate together via this L1/L2 router. Because each area does not have to have a unique L1/L2 router to reach the rest of the network, the number of L1/L2 routers is minimized.

In the network of Figure 10-55, Warsaw is in area 3. The Warsaw router is an L1/L2 router for the area. The only connection from Warsaw to the rest of the network is via Prague. Prague's configuration is modified so that Prague becomes the L2 router for both areas 1 and 3. Prague's new configuration is displayed in Example 10-60.

### Example 10-60. Prague is configured with multiple L1 areas.

```
router isis
 net 01.0004.c150.e700.00
 metric-style wide
 address-family ipv6
  multi-topology

router isis three
 net 03.0004.c150.e700.00
 metric-style wide
 address-family ipv6
  multi-topology

interface Serial0/0.1 point-to-point
 description Warsaw
 ip address 172.16.22.2 255.255.255.0
 ipv6 address 2001:db8:0:16::2/64
 ip router isis three
 ipv6 router isis three
```

The second IS-IS routing process is called "three." IS-IS three is assigned to the interface connecting Prague to Warsaw, interface Serial 0/0.1. IS-IS "three" is automatically configured as a L1 area, as seen by the show clns neighbor command on Prague (Example 10-61).

## Example 10-61. Multiple L1 areas are configured on Prague.

```
Prague#show clns neighbor
Area null:
System Id      Interface   SNPA                State  Holdtime  Type Protocol
Budapest       Se0/0.3     DLCI 115            Up     24        L2   IS-IS
Belgrade       Se0/0.2     DLCI 116            Up     26        L1   M-ISIS
Bucharest      Se0/0.4     DLCI 113            Up     22        L1L2 M-ISIS
Area three:
System Id      Interface   SNPA                State  Holdtime  Type Protocol
Warsaw         Se0/0.1     DLCI 117            Up     28        L1   M-ISIS
Prague#
```

Warsaw can now be configured as a L1 router, because all its neighbors exist in area 3 (see Example 10-62). In this simple network, this step is not needed because Warsaw's only connection outside area 3 is to Prague, which only uses L1 communication with Warsaw. This step might be desired in larger networks, to ensure that Warsaw does not attempt L2 connections to other routers.

## Example 10-62. Warsaw is configured as an L1 router.

```
router isis
 is-type level-1
```

Example 10-63 shows Warsaw's IS-IS database and the route table as a L1/L2 router. Example 10-64 shows the same tables with Warsaw as a L1 router.

## Example 10-63. Warsaw's database includes entries for area 3 routers and all L2 routers.

```
Warsaw#show is database
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Prague.00-00          0x0000000A   0x08B1        1048              1/0/0
Warsaw.00-00        * 0x00000075   0x3D13        1085              1/0/0
IS-IS Level-2 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Bucharest.00-00       0x0000007A   0x59D0        1060              0/0/0
Prague.00-00          0x00000090   0x433A        1078              0/0/0
Warsaw.00-00        * 0x00000001   0xA828        1080              0/0/0
Budapest.00-00        0x00000073   0xFE95        344               0/0/0
Budapest.01-00        0x00000066   0xFF8B        444               0/0/0
Warsaw#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     172.16.0.0/24 is subnetted, 10 subnets
i L2    172.16.21.0 [115/20] via 172.16.22.2, Serial0/0.1
C       172.16.22.0 is directly connected, Serial0/0.1
```

```
C       172.16.23.0 is directly connected, FastEthernet0/0
i L2    172.16.12.0 [115/30] via 172.16.22.2, Serial0/0.1
i L2    172.16.8.0 [115/30] via 172.16.22.2, Serial0/0.1
i L2    172.16.9.0 [115/30] via 172.16.22.2, Serial0/0.1
i L2    172.16.10.0 [115/20] via 172.16.22.2, Serial0/0.1
i L2    172.16.11.0 [115/20] via 172.16.22.2, Serial0/0.1
i L2    172.16.4.0 [115/30] via 172.16.22.2, Serial0/0.1
i L2    172.16.5.0 [115/30] via 172.16.22.2, Serial0/0.1
Warsaw#
```

Example 10-64. Warsaw's database includes only area 3 routers and links, and its route tables contain only a default route for addresses not in area 3.

```
Warsaw#show is database

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Prague.00-00          0x00000009   0x2BDF        923               1/0/0
Warsaw.00-00        * 0x00000073   0x6CF3        924               0/0/0
Warsaw#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 172.16.22.2 to network 0.0.0.0
     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.22.0 is directly connected, Serial0/0.1
C       172.16.23.0 is directly connected, FastEthernet0/0
i*L1 0.0.0.0/0 [115/10] via 172.16.22.2, Serial0/0.1
Warsaw#
```

When Warsaw was the L1/L2 router for area 3, the IS database contained entries for each router in area 3 and entries for each L2 router and all the addresses those L2 routers could reach. The route table contained entries for every reachable address in the network. As an L1 router, Warsaw has reduced the size of its IS-IS database and its route table significantly.

PREV NEXT

# Troubleshooting Integrated IS-IS

The basic methods for troubleshooting IS-IS are very similar to the methods for troubleshooting C discussed in Chapter 8. A major aspect of troubleshooting Integrated IS-IS that is different from t IP routing protocols is that IS-IS uses CLNS PDUs rather than IP packets. If you are troubleshootir protocol itself, remember that you are troubleshooting CLNS, not IP.

As with all routing protocols, the first troubleshooting step is to check the route table for accurate information. If an expected route entry is missing or incorrect, the remainder of the troubleshootir to determine the source of the problem.

After the route table, the link-state database is the most important source of troubleshooting infor As recommended in Chapter 8, it is good practice to keep a copy of the L1 link-state database for area, and a copy of the L2 link-state database. These stored copies of the databases should be up regularly, as part of your routine baselining procedures. When things go wrong, these stored datal provide a steady-state reference.

When examining an individual router's configuration, consider the following:

- Does the `net` statement under the IS-IS configuration specify the correct NET? Are the Area System ID correct for this router? Does the NET comply with whatever CLNS addressing conv being used in this network?
- Is IS-IS enabled on the correct interfaces with the `ip router isis` or `ipv6 router isis` comma
- Are the IP addresses and subnet masks or prefixes correct? It is doubly important to check tl an Integrated IS-IS environment because a misconfigured IP address will not prevent an IS- adjacency from being partially established.

## Troubleshooting IS-IS Adjacencies

The command `show clns is-neighbors` displays the IS-IS neighbor table. The entire table is disp default, or you can specify a particular interface. From this table, you can observe whether all exp neighbors are present and whether they are the correct type. For more information, such as the a addresses and IP addresses associated with each neighbor and the uptime of each neighbor, use t `clns is-neighbors detail` command.

When examining adjacencies, consider the following:

- Are the router levels configured correctly? L1 routers can establish adjacencies only with L1 a L1/L2 routers, and L2 routers can establish adjacencies only with L2 and L1/L2 routers.
- Are Hellos being sent from both neighbors? Are the Hellos the correct level, and do they cont correct parameters? The command `debug isis adj-packets` is useful for observing Hellos.
- If authentication is being used, are the passwords and authentication mode the same betwee neighbors? Remember that area (level 1) and domain (level 2) authentication do not regulat adjacencies, only the exchange of LSPs.
- Are any access lists blocking IS-IS or CLNS?

A change has been made on the router Bonn in Figure 10-54. Bonn and Frankfurt are no longer ac via IPv4 from other locations. A look at Bonn's IPv4 route table (Example 10-65) shows that there IP routes learned via Madrid (interface Serial0/0.1).

Example 10-65. Bonn's IPv4 route table shows IS-IS learned routes fro
Frankfurt on FA0/0, but nothing from Madrid on Serial0/0.1.

```
Bonn#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 8 subnets, 2 masks
C       172.16.24.0/24 is directly connected, Loopback1
C       172.16.25.0/24 is directly connected, Loopback2
i L1    172.16.16.0/24 [115/20] via 172.16.19.2, FastEthernet0/0
i su    172.16.16.0/21 [115/10] via 0.0.0.0, Null0
i L1    172.16.17.0/24 [115/20] via 172.16.19.2, FastEthernet0/0
i L1    172.16.18.0/24 [115/20] via 172.16.19.2, FastEthernet0/0
C       172.16.19.0/24 is directly connected, FastEthernet0/0
C       172.16.9.0/24 is directly connected, Serial0/0.1
Bonn#
```

The command show clns is-neighbors shows an adjacency exists from Bonn to Madrid (Example
However, notice Madrid's type is IS rather than L1 or L2. This indicates a problem with the adjacer
debug isis adj-packets gives a hint as to where the problem exists (Example 10-67).

Example 10-66. Bonn's CLNS IS-IS neighbor table shows an adjacency e
to Madrid and to Frankfurt, but there is something wrong with the Madr
adjacency.

```
Bonn#show clns is-neighbors
System Id      Interface   State  Type Priority  Circuit Id          Format
Madrid         Se0/0.1     Up     IS   0         00                  Phase V
Frankfurt      Fa0/0       Up     L1   64        Bonn.03             Phase V
Bonn#
```

Example 10-67. The details of IS-IS Hellos (IIHs) can be observed with
*debug isis adj-packets* command. This display is from router Bonn in Fig
10-54 and shows that there is a problem with the IP address on Serial0

```
Bonn#debug isis adj-packets
IS-IS Adjacency related packets debugging is on
Bonn#
ISIS-Adj: Sending serial IIH on Serial0/0.1, length 1499
ISIS-Adj: Rec serial IIH from DLCI 171 (Serial0/0.1), cir type L2, cir id 01, lengt
ISIS-Adj: No usable IP interface addresses in serial IIH from Serial0/0.1
ISIS-Adj: Sending L1 LAN IIH on FastEthernet0/0, length 1497
ISIS-Adj: Sending L2 LAN IIH on FastEthernet0/0, length 1497
ISIS-Adj: Sending L1 LAN IIH on FastEthernet0/0, length 1497
ISIS-Adj: Rec L1 IIH from 0000.0c8d.34f1 (FastEthernet0/0), cir type L1, cir id
0000.0C0A.2AA9.03, length 1497
```

Reviewing the IPv4 addresses on Bonn shows that the IP address on Serial0/0.1 have been inadve changed. IOS checks the addresses in the TLVs before establishing adjacencies. If the IPv4 addres Interface Address TLV, 132, does not belong to the same subnet as the receiving interface, an adj formed, but the type of the adjacency will be IS rather then L1 or L2, indicating that there is a pro with the configuration affecting the adjacency. After correcting the problem, Bonn's CLNS IS neigh table (Example 10-68) and IPv4 route table look good.

## Example 10-68. A problem free CLNS neighbor table shows the adjacenc is L1, L2, or L1/L2. Bonn's route table shows IP routes from both adjace neighbors.

```
Bonn#show clns is-neighbors
System Id      Interface   State  Type Priority  Circuit Id         Format
Madrid         Se0/0.1     Up     L2   0         00                 Phase V
Madrid         Fa0/0       Up     L1   64        Bonn.03            Phase V
Bonn#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 11 subnets, 2 masks
C       172.16.24.0/24 is directly connected, Loopback1
C       172.16.25.0/24 is directly connected, Loopback2
i L2    172.16.21.0/24 [115/20] via 172.16.9.1, Serial0/0.1
i L1    172.16.16.0/24 [115/20] via 172.16.19.2, FastEthernet0/0
i su    172.16.16.0/21 [115/10] via 0.0.0.0, Null0
i L1    172.16.17.0/24 [115/20] via 172.16.19.2, FastEthernet0/0
i L1    172.16.18.0/24 [115/20] via 172.16.19.2, FastEthernet0/0
C       172.16.19.0/24 is directly connected, FastEthernet0/0
i L2    172.16.8.0/24 [115/20] via 172.16.9.1, Serial0/0.1
C       172.16.9.0/24 is directly connected, Serial0/0.1
i L2    172.16.0.0/21 [115/30] via 172.16.9.1, Serial0/0.1
Bonn#
```

The output from the command debug isis adj-packets performed on a problem free Bonn is sho Example 10-69. Information is being received over Serial0/0.1.

## Example 10-69. The details of IS-IS Hellos (IIHs) are observed on a cor configured router, Bonn.

```
Bonn#debug isis adj-packets
IS-IS Adjacency related packets debugging is on
Bonn#
ISIS-Adj: Sending L1 LAN IIH on FastEthernet0/0, length 1497
ISIS-Adj: Sending L2 LAN IIH on FastEthernet0/0, length 1497
ISIS-Adj: Rec L1 IIH from 0000.0c8d.34f1 (FastEthernet0/0), cir type L1, cir id
0000.0C0A.2AA9.03, length 1497
ISIS-Adj: Rec serial IIH from DLCI 171 (Serial0/0.1), cir type L2, cir id 01, lengt
ISIS-Adj: Local mode (IP, IPv6), remote mode (IP, IPv6)
ISIS-Adj: rcvd state UP, old state UP, new state UP
ISIS-Adj: Action = ACCEPT
ISIS-Adj: Sending L1 LAN IIH on FastEthernet0/0, length 1497l
ISIS-Adj: Sending serial IIH on Serial0/0.1, length 1499
```

```
Bonn#
```

A similar adjacency problem exists with IPv6 in single topology mode, if IPv6 is not configured on link that is configured with IPv4. When the network in Figure 10-54 was configured with IPv6 in si topology mode, a problem arose between Bonn and Frankfurt. Frankfurt does not support IPv6. Ex 10-48 shows the output from the debug is adj-packets on Bonn. The debug's output shows that a problem with the link-local address: "No usable IPv6 linklocal addresses in LAN IIH from FastEthernet0/0." IOS on Bonn checks for valid addresses and address families before establishing adjacencies. In single topology mode, if both IPv4 and IPv6 are configured on the router, every lin has an IPv4 address must also have an IPv6 address, every link that has an IPv6 address must als an IPv4 address, and neighbor routers must also have valid IPv4 and IPv6 addresses. TLV 232 cor the IPv6 Link Local address in Hello packets. If in single topology mode, a neighbor router does no a valid IPv6 link-local address in its hello packets, the adjacency will be formed as a type IS (agai indicating a problem with the configuration).

There is an IS-IS process configuration command, log-adjacency-changes. This command will lo adjacency changes to the log, whether buffered or sent to a log server. Perusing the log is very us troubleshooting adjacency problems that occurred at some time in the past.

## Troubleshooting the IS-IS Link-State Database

The IS-IS link-state database is examined with the command show isis database. If the router is L1/L2, both of the databases are displayed by default. To observe only one of the databases, use t level-1 or level-2 keywords. To examine the LSPs in more detail, use the detail keyword. A singl can be observed by specifying its LSPID, as in Example 10-70.

### Example 10-70. This LSP is from the L2 database of router Zurich in Figu 10-54.

```
Zurich#show isis database detail Madrid.00-00
IS-IS Level-2 LSP Madrid.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Madrid.00-00          0x0000007C   0x5874        1023              0/0/0
  Auth:         Length: 17
  Area Address: 03
  Topology:     IPv4 (0x0) IPv6 (0x2)
  NLPID:        0xCC 0x8E
  Hostname: Madrid
  IP Address: 172.16.21.2
  IPv6 Address: 2001:DB8:0:15::2
  Metric: 10         IS-Extended Zurich.00
  Metric: 10         IS-Extended Bonn.00
  Metric: 10         IS-Extended Geneva.00
  Metric: 10         IS (MT-IPv6) Zurich.00
  Metric: 10         IS (MT-IPv6) Bonn.00
  Metric: 10         IS (MT-IPv6) Geneva.00
  Metric: 10         IP 172.16.21.0/24
  Metric: 10         IP 172.16.8.0/24
  Metric: 10         IP 172.16.9.0/24
  Metric: 10         IPv6 (MT-IPv6) 2001:DB8:0:8::/64
  Metric: 10         IPv6 (MT-IPv6) 2001:DB8:0:9::/64
  Metric: 10         IPv6 (MT-IPv6) 2001:DB8:0:15::/64
Zurich#
```

A sequence number that is noticeably higher than the sequence numbers of other LSPs might indi instability either within the area or on the level 2 backbone. Another hint of instability is an LSP H

that never gets very small. If instability is suspected, use the command show isis spf-log to list calculations recently performed by the router.

In Example 10-71, the SPF log for router Geneva in Figure 10-54 is shown. Nothing but the period calculations triggered by the 15-minute database refreshes are shown until approximately three m before the log was displayed. At that time, frequent SPF calculations began occurring, indicating fr changes of some sort in the network.[27]

[27] The first four triggering events were caused by "bouncing" Zurich's serial interface several times, and the next three events caused by deleting and then misconfiguring the link password. The last event was caused when the correct password was co

## Example 10-71. This SPF log reveals instability in area 1 of Figure 10-54

```
Geneva#sh isis spf-log
     Level 1 SPF log
   When      Duration   Nodes   Count    Last trigger LSP     Triggers
02:43:09          12       3       1                          PERIODIC
02:28:08          12       3       1                          PERIODIC
02:13:06          12       3       1                          PERIODIC
01:58:05          12       3       1                          PERIODIC
01:43:03          12       3       1                          PERIODIC
01:28:02          12       3       1                          PERIODIC
01:13:00          12       3       1                          PERIODIC
00:57:59          12       3       1                          PERIODIC
00:42:58          12       3       1                          PERIODIC
00:27:56          12       3       1                          PERIODIC
00:12:55          12       3       1                          PERIODIC
00:03:08          8        3       1   0000.0C76.5B7C.00-00  LSPHEADER
00:02:35          8        3       1   0000.0C76.5B7C.00-00  LSPHEADER
00:02:23          8        3       1   0000.0C76.5B7C.00-00  LSPHEADER
00:01:50          8        3       1   0000.0C76.5B7C.00-00  LSPHEADER
00:01:14          4        1       1   0000.0C0A.2C51.00-00  TLVCONTENT
00:00:46          4        2       2   0000.0C0A.2C51.04-00  NEWLSP TLVCONTENT
00:00:20          4        1       3   0000.0C0A.2C51.00-00  NEWADJ TLVCONTENT
00:00:08          8        3       1   0000.0C76.5B7C.02-00  TLVCONTENT
Geneva#
```

To further investigate instabilities revealed by the SPF log, three useful debug commands are avai Example 10-72, Example 10-73, and Example 10-74 show output from these three debug function each case, the debug messages show the results of disconnecting and reconnecting the serial inte Bonn in Figure 10-54 from the perspective of Frankfurt. The first, debug isis spf-triggers (Exam 72), displays messages pertaining to events that trigger an SPF calculation. The second command debug isis spf-events (Example 10-73). This command displays a detailed account of the SPF calculations caused by a triggering event. The third command, debug isis spf-statistics (Exampl 74), displays information about the SPF calculation itself. Of particular interest is the time taken to complete the calculation, which can reveal performance problems on the router.

## Example 10-72. *debug isis spf-triggers* displays messages about events trigger an SPF calculation.

```
Frankfurt#debug isis spf-triggers
IS-IS SPF triggering events debugging is on
Frankfurt#
ISIS-Spf: L1, LSP fields changed 0000.0C0A.2AA9.00-00
ISIS-Spf: L1, LSP fields changed 0000.0C0A.2AA9.00-00
Frankfurt#
```

Example 10-73. *debug isis spf-events* displays the details of an SPF calculation.

```
Frankfurt#debug isis spf-events
IS-IS SPF events debugging is on
Frankfurt#
ISIS-Spf: L1 LSP 4 (0000.0C0A.2AA9.00-00) flagged for recalculation from 37D73FA
ISIS-Spf: Calculating routes for L1 LSP 4 (0000.0C0A.2AA9.00-00)
ISIS-Spf: Add 172.16.19.0/255.255.255.0 to IP RIB, metric 20
ISIS-Spf: Next hop 0000.0C0A.2AA9/172.16.19.1 (Ethernet0) (rejected)
ISIS-Spf: Redundant IP route 172.16.24.0/255.255.255.0, metric 20, not added
ISIS-Spf: Redundant IP route 172.16.25.0/255.255.255.0, metric 20, not added
ISIS-Spf: Aging L1 LSP 4 (0000.0C0A.2AA9.00-00), version 105
ISIS-Spf: Aging IP 172.16.9.0/255.255.255.0, next hop 172.16.19.1
ISIS-Spf: Deleted NDB
ISIS-Spf:  Compute L1 SPT
ISIS-Spf: 3 nodes for level-1
ISIS-Spf: Move 0000.0C8D.34F1.00-00 to PATHS, metric 0
ISIS-Spf: Add 0000.0C0A.2AA9.03-00 to TENT, metric 10
ISIS-Spf: Move 0000.0C0A.2AA9.03-00 to PATHS, metric 10
ISIS-Spf: thru 0/2147483647/0, delay 0/0/0, mtu 0/2147483647/0, hops 0/0/0, ticks
0/0/0
ISIS-Spf: considering adj to 0000.0C0A.2AA9 (Ethernet0) metric 10, level 1, circuit
3, adj 1
ISIS-Spf:   (accepted)
ISIS-Spf: Add 0000.0C0A.2AA9.00-00 to TENT, metric 10
ISIS-Spf:   Next hop 0000.0C0A.2AA9 (Ethernet0)
ISIS-Spf: Move 0000.0C0A.2AA9.00-00 to PATHS, metric 10
ISIS-Spf: Add 172.16.19.0/255.255.255.0 to IP RIB, metric 20
ISIS-Spf: Next hop 0000.0C0A.2AA9/172.16.19.1 (Ethernet0) (rejected)
ISIS-Spf: Redundant IP route 172.16.24.0/255.255.255.0, metric 20, not added
ISIS-Spf: Redundant IP route 172.16.25.0/255.255.255.0, metric 20, not added
ISIS-Spf: Aging L1 LSP 2 (0000.0C8D.34F1.00-00), version 101
ISIS-Spf: Aging L1 LSP 3 (0000.0C0A.2AA9.03-00), version 99
ISIS-Spf: Aging L1 LSP 4 (0000.0C0A.2AA9.00-00), version 106
ISIS-Spf: Remove stale 0/0 from IP RIB
ISIS-Spf: L1 LSP 4 (0000.0C0A.2AA9.00-00) flagged for recalculation from 37D73FA
ISIS-Spf: Calculating routes for L1 LSP 4 (0000.0C0A.2AA9.00-00)
ISIS-Spf: Add 172.16.19.0/255.255.255.0 to IP RIB, metric 20
ISIS-Spf: Next hop 0000.0C0A.2AA9/172.16.19.1 (Ethernet0) (rejected)
ISIS-Spf: Add 172.16.9.0/255.255.255.0 to IP RIB, metric 20
ISIS-Spf: Next hop 0000.0C0A.2AA9/172.16.19.1 (Ethernet0) (accepted)
ISIS-Spf: Redundant IP route 172.16.24.0/255.255.255.0, metric 20, not added
ISIS-Spf: Redundant IP route 172.16.25.0/255.255.255.0, metric 20, not added
ISIS-Spf: Aging L1 LSP 4 (0000.0C0A.2AA9.00-00), version 107
```

Example 10-74. *debug isis spf-statistics* displays statistical information the SPF calculation.

```
Frankfurt#debug isis spf-statistics
IS-IS SPF Timing and Statistics Data debugging is on
Frankfurt#
ISIS-Spf:  Compute L1 SPT
ISIS-Spf: Complete L1 SPT,
ISIS-Spf:  Compute time 0.032/0.032, 2/1 nodes, 0/2 links, 0 suspends
ISIS-Spf:  Compute L1 SPT
ISIS-Spf: Complete L1 SPT,
ISIS-Spf:  Compute time 0.036/0.036, 2/1 nodes, 0/2 links, 0 suspends
```

Within each area, every router must have an identical link-state database. Additionally, every L1/L
L2 router in the IS-IS domain must have an identical L2 database. If you suspect that a router's li
database is not correctly synchronized, examine the LSP IDs and the checksums. The same LSP ID
exist in every database, and the checksum of each LSP should be identical in every database.

Two debug commands can help you observe the synchronization process. The first is `debug isis u`
`packets` (Example 10-75), which displays information about SNPs and LSPs the router sends and
The second command, `debug isis snp-packets` (Example 10-76), displays information specific to
CSNPs and PSNPs the router sends and receives.

### Example 10-75. *debug isis update-packets* displays information about S and LSPs.

```
Frankfurt#debug isis update-packets
IS-IS Update related packet debugging is on
Frankfurt#
ISIS-Upd: Rec L1 LSP 0000.0C0A.2AA9.00-00, seq 10, ht 1199,
ISIS-Upd: from SNPA 0005.5e6b.50a0 (Ethernet0)
ISIS-Upd: LSP newer than database copy
ISIS-Upd: Leaf routes changed
ISIS-Upd: Rec L1 LSP 0000.0C0A.2AA9.00-00, seq 11, ht 1199,
ISIS-Upd: from SNPA 0005.5e6b.50a0 (Ethernet0)
ISIS-Upd: LSP newer than database copy
ISIS-Upd: Important fields changed
ISIS-Upd: Full SPF required
ISIS-Upd: Rec L1 LSP 0000.0C0A.2AA9.00-00, seq 12, ht 1199,
ISIS-Upd: from SNPA 0005.5e6b.50a0 (Ethernet0)
Frankfurt#
```

### Example 10-76. *debug isis snp-packets* displays detailed information ab CSNPs and PSNPs.

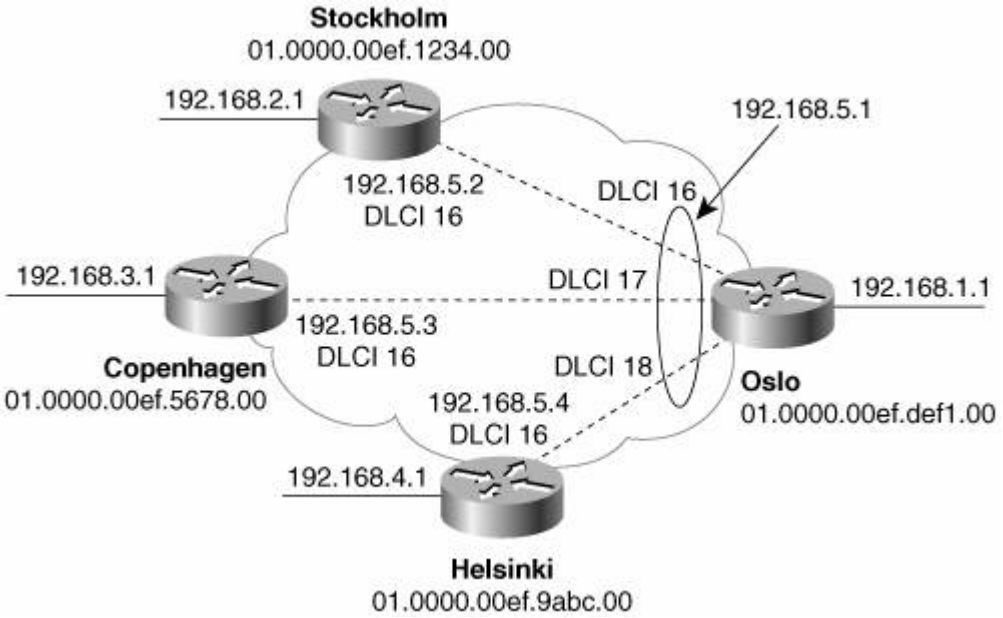
```
Frankfurt#debug isis snp-packets
IS-IS CSNP/PSNP packets debugging is on
Frankfurt#
ISIS-Snp: Rec L1 CSNP from 0000.0C0A.2AA9 (Ethernet0)
ISIS-Snp: CSNP range 0000.0000.0000.00-00 to FFFF.FFFF.FFFF.FF-FF
ISIS-Snp: Same entry 0000.0C04.DCC0.00-00, seq 9
ISIS-Snp: Same entry 0000.0C0A.2AA9.00-00, seq 1A
ISIS-Snp: Same entry 0000.0C0A.2AA9.01-00, seq 6
ISIS-Snp: Rec L1 CSNP from 0000.0C0A.2AA9 (Ethernet0)
ISIS-Snp: CSNP range 0000.0000.0000.00-00 to FFFF.FFFF.FFFF.FF-FF
ISIS-Snp: Same entry 0000.0C04.DCC0.00-00, seq 9
ISIS-Snp: Entry 0000.0C0A.2AA9.00-00, seq 1B is newer than ours (seq 1A), sending P
ISIS-Snp: Same entry 0000.0C0A.2AA9.01-00, seq 6
ISIS-Snp: Build L1 PSNP entry for 0000.0C0A.2AA9.00-00, seq 1A
ISIS-Snp: Sending L1 PSNP on Ethernet0
ISIS-Snp: Rec L1 CSNP from 0000.0C0A.2AA9 (Ethernet0)
ISIS-Snp: CSNP range 0000.0000.0000.00-00 to FFFF.FFFF.FFFF.FF-FF
ISIS-Snp: Same entry 0000.0C04.DCC0.00-00, seq 9
ISIS-Snp: Same entry 0000.0C0A.2AA9.00-00, seq 1B
ISIS-Snp: Same entry 0000.0C0A.2AA9.01-00, seq 6
ISIS-Snp: Rec L1 CSNP from 0000.0C0A.2AA9 (Ethernet0)
ISIS-Snp: CSNP range 0000.0000.0000.00-00 to FFFF.FFFF. FFFF.FF-FF
ISIS-Snp: Same entry 0000.0C04.DCC0.00-00, seq 9
```

```
ISIS-Snp: Same entry 0000.0C0A.2AA9.00-00, seq 1B
ISIS-Snp: Same entry 0000.0C0A.2AA9.01-00, seq 6
Frankfurt#
```

## Case Study: Integrated IS-IS on NBMA Networks

Figure 10-56 shows four routers running IS-IS connected by a partially meshed Frame Relay netw IP addresses, DLCIs, and NETs are shown. The IS-IS configurations of all routers have been verifie correct, and no authentication is configured.

Figure 10-56. IS-IS is not establishing adjacencies across the Frame F network.




The problem with this network is that no routes are being discovered (Example 10-77). The IP adc of neighbors' Frame Relay interfaces can be pinged, but the addresses of the routers' other interfa cannot be pinged (Example 10-78). The pings show that the Frame Relay PVCs are operational an is working, but that the routers are not routing.

Example 10-77. The route table of Oslo in Figure 10-56 does not contair IS-IS routes.

```
Oslo#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
C    192.168.1.0 is directly connected, TokenRing0
C    192.168.5.0 is directly connected, Serial0
Oslo#
```

## Example 10-78. Pings are successful to other interfaces connected to th Frame Relay network, but addresses reachable through the routers can pinged.

```
Oslo#ping 192.168.5.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.5.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/65/68 ms
Oslo#ping 192.168.2.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Oslo#ping 192.168.5.3
Type escape sequence to 5, 100-byte ICMP Echos to 192.168.5.3, timeout is 2 second
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/66/68 ms
Oslo#ping 192.168.3.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Oslo#ping 192.168.5.4
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.5.4, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/65/68 ms
Oslo#ping 192.168.4.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.4.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Oslo#
```

The next step is to check the IS-IS neighbor table. Oslo's neighbor table shows that Hellos have b received (Example 10-79) and that the router knows the System IDs of its neighbors. It also show the IP addresses and the area addresses of the neighbors are correct. However, the state of all of neighbors is *Init*, indicating that a full adjacency has not been established. A look at the link-state database verifies the absence of adjacencies; the only LSPs in Oslo's database are the router's ow (Example 10-80).

## Example 10-79. Oslo's IS-IS neighbor table shows that Hellos are being received but that the adjacency is not complete.

```
Oslo#show clns is-neighbors detail
System Id          Interface   State   Type Priority   Circuit Id         Format
0000.00EF.5678     Se0         Init    L1L2 0 /0       0000.0000.0000.00  Phase V
  Area Address(es): 01
  IP Address(es): 192.168.5.3
  Uptime: 1:11:20
0000.00EF.1234     Se0         Init    L1L2 0 /0       0500.0000.0000.00  Phase V
  Area Address(es): 01
  IP Address(es): 192.168.5.2
  Uptime: 1:11:15
0000.00EF.9ABC     Se0         Init    L1L2 0 /0       0700.0000.0000.00  Phase V
  Area Address(es): 01
  IP Address(es): 192.168.5.4
```

```
  Uptime: 1:11:20
Oslo#
```

## Example 10-80. Oslo's link-state database does not contain any LSPs fro any of its neighbors.

```
Oslo#show isis database
IS-IS Level-1 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.00EF.DEF1.00-00* 0x0000001F   0x8460        947           0/0/0
0000.00EF.DEF1.02-00* 0x00000010   0x695E        896           0/0/0
0000.00EF.DEF1.04-00* 0x00000002   0x2F2E        887           0/0/0
0000.00EF.DEF1.05-00* 0x00000008   0x1C3A        847           0/0/0
IS-IS Level-2 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.00EF.DEF1.00-00* 0x00000013   0x81BE        829           0/0/0
Oslo#
```

The fact that Hellos are being received, but adjacencies are not being established, points to a prob the Hellos themselves. If the parameters in the Hellos are not correct, the PDU is dropped. So deb adj-packets is enabled to observe the Hellos. Of particular interest in the debug output (Example are the "encapsulation failed" messages. These messages show that the router apparently cannot the received Hellos and is dropping them.

## Example 10-81. The results of enabling *debug isis adj-packets* show tha Hellos are being dropped because of encapsulation failures.

```
Oslo#debug isis adj-packets
IS-IS Adjacency related packets debugging is on
Oslo#
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Encapsulation failed for level 2 IIH on Serial0
ISIS-Adj: Rec serial IIH from DLCI 17 on Serial0, cir type 3, cir id 00
ISIS-Adj: rcvd state 2, old state 1, new state 1
ISIS-Adj: Action = 1, new_type = 3
ISIS-Adj: Sending L2 IIH on TokenRing0
ISIS-Adj: Encapsulation failed for level 1 IIH on Serial0
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Rec serial IIH from DLCI 18 on Serial0, cir type 3, cir id 07
ISIS-Adj: rcvd state 2, old state 1, new state 1
ISIS-Adj: Action = 1, new_type = 3
ISIS-Adj: Encapsulation failed for level 2 IIH on Serial0
ISIS-Adj: Sending L2 IIH on TokenRing0
ISIS-Adj: Encapsulation failed for level 1 IIH on Serial0
ISIS-Adj: Rec serial IIH from DLCI 16 on Serial0, cir type 3, cir id 05
ISIS-Adj: rcvd state 2, old state 1, new state 1
ISIS-Adj: Action = 1, new_type = 3
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Encapsulation failed for level 2 IIH on Serial0no debu
ISIS-Adj: Sending L2 IIH on TokenRing0
ISIS-Adj: Encapsulation failed for level 1 IIH on Serial0g a
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Rec serial IIH from DLCI 17 on Serial0, cir type 3, cir id 00
ISIS-Adj: rcvd state 2, old state 1, new state 1
ISIS-Adj: Action = 1, new_type = 3
ISIS-Adj: Encapsulation failed for level 2 IIH on Serial0ll
```

Any time an "encapsulation failed" message is received, suspicion should fall on the data link and connected interfaces. Examining the interfaces with the show interface serial command reveals significant error rates, so it is unlikely that the Hellos are being corrupted on the Frame Relay PVC next step is to examine the interface configurations. The interface configurations for the four route Figure 10-56 are displayed in Example 10-82 through Example 10-85.

## Example 10-82. Oslo's interface configuration.

```
interface Serial0
 ip address 192.168.5.1 255.255.255.0
 ip router isis
 encapsulation frame-relay
 frame-relay interface-dlci 16
 frame-relay interface-dlci 17
 frame-relay interface-dlci 18
```

## Example 10-83. Stockholm's interface configuration.

```
interface Serial0
 no ip address
 encapsulation frame-relay
!
interface Serial0.16 point-to-point
 ip address 192.168.5.2 255.255.255.0
 ip router isis
 frame-relay interface-dlci 16
```

## Example 10-84. Copenhagen's interface configuration.

```
interface Serial0
 no ip address
 encapsulation frame-relay
!
interface Serial0.16 point-to-point
 ip address 192.168.5.3 255.255.255.0
 ip router isis
 frame-relay interface-dlci 16
```

## Example 10-85. Helsinki's interface configuration.

```
interface Serial0
 no ip address
 encapsulation frame-relay
!
interface Serial0.16 point-to-point
 ip address 192.168.5.4 255.255.255.0
 ip router isis
 frame-relay interface-dlci 16
```

A comparison of these configurations reveals the problem, although it might not be readily appare Stockholm, Copenhagen, and Helsinki are all configured with point-to-point subinterfaces. Oslo is

subinterfaces. By default, a Cisco serial interface with Frame Relay encapsulation is a multi-point i
Therefore, Stockholm, Copenhagen, and Helsinki are sending point-to-point IS-IS Hellos, and Oslo
sending L1 and L2 IS-IS LAN Hellos.

IS-IS does not have a configuration option similar to OSPF's `ip ospf network` command; therefor
must be reconfigured with point-to-point subinterfaces, and the IP addresses must be changed so
each PVC is a different subnet (Example 10-86).

### Example 10-86. After Oslo has been configured with point-to-point subinterfaces and the PVCs have been readdressed as individual subnet IS routing is functioning.

```
Oslo#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
C    192.168.1.0 is directly connected, TokenRing0
i L1 192.168.2.0 [115/20] via 192.168.5.5, Serial0.16
i L1 192.168.3.0 [115/20] via 192.168.5.9, Serial0.17
i L1 192.168.4.0 [115/20] via 192.168.5.13, Serial0.18
     192.168.5.0 is variably subnetted, 4 subnets, 2 masks
C       192.168.5.12 255.255.255.252 is directly connected, Serial0.18
C       192.168.5.8 255.255.255.252 is directly connected, Serial0.17
C       192.168.5.4 255.255.255.252 is directly connected, Serial0.16
C       192.168.5.0 255.255.255.0 is directly connected, Serial0
Oslo#
```

PREV NEXT

# Looking Ahead

Now that all of the IP IGPs have been examined, the next step is to study the tools available to help you control your network. Part III, "Route Control and Interoperability," covers redistribution, default routes, On-Demand Routing, route filters, and route maps.

# Summary Table: Chapter 10 Command Review

| Command | Description |
|---|---|
| address-family ipv6 | Specifies that the following commands apply to the IPv6 address family. |
| area-password *password* | Configures IS-IS area (level 1) authentication. |
| authentication key-chain *name-of-chain* [level-1 \| level-2] | Specifies the preconfigured key-chain to use for authenticating either level-1 or level-2 router. Level-1 indicates area-wide authentication, level-2 indicates domain-wide authentication. |
| authentication mode {md5 \| text} [level-1 \| level-2] | Specifies the authentication mode to use between routers in an area or domain. |
| authentication send-only [level-1 \| level-2] | Used when configuring authentication without invalidating existing databases, this command sends authentication information, but doesn't expect to receive matching authentication information. |
| debug isis adj-packets | Displays IS-IS Hello PDU activity. |
| debug isis spf-events | Displays details of events triggering an IS-IS SPF calculation. |
| debug isis snp-packets | Displays information about SNPs sent and received by the router. |
| debug isis spf-statistics | Displays statistical information about IS-IS SPF calculations. |
| debug isis spf-triggers | Displays events that trigger IS-IS SPF calculations. |
| debug isis update-packets | Displays information about LSPs, CSNPs, and PSNPs sent and received by the router. |
| default-information originate [route-map *map-name*] | Generates a default IP route into an IS-IS domain. |
| domain-password *password* | Configures IS-IS domain (level 2) authentication. |
| ignore-lsp-errors | Configures an IS-IS router to ignore errored LSPs rather than triggering a purge of the LSPs. |
| ip router isis [*tag*] | Enables IPv4 IS-IS routing on an interface. |
| ipv6 router isis [*tag*] | Enables IPv6 IS-IS routing on an interface. |
| isis authentication key-chain *name-of-chain* [level-1 \| level-2] | Specifies the preconfigured key-chain to use for authenticating either level-1 or level-2 on an interface. |
| isis authentication mode {md5 \| text} [level-1 \| level-2] | Specifies the authentication mode to use on an interface. |
| isis authentication send-only [level-1 \| level-2] | Used when configuring authentication without breaking existing adjacencies, this command sends authentication information, but doesn't expect to receive matching authentication information. |

| Command | Description |
|---|---|
| isis csnp-interval *seconds* {level-1 \| level-2} | Specifies the interval in which an IS-IS Designated Router sends CSNPs. |
| isis hello-interval *seconds* {level-1 \| level-2} | Specifies the interval between transmissions of IS-IS Hello PDUs. |
| isis hello-multiplier *multiplier* {level-1 \| level-2} | Specifies the number of IS-IS Hello PDUs a neighbor must miss before declaring its adjacency to the originating router down. |
| isis lsp-interval *milliseconds* | Changes the default delay between transmissions of LSPs on an interface. |
| isis mesh-group [*number* \| blocked] | Assigns an interface to a numbered mesh group, or entirely blocks LSP flooding on the interface. |
| isis metric *default-metric* {level-1 \| level-2} | Specifies an interface's IS-IS default metric. |
| isis password *password* {level-1 \| level-2} | Configures authentication between two IS-IS neighbors. |
| isis priority *value* {level-1 \| level-2} | Specifies the priority of an interface to be used for DR election. |
| isis retransmit-interval *seconds* | Specifies the time a router will wait for an acknowledgment after sending an LSP on a point-to-point link before retransmitting the LSP. |
| is-type {level-1 \| level-1-2 \| level-2-only} | Configures the router as an L1, L1/L2, or L2 IS-IS router. |
| log-adjacency-changes | Sends all adjacency change events to the log. |
| lsp-refresh-interval *seconds* | Changes the default period between refreshes of locally originated LSPs. |
| lsp-gen-interval [level-1 \| level-2] *lsp-max-wait* [lsp-initial-wait lsp-second-wait] | Changes the default values for the exponential backoff algorithm regulating the generation of LSPs. |
| max-lsp-lifetime *seconds* | Changes the default value of the Remaining Lifetime of locally originated LSPs. |
| Metric-style [narrow \| wide \| transition] (level-1 \| level-2 \| level-1-2} | Specifies the type of metric style to use. |
| Multi-topology {transition} | Enables multi-topology for IPv6. |
| net *network-entity-title* | Configures an IS-IS router's NET. |
| prc-interval *prc-max-wait* [prc-initial-wait prc-second-wait] | Changes the default values for the exponential backoff algorithm used with partial route calculations. |
| Redistribute isis {ip} level-2 into level-2 distribute-list [access-list number \| prefix-list name] | Enables leaking of routes from IS-IS level-2 into level-1 areas. |
| router isis [*tag*] | Enables an IS-IS routing process. |
| set-overload-bit [on-startup {*seconds*} \| wait-for-bgp] [suppress {[interlevel] \| [external]}] | Manually sets the Overload bit in a router's LSP to one. |

| Command | Description |
| --- | --- |
| show clns is-neighbor [*type number*] [detail] | Displays the IS-IS neighbor table. |
| show clns protocol | Displays information about how clns and integrated IS-IS is configured on the router. |
| show isis database [level-1] [level-2] [l1] [l2] [detail] [*lspid*] | Displays an IS-IS link-state database. |
| show isis hostname | Displays the system IDs and hostnames of routers in the network. The information is obtained from type 137 TLVs. |
| show isis ipv6 topology | Displays the IPv6 topology. |
| show isis spf-log | Displays how often and why the router has run a full SPF calculation. |
| show isis topology | Displays the IPv4/CLNS topology. |
| spf-interval [level-1 \| level-2] *spf-max-wait* [spf-initial-wait spf-second-wait] | Changes the default parameters of the exponential backoff algorithm used by the SPF calculation. |
| summary-address *address-mask* {level-1 \| level-1-2 \| level-2} | Configures IP address summarization. |
| summary-prefix *ipv6_prefix/prefix_length* {level-1 \| level-1-2 \| level-2} | Configures IPv6 address summarization. |
| which-route {*nsap-address* \| *clns-name*} | Displays the routing table in which the specified CLNS destination is found and displays details of the associated IP addresses and area addresses. |

PREV NEXT

# Review Questions

1. What is an intermediate system?
2. What is a Network Protocol Data Unit?
3. What is the difference between an L1, an L2, and an L1/L2 router?
4. What is the default level of a Cisco router?
5. Explain the basic difference between an IS-IS area and an OSPF area.
6. What adjacency types, if any, are formed between two L1/L2 routers with the same Area IDs? What if the AIDs of the L1/L2 routers are different?
7. What adjacency types, if any, are formed between two L2-only routers with the same AIDs? What if the AIDs of the L2-only routers are different?
8. What is a network entity title (NET)?
9. To what value must the NSAP Selector be set in a NET?
10. What is the purpose of a System ID?
11. How does a router determine what area it is in?
12. Does IS-IS elect a Backup Designated Router on a broadcast subnetwork?
13. What is the purpose of the Pseudonode ID?
14. What is the default maximum age (MaxAge) of an IS-IS LSP? What is the maximum configurable MaxAge?
15. What is the basic difference between the way OSPF ages its LSAs and the way IS-IS ages its LSPs?
16. How often does an IS-IS router refresh its LSPs?
17. What is a Complete Sequence Number Packet (CSNP)? How is it used?
18. What is a Partial Sequence Number Packet (PSNP)? How is it used?
19. What is the purpose of the Overload (OL) bit?
20. What is the purpose of the Attached (ATT) bit?
21. What is the purpose of the Up/Down bit?
22. What metrics are specified by the ISO for IS-IS? How many of these metrics does the Cisco IOS support?
23. What are the two default metric styles and what are their maximum values?
24. What is the maximum metric value of an IS-IS route?

25 What is the difference between a level 1 IS-IS metric and a level 2 IS-IS metric?

26 What is the difference between an internal IS-IS metric and an external IS-IS metric?

27 How many adjacencies are created on a single link between two routers running both IPv4 and IPv6 in single topology mode? How many in multi-topology mode?

28 How many L1 areas can be active on a single router? How many L2 areas?

29 What are the two mesh group modes other than Inactive, and what are the advantages and disadvantages of each?

PREV NEXT

# Configuration Exercises

1 Table 10-8 shows the interfaces, interface addresses, and subnet masks of 11 routers. The table also designates that routers belong in the same area. Write Integrated IS-IS configurations for the routers, using the following guidelines:

a. Make up your own System IDs for the routers.

b. Use the shortest NETs possible.

c. Configure routers as L1, L2, or L1/L2 as appropriate.

Table 10-8. Router information for Configuration Exercises 1 through 5.

| Router | Area | Interface | Address/Mask |
|---|---|---|---|
| A | 0 | E0 | 192.168.1.17/28 |
| | | E1 | 192.168.1.50/28 |
| B | 0 | E0 | 192.168.1.33/28 |
| | | E1 | 192.168.1.51/28 |
| C | 0 | E0 | 192.168.1.49/28 |
| | | S0 | 192.168.1.133/30 |
| D | 2 | S0 | 192.168.1.134/30 |
| | | S1 | 192.168.1.137/30 |
| E | 2 | S0 | 192.168.1.142/30 |
| | | S1 | 192.168.1.145/30 |
| | | S2 | 192.168.1.138/30 |
| F | 2 | S0 | 192.168.1.141/30 |
| | | S1 | 192.168.1.158/30 |
| G | 1 | E0 | 192.168.1.111/27 |
| | | S0 | 192.168.1.157/30 |
| H | 1 | E0 | 192.168.1.73/27 |
| | | E1 | 192.168.1.97/27 |
| I | 3 | E0 | 192.168.1.225/29 |

| Router | Area | Interface | Address/Mask |
|---|---|---|---|
| | | E1 | 192.168.1.221/29 |
| | | S0 | 192.168.1.249/30 |
| | | S1 | 192.168.1.146/30 |
| J | 3 | E0 | 192.168.1.201/29 |
| | | E1 | 192.168.1.217/29 |
| K | 3 | EO | 192.168.1.209/29 |
| | | S0 | 192.168.1.250/30 |

Tip: Draw a picture of the routers and subnets first.

2 Configure IPv6 on each router, in multi-topology mode. Use the address 2001:db8:0:x::/64, where x is the hexadecimal value of the subnet (4th octet only) of the corresponding IPv4 address on each link.

3 Configure authentication between all routers in area 2 of Table 10-8. Use the password "Eiffel" between routers D and E; use the password "Tower" between routers E and F. Use type md5 authentication.

4 Configure level 1 authentication in area 1 of Table 10-8. Use the password "Scotland." Use cleartext passwords with no key chains.

5 Configure level 2 authentication on the routers in Table 10-8. Use the password "Vienna." Use cleartext passwords with key chains.

6 Configure the L1/L2 routers in areas 0, 1, and 3 in Table 10-8 to summarize their L1 addresses, both IPv4 and IPv6.

PREV NEXT

# Troubleshooting Exercises

1 Example 10-87 and Example 10-88 show the IS-IS neighbor tables of two routers, router A router B, connected to via a serial interface. What is wrong?

### Example 10-87. The IS-IS neighbor table of router A, Troubleshooti Exercise 1.

```
Router_A#show clns is-neighbors detail
System Id      Interface  State  Type Priority  Circuit Id         Format
0000.00EF.DCBA To0        Up     L1L2 64/64     0000.00EF.DCBA.04  Phase V
  Area Address(es): 01
  IP Address(es): 192.168.11.2
  Uptime: 0:09:25
0000.00EF.5678 Se0.17     Up     IS   0 /0      00                 Phase V
  Area Address(es): 01
  IP Address(es): 192.168.5.9
  Uptime: 1:28:22
0000.00EF.9ABC Se0.18     Up     L1L2 0 /0      07                 Phase V
  Area Address(es): 01
  IP Address(es): 192.168.5.13
  Uptime: 1:29:45
0000.00EF.1234 Se0.16     Up     L1L2 0 /0       06                 Phase V
  Area Address(es): 01
  IP Address(es): 192.168.5.5
  Uptime: 1:29:45
Router_A#
```

### Example 10-88. The IS-IS neighbor table of router B, Troubleshooti Exercise 1.

```
Router_B#show clns is-neighbors detail
System Id      Interface  State  Type Priority  Circuit Id          Format
0000.00EF.DEF1 Se0.17     Up     IS 0 /0        00                  Phase V
  Area Address(es): 01
  IP Address(es): 10.1.1.1
  Uptime: 0:11:06
Router_B#
```

2 Example 10-89 shows debug messages from a router that is not establishing an adjacency \ a neighbor on its interface TO0. What is wrong?

### Example 10-89. The debug output for Troubleshooting Exercise 2.

```
Router_B#debug isis adj-packets
IS-IS Adjacency related packets debugging is on
Router_B#
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Sending L1 IIH on TokenRing1
```

```
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Sending L1 IIH on TokenRing1
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Sending L1 IIH on TokenRing1
ISIS-Adj: Rec L2 IIH from 0000.3090.c7df (TokenRing0), cir type 2, cir id 000
0EF.DCBA.04
ISIS-Adj: is-type mismatch
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Sending L1 IIH on TokenRing1
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Sending L1 IIH on TokenRing1
ISIS-Adj: Sending L1 IIH on TokenRing1
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Rec L2 IIH from 0000.3090.c7df (TokenRing0), cir type 2, cir id 000
0EF.DCBA.04
ISIS-Adj: is-type mismatch
ISIS-Adj: Sending L1 IIH on TokenRing1
ISIS-Adj: Sending L1 IIH on TokenRing0
ISIS-Adj: Sending L1 IIH on TokenRing1
ISIS-Adj: Sending L1 IIH on TokenRing0
 IIH on TokenRing0L1 IIH on TokenRing1
```

PREV NEXT

# Part III: Route Control and Interoperability

Chapter 11 Route Redistribution

Chapter 12 Default Routes and On-Demand Routing

Chapter 13 Route Filtering

Chapter 14 Route Maps

# Chapter 11. Route Redistribution

This chapter covers the following subjects:

- Principles of Redistribution
- Configuring Redistribution

A router performs redistribution when it uses a routing protocol to advertise routes that were learned by some other means. Those "other means" might be another routing protocol, static routes, or a direct connection to the destination network. For example, a router might be running both an Open Shortest Path First (OSPF) process and a Routing Information Protocol (RIP) process. If the OSPF process is configured to advertise routes learned by the RIP process, it is said to be "redistributing RIP."

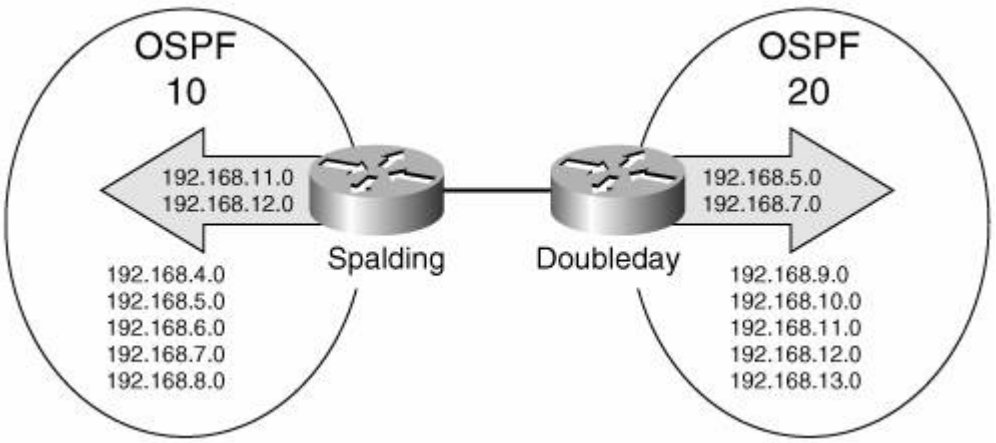
Running a single routing protocol throughout the entire IP network is usually more desirable than running multiple protocols, both from a configuration management perspective and from a fault management perspective. However, the realities of modern networking frequently force the acceptance of multiprotocol IP routing domains. As departments, divisions, and entire companies merge, their formerly autonomous networks must be consolidated.

In most cases, the networks that are to be consolidated were implemented differently and have evolved differently, to meet different needs or merely as the result of different design philosophies. This diversity can make the migration to a single routing protocol a complex undertaking. In some cases, corporate politics can force the use of multiple routing protocols. And in a few cases, multiple routing protocols might be the result of network administrators who do not work and play well together.

Multi-vendor environments are another factor that can necessitate redistribution. For example, a network running the Cisco EIGRP might be merged with a network using another manufacturer's routers, which support only RIP or OSPF. Without redistribution, either the Cisco routers would have to be reconfigured to an open protocol, or the non-Cisco routers would have to be replaced with Cisco routers.

Redistribution is necessary when multiple routing protocols are "thrown together," but redistribution can also be part of a well-thought-out network design. Figure 11-1 shows an example. Here two OSPF process domains are connected, but the OSPF processes do not communicate directly. Instead, static routes are configured on each router to selected networks in the other OSPF domain.

Figure 11-1. In this network, static routes are configured on each router and redistributed into OSPF. As a result, the advertisement of prefixes between the two OSPF domains is carefully controlled.




For instance, router Spalding has static routes to networks 192.168.11.0 and 192.168.12.0. Spalding then redistributes these static routes into OSPF, and OSPF advertises the routes to the other routers in OSPF 10. The result is that the other networks in OSPF 20 are hidden from OSPF 10. Redistribution has allowed the dynamic characteristics of OSPF to be mixed with the precise control of static routes.

Static routes redistributed into a dynamic routing protocol are also very useful, if not essential, in dial-up environments. The periodic management traffic of a dynamic protocol can cause the dial-up line to be "always up." By blocking routing updates or Hellos across the link and configuring static routes on each side, the administrator can ensure that the link will come up only when user traffic must traverse it. And by redistributing the static routes into the dynamic routing protocol, all routers on both sides of the dial-up link have full knowledge of all subnets on the opposite side of the link.

Note that with few exceptions,[1] the existence of more than one routing protocol on the same router does not mean that redistribution will occur automatically. Redistribution must be explicitly configured. The configuration of multiple routing protocols on a single router without redistribution is called *ships in the night* (SIN) routing. The router will pass routes to its peers in each process domain, but the process domains will have no knowledge of each otherlike ships passing in the night. Although SIN routing usually refers to multiple routing protocols routing multiple routed protocols on the same router (such as OSPF routing IP and NLSP routing IPX), it can also refer to two IP protocols routing for separate IP domains on a single router.

[1] Within IP, IGRP and EIGRP processes with the same autonomous system number will redistribute automatically.

PREV NEXT

# Principles of Redistribution

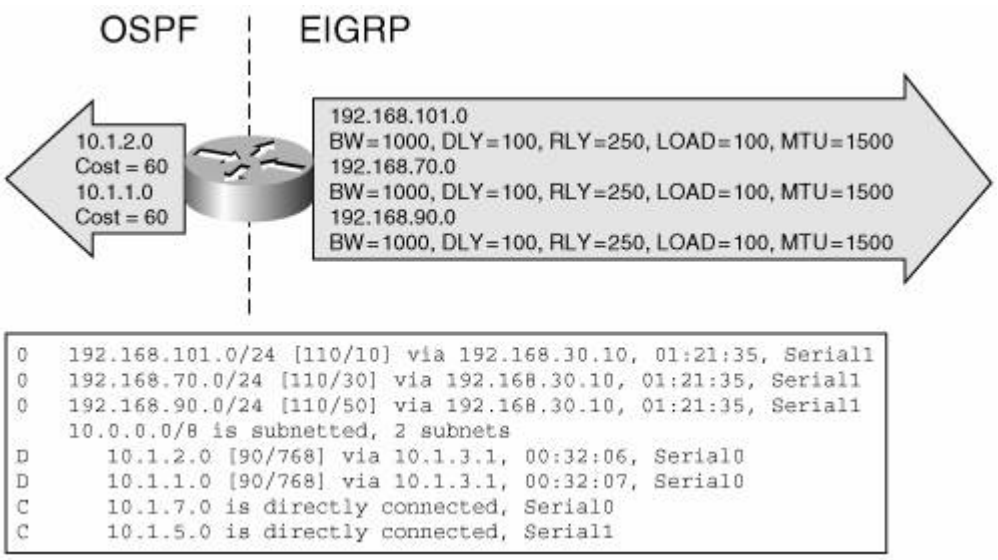
The capabilities of the IP routing protocols vary widely. The protocol characteristics that have the most bearing on redistribution are the differences in metrics and administrative distances, and each protocol's classful or classless capabilities. Failure to give careful consideration to these differences when redistributing can lead to, at best, a failure to exchange some or all routes or, at worst, routing loops and black holes.

## Metrics

The routers of Figure 11-1 are redistributing static routes into OSPF, which then advertises the routes to other OSPF-speaking routers. Static routes have no metric associated with them, but every OSPF route must have a cost. Another example of conflicting metrics is the redistribution of RIP routes into EIGRP. RIP's metric is hop count, whereas EIGRP uses bandwidth and delay. In both cases, the protocol into which routes are redistributed must be able to associate its own metrics with those routes.

So the router performing the redistribution must assign a metric to the redistributed routes. Figure 11-2 shows an example. Here EIGRP is being redistributed into OSPF, and OSPF is being redistributed into EIGRP. OSPF does not understand the composite metric of EIGRP, and EIGRP does not understand OSPF cost. As a result, part of the redistribution process is that the router must assign a cost to each EIGRP route before passing it to OSPF. Likewise, the router must assign a bandwidth, delay, reliability, load, and MTU to each OSPF route before passing it to EIGRP. If incorrect metrics are assigned, redistribution will fail.

Figure 11-2. When routes are redistributed, a metric that is understandable to the receiving protocol must be assigned to the routes.




Case studies later in this chapter show how to configure routers to assign metrics for purposes of redistribution.

## Administrative Distances

The diversity of metrics presents another problem: If a router is running more than one routing protocol and learns a route to the same destination from each of the protocols, which route should be selected? Each protocol uses its own metric scheme to define the best route. Comparing routes with different metrics, such as cost and hop count, is like comparing apples and oranges.

The answer to the problem is administrative distances. Just as metrics are assigned to routes so that the most preferred route may be determined, administrative distances are assigned to route sources so that the most preferred source may be determined. Think of an administrative distance as a measure of believability. The lower the administrative distance, the more believable the protocol.

For example, suppose a router running RIP and EIGRP learns of a route to network 192.168.5.0 from a RIP-speaking neighbor and another route to the same destination from an EIGRP-speaking neighbor. Because of EIGRP's composite metric, that protocol is more likely to have determined the optimum route. Therefore, EIGRP should be believed over RIP.

Table 11-1 shows the default Cisco administrative distances. EIGRP has an administrative distance of 90, whereas RIP has an administrative distance of 120. Therefore, EIGRP is deemed more trustworthy than RIP.

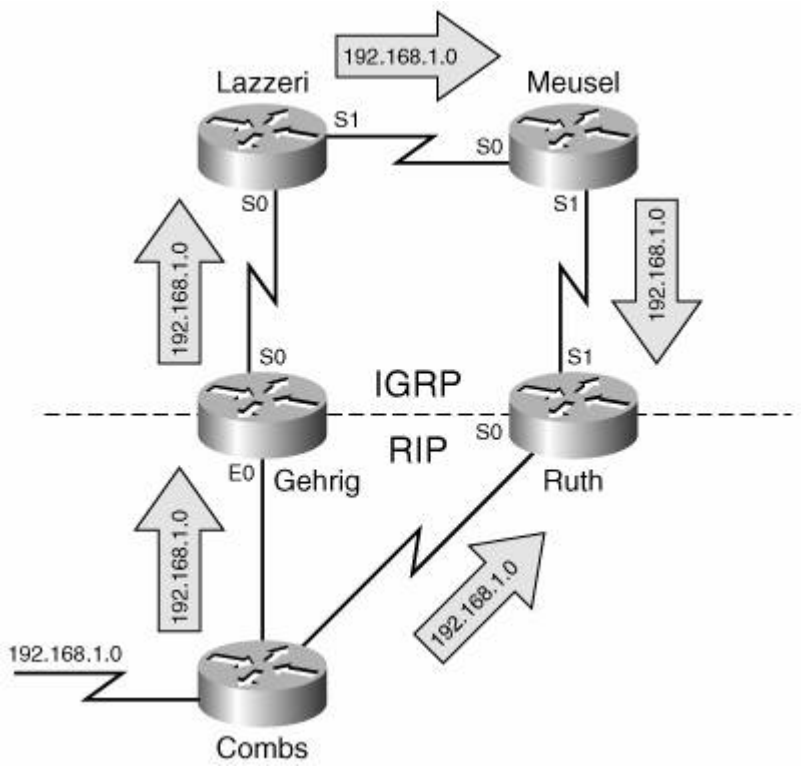
Table 11-1. Cisco default administrative distances.

| Route Source | Administrative Distance |
|---|---|
| Connected interface[*] | 0 |
| Static route | 1 |
| EIGRP summary route | 5 |
| External BGP | 20 |
| EIGRP | 90 |
| IGRP | 100 |
| OSPF | 110 |
| IS-IS | 115 |
| RIP | 120 |
| EGP | 140 |
| External EIGRP | 170 |
| Internal BGP | 200 |
| Unknown | 255 |

[*] Recall from Chapter 3, "Static Routing," that when a static route refers to an interface instead of a next-hop address, the destination is considered to be a directly connected network.

Although administrative distances help resolve the confusion of diverse metrics, they can cause problems for redistribution. For example, in Figure 11-3, both Gehrig and Ruth are redistributing RIP-learned routes into IGRP. Gehrig learns about network 192.168.1.0 via RIP and advertises the network into the IGRP domain. As a result, Ruth learns about network 192.168.1.0 not only from Combs via RIP, but also from Meusel via IGRP.

Figure 11-3. Network 192.168.1.0 is being advertised to Ruth via both RIP and IGRP.




Example 11-1 shows Ruth's route table. Notice that the route to 192.168.1.0 is an IGRP route. Ruth has chosen the IGRP route because, compared to RIP, IGRP has a lower administrative distance. Ruth will send all packets to 192.168.1.0 over the "scenic route" through Meusel, instead of sending them directly to Combs.

Example 11-1. Although the optimal route to network 192.168.1.0 from Ruth is through Combs, out S0, Ruth is instead routing to that network through Meusel, out S1.

```
Ruth#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
I    192.168.1.0/24 [100/16100] via 192.168.5.1, 00:00:00, Serial1
I    192.168.2.0/24 [100/12576] via 192.168.5.1, 00:00:00, Serial1
I    192.168.3.0/24 [100/12476] via 192.168.5.1, 00:00:01, Serial1
I    192.168.4.0/24 [100/10476] via 192.168.5.1, 00:00:01, Serial1
C    192.168.5.0/24 is directly connected, Serial1
```

```
C    192.168.6.0/24 is directly connected, Serial0
Ruth#
```

Split horizon prevents a routing loop in the network of Figure 11-3. Both Gehrig and Ruth initially advertise subnet 192.168.1.0 into the IGRP domain, and the four IGRP routers eventually converge on a single path to that subnet. However, the convergence is unpredictable. This condition can be seen by rebooting routers Lazzeri and Meusel. After the reboot, Ruth's route table shows that it is using Combs as the next-hop router to reach 192.168.1.0 (Example 11-2).

## Example 11-2. The convergence of the network in Figure 11-3 is unpredictable. After a reboot of the routers, Ruth is now routing to 192.168.1.0 through Combs.

```
Ruth#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
R    192.168.1.0/24 [120/1] via 192.168.6.2, 00:00:23, Serial0
I    192.168.2.0/24 [100/12576] via 192.168.5.1, 00:00:22, Serial1
I    192.168.3.0/24 [100/12476] via 192.168.5.1, 00:00:22, Serial1
I    192.168.4.0/24 [100/10476] via 192.168.5.1, 00:00:22, Serial1
C    192.168.5.0/24 is directly connected, Serial1
C    192.168.6.0/24 is directly connected, Serial0
Ruth#
```

Convergence after the reboot is not only unpredictable but also slow. Example 11-3 shows Gehrig's route table approximately three minutes after the reboot. It is using Lazzeri as a next-hop router to subnet 192.168.1.0, but pings to a working address on that link fail. Lazzeri's route table (Example 11-4) shows the problem: Lazzeri is using Gehrig as a next-hop router. A routing loop exists.

## Example 11-3. Soon after the reboot, Gehrig is routing packets to 192.168.1.0 via Lazzeri.

```
Gehrig#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
I    192.168.1.0/24 [100/16100] via 192.168.3.2, 00:02:38, Serial0
C    192.168.2.0/24 is directly connected, Ethernet0
C    192.168.3.0/24 is directly connected, Serial0
I    192.168.4.0/24 [100/10476] via 192.168.3.2, 00:00:29, Serial0
I    192.168.5.0/24 [100/12476] via 192.168.3.2, 00:00:29, Serial0
I    192.168.6.0/24 [100/14476] via 192.168.3.2, 00:00:39, Serial0
Gehrig#ping 192.168.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
.....
```

```
Success rate is 0 percent (0/5)
Gehrig#
```

## Example 11-4. Lazzeri is routing packets to 192.168.1.0 via Gehrig, creating a routing loop. Notice the age of the route.

```
Lazzeri#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
I    192.168.1.0/24 [100/12100] via 192.168.3.1, 00:04:21, Serial0
I    192.168.2.0/24 [100/8576] via 192.168.3.1, 00:00:33, Serial0
C    192.168.3.0/24 is directly connected, Serial0
C    192.168.4.0/24 is directly connected, Serial1
I    192.168.5.0/24 [100/10476] via 192.168.4.2, 00:00:53, Serial1
I    192.168.6.0/24 [100/12100] via 192.168.3.1, 00:02:32, Serial0
Lazzeri#
```

Here's the sequence of events leading to the loop:

1. While Lazzeri and Meusel are rebooting, both Gehrig and Ruth have route table entries showing network 192.168.1.0 as reachable via Combs.
2. As Lazzeri and Meusel become active, both Gehrig and Ruth send IGRP updates that include subnet 192.168.1.0. Simply by the "luck of the draw," Ruth sends its update slightly earlier than Gehrig does.
3. Meusel, receiving Ruth's update, makes Ruth the next-hop router and sends an update to Lazzeri.
4. Lazzeri, receiving Meusel's update, makes Meusel the next-hop router.
5. Lazzeri and Gehrig send updates to each other at about the same time. Lazzeri makes Gehrig the next-hop router to 192.168.1.0 because its route is metrically closer than Meusel's route. Gehrig makes Lazzeri the next-hop router to 192.168.1.0 because its IGRP advertisement has a lower administrative distance than Combs RIP advertisement. The loop is now in effect.

Split horizon and the invalid timers will eventually sort things out. Lazzeri is advertising 192.168.1.0 to Meusel, but Meusel continues to use the metrically closer route via Ruth. And since Ruth is the next-hop router, split horizon is in effect for 192.168.1.0 at Meusel's S1 interface. Meusel is also advertising 192.168.1.0 to Lazzeri, but Lazzeri sees Gehrig as metrically closer.

Lazzeri and Gehrig see each other as the next-hop router to 192.168.1.0, so they will not advertise the route to each other. The route will age in both of their route tables until the invalid timer expires (Example 11-5).

## Example 11-5. When the invalid timer for the route to 192.168.1.0 expires, the route is declared unreachable and the holddown timer is started.

```
Lazzeri#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
I    192.168.1.0/24 is possibly down, routing via 192.168.3.1, Serial0
I    192.168.2.0/24 [100/8576] via 192.168.3.1, 00:00:57, Serial0
C    192.168.3.0/24 is directly connected, Serial0
C    192.168.4.0/24 is directly connected, Serial1
I    192.168.5.0/24 [100/10476] via 192.168.4.2, 00:01:25, Serial1
I    192.168.6.0/24 is possibly down, routing via 192.168.3.1, Serial0
Lazzeri#
```

When Lazzeri's invalid timer expires, the route to 192.168.1.0 will be put into holddown. Although Meusel is advertising a route to that network, Lazzeri cannot accept it until the holddown timer expires. Example 11-6 shows that Lazzeri has finally accepted the route from Meusel, and Example 11-7 shows that Gehrig is successfully reaching 192.168.1.0 through Lazzeri. It took more than nine minutes for these two routers to converge, and the route they are using is still not the optimal route.

## Example 11-6. After the holddown timer for 192.168.1.0 expires, Lazzeri accepts the route advertised by Meusel.

```
Lazzeri#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
I    192.168.1.0/24 [100/14100] via 192.168.4.2, 00:00:27, Serial1
I    192.168.2.0/24 [100/8576] via 192.168.3.1, 00:00:02, Serial0
C    192.168.3.0/24 is directly connected, Serial0
C    192.168.4.0/24 is directly connected, Serial1
I    192.168.5.0/24 [100/10476] via 192.168.4.2, 00:00:28, Serial1
I   192.168.6.0/24 [100/12476] via 192.168.4.2, 00:00:28, Serial1
Lazzeri#
```

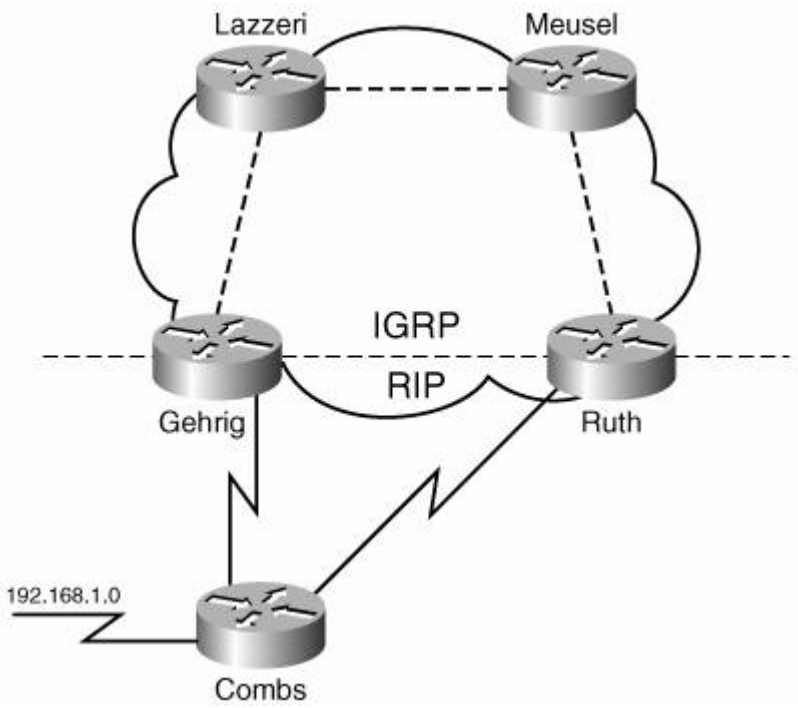
## Example 11-7. Gehrig can now reach subnet 192.168.1.0 via Lazzeri.

```
Gehrig#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
I    192.168.1.0/24 [100/16100] via 192.168.3.2, 00:00:32, Serial0
C    192.168.2.0/24 is directly connected, Ethernet0
C    192.168.3.0/24 is directly connected, Serial0
I    192.168.4.0/24 [100/10476] via 192.168.3.2, 00:00:33, Serial0
I    192.168.5.0/24 [100/12476] via 192.168.3.2, 00:00:33, Serial0
I    192.168.6.0/24 [100/14476] via 192.168.3.2, 00:00:33, Serial0
```

```
Gehrig#ping 192.168.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 52/72/108 ms
Gehrig#
```

Administrative distances can cause even worse problems than the sub-optimal routes, unpredictable behavior, and slow convergence of the previous example. For example, Figure 11-4 shows essentially the same network as in Figure 11-3, except that the links between the IGRP routers are Frame Relay PVCs. By default, IP split horizon is turned off on Frame Relay interfaces. As a result, permanent routing loops will form between Lazzeri and Gehrig and between Meusel and Ruth. Subnet 192.168.1.0 is unreachable from the IGRP domain.

Figure 11-4. Because IP split horizon is turned off by default on Frame Relay interfaces, permanent routing loops will form in this network.




Several tools and strategies exist to prevent routing loops when redistributing. The administrative distances can be manipulated, and route filters or route maps can be used. Chapter 13 covers route filters, and Chapter 14 covers route maps. These chapters also demonstrate techniques for changing administrative distances.

## Redistributing from Classless to Classful Protocols

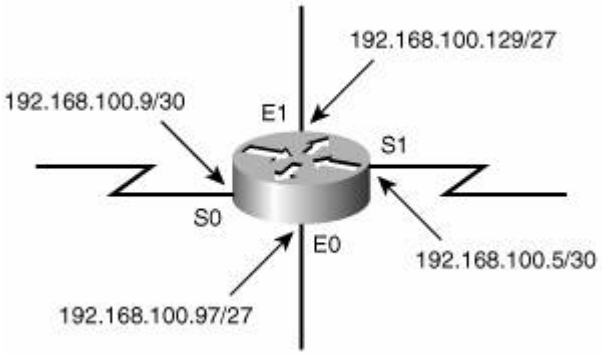
Careful consideration must be given to the effects of redistributing routes from a classless routing process domain into a classful domain. To understand why, it is necessary to first understand how a classful routing protocol reacts to variable subnetting. Recall from Chapter 5, "Routing Information Protocol (RIP)," that classful routing protocols do not advertise a mask with each route. For every route a classful router receives, one of two situations will apply:

- The router will have one or more interfaces attached to the major network.
- The router will have no interfaces attached to the major network.

In the first case, the router must use its own configured mask for that major network to correctly determine the subnet of a packet's destination address. In the second case, only the major network address itself can be included in the advertisement because the router has no way of knowing which subnet mask to use.

Figure 11-5 shows a router with four interfaces connected to subnets of 192.168.100.0. The network is variably subnettedtwo interfaces have 27-bit masks, and two interfaces have 30-bit masks. If the router is running a classful protocol such as IGRP, it cannot use the 27-bit mask to derive 30-bit subnets, and it cannot use the 30-bit mask to derive 27-bit subnets. So, how does the protocol cope with the conflicting masks?

## Figure 11-5. If this router is running a classful routing protocol, what mask should it choose?



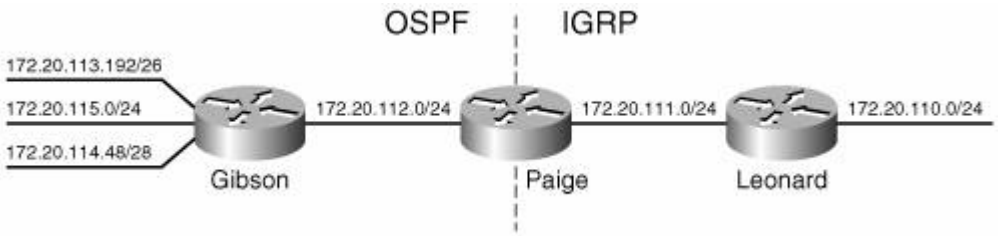

In Example 11-8, debugging is used to observe the IGRP advertisements sent by the router in Figure 11-5. Notice that subnet 192.168.100.128/27 is advertised out interface E0, which has a 27-bit mask, but neither 192.168.100.4/30 nor 192.168.100.8/30 is advertised out that interface. Similarly, 192.168.100.8/30 is advertised out interface S1, which has a 30-bit mask, but neither 192.168.100.96/27 nor 192.168.100.128/27 is advertised out that interface. The same situation applies to all four interfaces. Only subnets of 192.168.100.0 whose masks match the interface mask are advertised. As a result, IGRP-speaking neighbors on interfaces E0 and E1 will have no knowledge of the 30-bit subnets, and IGRP-speaking neighbors on interfaces S0 and S1 will have no knowledge of the 27-bit subnets.

## Example 11-8. A classful routing protocol will not advertise routes between interfaces whose masks do not match.

```
O'Neil#debug ip igrp transactions
IGRP protocol debugging is on
O'Neil#
IGRP: sending update to 255.255.255.255 via Ethernet0 (192.168.100.97)
      subnet 192.168.100.128, metric=1100
IGRP: sending update to 255.255.255.255 via Ethernet1 (192.168.100.129)
      subnet 192.168.100.96, metric=1100
IGRP: sending update to 255.255.255.255 via Serial0 (192.168.100.9)
      subnet 192.168.100.4, metric=8476
IGRP: sending update to 255.255.255.255 via Serial1 (192.168.100.5)
      subnet 192.168.100.8, metric=8476
O'Neil#
```

This behavior of only advertising routes between interfaces with matching masks also applies when redistributing from a classless routing protocol into a classful routing protocol. In Figure 11-6, the subnets of the OSPF domain are variably subnetted, and Paige is redistributing OSPF-learned routes into IGRP.

Figure 11-6. Paige is redistributing its OSPF-learned routes into IGRP.




As Example 11-9 shows, Paige knows about all of the subnets in both the OSPF and the IGRP domain. And because OSPF is classless, the router knows which masks are associated with each subnet connected to Gibson. Paige's IGRP process is using a 24-bit mask; therefore, 172.20.113.192/26 and 172.20.114.48/28 are not compatible and are not advertised (Example 11-10). Notice that IGRP does advertise 172.20.112.0/24 and 172.20.115.0/24. The result is that the only subnets within the OSPF domain that Leonard knows of are the ones with a 24-bit mask (Example 11-11).

Example 11-9. Paige knows about all six subnets of Figure 11-6, either from OSPF, IGRP, or a direct connection.

```
Paige#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
     172.20.0.0/16 is variably subnetted, 6 subnets, 3 masks
O       172.20.113.192/26 [110/74] via 172.20.112.1, 00:01:35, Ethernet1
C       172.20.112.0/24 is directly connected, Ethernet1
O       172.20.115.0/24 [110/80] via 172.20.112.1, 00:01:35, Ethernet1
I       172.20.110.0/24 [100/1600] via 172.20.111.1, 00:00:33, Ethernet0
C       172.20.111.0/24 is directly connected, Ethernet0
O       172.20.114.48/28 [110/74] via 172.20.112.1, 00:01:35, Ethernet1
Paige#
```

Example 11-10. Only the OSPF-learned routes with a 24-bit mask are successfully redistributed into the IGRP domain, which is also using a 24-bit mask.

```
Paige#debug ip igrp transactions
IGRP protocol debugging is on
Paige#
IGRP: received update from 172.20.111.1 on Ethernet0
```

```
      subnet 172.20.110.0, metric 1600 (neighbor 501)
IGRP: sending update to 255.255.255.255 via Ethernet0 (172.20.111.2)
      subnet 172.20.112.0, metric=1100
      subnet 172.20.115.0, metric=1100
Paige#
```

Example 11-11. The only subnets within the OSPF domain known at Leonard are the ones with a 24-bit mask. 172.20.113.192/26 and 172.20.114.48/28 are unreachable from here.

```
Leonard#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
     172.20.0.0/24 is subnetted, 4 subnets
I       172.20.112.0 [100/1200] via 172.20.111.2, 00:00:48, Ethernet1
I       172.20.115.0 [100/1200] via 172.20.111.2, 00:00:48, Ethernet1
C       172.20.110.0 is directly connected, Ethernet0
C       172.20.111.0 is directly connected, Ethernet1
Leonard#
```

PREV NEXT

# Configuring Redistribution

Redistribution is configured in two steps:

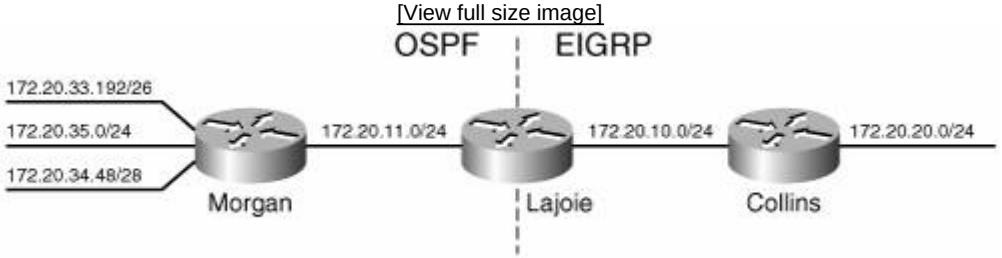
Step 1. In the routing protocol configuration that is to receive the redistributed routes, use the redistribute command to specify the source of the routes.

Step 2. Specify the metric to be assigned to the redistributed routes.

For example, the EIGRP configuration of Lajoie in Figure 11-7 is displayed in Example 11-12.

Figure 11-7. Configuring redistribution sample network.

[View full size image]




Example 11-12. Lajoie's EIGRP configuration shows OSPF being redistributed into the EIGRP process.

```
router eigrp 1
 redistribute ospf 1 metric 10000 100 255 1 15003
 passive-interface Ethernet1
 network 172.20.0.0
```

This configuration redistributes routes discovered by OSPF process 1 into EIGRP process 1. The metric portion of the command assigns EIGRP metrics to the routes. In order, the numbers specify the following:

- Bandwidth, in kilobits per second
- Delay, in tens of microseconds
- Reliability, as a fraction of 255
- Load, as a fraction of 255
- MTU, in octets

The OSPF configuration of Lajoie is in Example 11-13

Example 11-13. Lajoie's OSPF configuration shows the redistribution of

EIGRP routes into OSPF.

```
router ospf 1
 redistribute eigrp 1 metric 30 metric-type 1 subnets
 network 172.20.11.2 0.0.0.0 area 0
```

This configuration redistributes routes discovered by EIGRP process 1 into OSPF process 1. The metric portion of the command assigns an OSPF cost of 30 to each redistributed route. The redistribution makes Lajoie an ASBR in the OSPF domain, and the redistributed routes are advertised as external routes. The metric-type portion of the command specifies that the external type of the routes is E1. The subnets keyword, used only when redistributing routes into OSPF, specifies that subnet details will be redistributed. Without it, only major network addresses are redistributed. More will be said about the subnets keyword in the case studies.

An alternative method of specifying the metrics to be assigned to redistributed routes is to use the default-metric command. For example, the previous OSPF configuration can also be written as in Example 11-14.

Example 11-14. Lajoie's OSPF configuration uses the *default-metric* command to assign an OSPF metric to all routes redistributed into OSPF.

```
router ospf 1
 redistribute eigrp 1 metric-type 1 subnets
 default-metric 30
 network 172.20.11.2 0.0.0.0 area 0
```

The results of this configuration are exactly the same as the previous configuration. The default-metric command is useful when routes are being redistributed from more than one source. For example, suppose router Lajoie in Figure 11-7 was running not only EIGRP and OSPF but also RIP. The OSPF configuration might be as displayed in Example 11-15.

Example 11-15. Lajoie's OSPF configuration uses the *default-metric* command to assign an OSPF metric to routes redistributed from both EIGRP and RIP.

```
router ospf 1
 redistribute eigrp 1 metric-type 1 subnets
 redistribute rip metric-type 1 subnets
default-metric 30
 network 172.20.11.2 0.0.0.0 area 0
```

Here an OSPF cost of 30 will be assigned to all EIGRP- and RIP-learned routes.

The two methods of assigning metrics can also be used with each other. For example, suppose Lajoie were to be configured to redistribute OSPF and RIP into EIGRP, but that RIP routes were to be advertised with a different set of metrics than the OSPF routes. The configuration might be like Example 11-16.

Example 11-16. Lajoie's EIGRP configuration uses a metric with the *redistribution* command to assign metrics to addresses redistributed from RIP, and uses the *default metric* command to assign the EIGRP metric to all other redistributed addresses.

```
router eigrp 1
 redistribute ospf 1
 redistribute rip metric 50000 500 255 1 1500
 default-metric 10000 100 255 1 1500
 passive-interface Ethernet1
 network 172.20.0.0
```

The metrics assigned using the metric keyword with the redistribute command take precedence over metrics assigned with the default-metric command. RIP-learned routes will be advertised into EIGRP with the metrics specified on the redistribute rip line, and the OSPF-learned routes will be advertised with the metrics specified by the default-metric command.

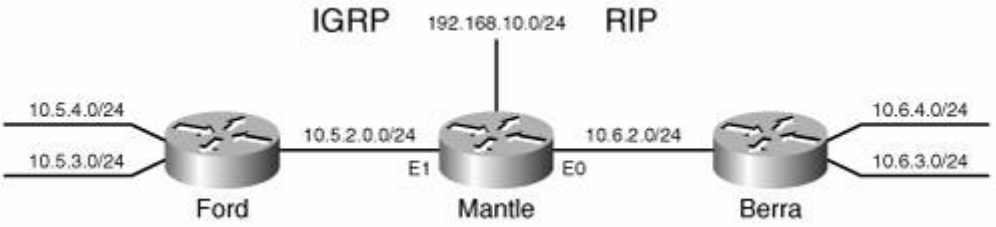
If neither the metric keyword nor the default-metric command specifies a metric, the metric will default to 20 for routes redistributed into OSPF and to 0 for routes redistributed into other protocols. The 0 metric will be understood by IS-IS, but not by RIP, whose hop count must be between 1 and 16. The 0 metric is also incompatible with the EIGRP multi-metric format. These two protocols must have the appropriate metrics assigned to any redistributed routes, or redistribution will not work. The following case studies examine techniques for configuring redistribution into the various IP IGPs. In addition, they are arranged so that the more generic issues of redistributing classful to classful, classless to classless, and classless to classful can be examined.

## Case Study: Redistributing IGRP and RIP

In the network of Figure 11-8, Ford is running IGRP, and Berra is running RIP. Mantle's routing configuration is shown in Example 11-17.

Figure 11-8. Ford is running IGRP, and Berra is running RIP. Mantle is performing redistribution.

[View full size image]




Example 11-17. Mantle is running both IGRP and RIP and performing redistribution between the two protocols, using both the default metric and the metric assigned with the *redistribution* command.

```
router rip
 redistribute igrp 1 metric 5
 passive-interface Ethernet1
 network 10.0.0.0
!
router igrp 1
 redistribute rip
 default-metric 1000 100 255 1 1500
 passive-interface Ethernet0
```

```
 network 10.0.0.0
```

Both methods of assigning metrics are used here for demonstration purposes. In most cases, a redistribution scheme as simple as this will use one method or the other.

Notice that Mantle is also connected to a stub network (192.168.10.0/24). In this case, the stub network should be advertised into the IGRP domain, but not into the RIP domain. One way to accomplish this configuration is to simply add the appropriate network statement under IGRP. However, doing so will create unnecessary IGRP broadcasts on the stub network. Another way to achieve the desired configuration is to use redistribution as in Example 11-18.

### Example 11-18. Mantle redistributes the connected stub network into IGRP.

```
router rip
 redistribute igrp 1 metric 5
 passive-interface Ethernet1
 network 10.0.0.0
!
router igrp 1
 redistribute connected
 redistribute rip
 default-metric 1000 100 255 1 1500
 passive-interface Ethernet0
 network 10.0.0.0
```

The redistribute connected command will redistribute all directly connected networks. If network 192.168.10.0/24 is to be advertised into the IGRP domain and the RIP domain, the configuration would be as in Example 11-19.

### Example 11-19. Mantle's configuration shows the connected stub network advertised into both RIP, assigning the metric during redistribution, and IGRP, assigning the metric using the *default metric* command.

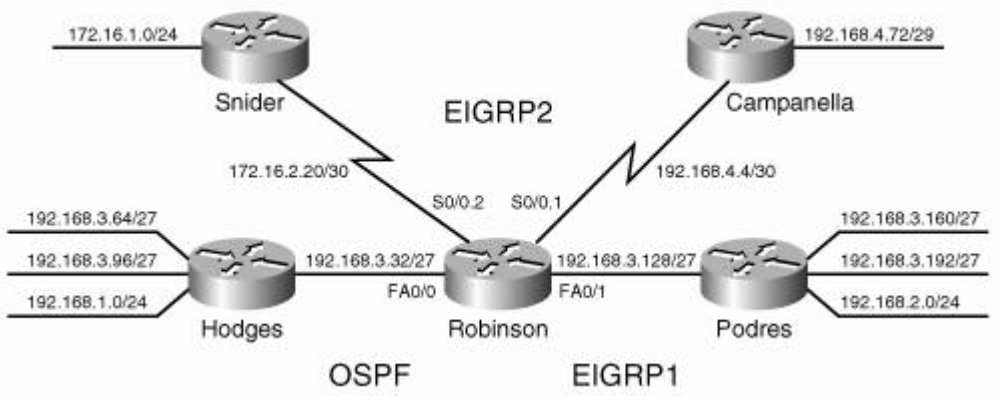
```
router rip
 redistribute connected metric 5
 redistribute igrp 1 metric 5
 passive-interface Ethernet1
 network 10.0.0.0
!
router igrp 1
 redistribute connected
 redistribute rip
 default-metric 1000 100 255 1 1500
 passive-interface Ethernet0
 network 10.0.0.0
```

## Case Study: Redistributing EIGRP and OSPF

The network of Figure 11-9 has an OSPF domain and two EIGRP domains. Router Hodges is running OSPF process 1. Podres is running EIGRP process 1, and on Snider and Campanella, EIGRP process 2 is running.

Figure 11-9. Hodges is running OSPF, and Podres is running EIGRP 1. Snider and Campanella are running EIGRP 2.

[View full size image]




Robinson has the configuration in Example 11-20.

## Example 11-20. Robinson is redistributing the EIGRP1, EIGRP2, and OSPF processes.

```
router eigrp 1
 redistribute ospf 1 metric 1000 100 1 255 1500
 redistribute eigrp 2 passive-interface FastEthernet0/0
 network 192.168.3.0
!
router eigrp 2
 redistribute ospf 1 metric 1000 100 1 255 1500
 redistribute eigrp 1 network 192.168.4.0
 network 172.16.0.0
!
router ospf 1
 redistribute eigrp 1
 redistribute eigrp 2 metric 100
 default-metric 50
 network 192.168.3.33 0.0.0.0 area 0
```

Notice that although redistribution must be configured between the EIGRP processes, no metrics are configured. The processes use the same metrics, so the metrics can be tracked accurately across the redistribution boundary. Example 11-21 shows Podres' route table. The redistributed routes are tagged as EIGRP external routes.

## Example 11-21. The route table of Podres in Figure 11-9.

```
Podres#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E  EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o  ODR
```

```
       P - periodic downloaded static route

Gateway of last resort is not set

172.16.0.0/16 is variably subnetted, 3 subnets, 3 masks
D EX    172.16.2.20/30 [170/2195456] via 192.168.3.129, 00:00:26, Ethernet0
D EX    172.16.0.0/16 [170/2195456] via 192.168.3.129, 00:00:26, Ethernet0
D EX    172.16.1.0/24 [170/2221056] via 192.168.3.129, 00:00:26, Ethernet0
     192.168.4.0/24 is variably subnetted, 3 subnets, 3 masks
D EX    192.168.4.72/29 [170/2221056] via 192.168.3.129, 00:00:26, Ethernet0
D EX    192.168.4.4/30 [170/2195456] via 192.168.3.129, 00:00:26, Ethernet0
D EX    192.168.4.0/24 [170/2195456] via 192.168.3.129, 00:00:27, Ethernet0
D EX 192.168.1.0/24 [170/2611200] via 192.168.3.129, 00:00:28, Ethernet0
C    192.168.2.0/24 is directly connected, Ethernet3
     192.168.3.0/24 is variably subnetted, 7 subnets, 2 masks
D EX    192.168.3.96/27 [170/2611200] via 192.168.3.129, 00:00:28, Ethernet0
D EX    192.168.3.64/27 [170/2611200] via 192.168.3.129, 00:00:28, Ethernet0
D       192.168.3.32/27 [90/284160] via 192.168.3.129, 00:00:35, Ethernet0
D       192.168.3.0/24 is a summary, 00:07:18, Null0
C       192.168.3.192/27 is directly connected, Ethernet2
C       192.168.3.160/27 is directly connected, Ethernet1
C       192.168.3.128/27 is directly connected, Ethernet0
Podres#
```

Example 11-22 shows Hodges' route table, which has some problems. Recall from Chapter 8, "OSPFv2," that routes redistributed into OSPF are either type 1 (E1) or type 2 (E2) external routes. The only routes that seem to have been redistributed, indicated by the E2 tag, are the major network addresses 192.168.2.0/24, 172.16.0.0/16, and 192.168.4.0/24. The reason for this is the absence of the `subnets` keyword in Robinson's redistribution statements. Without this keyword, only major network addresses within the non-OSPF process will be redistributed.

### Example 11-22. The route table of Hodges contains only major network redistributed routes, indicated by the E2 tag.

```
Hodges#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
O E2 172.16.0.0/16 [110/100] via 192.168.3.33, 00:24:41, FastEthernet0/0
O E2 192.168.4.0/24 [110/100] via 192.168.3.33, 00:24:41, FastEthernet0/0
C    192.168.1.0/24 is directly connected, Serial0/0
O E2 192.168.2.0/24 [110/50] via 192.168.3.33, 00:02:57, FastEthernet0/0
     192.168.3.0/27 is subnetted, 3 subnets
C       192.168.3.96 is directly connected, Serial0/0
C       192.168.3.64 is directly connected, Serial0/0
C       192.168.3.32 is directly connected, FastEthernet0/0
Hodges#
```

Robinson's configuration is changed to include the `subnets` keyword as displayed in Example 11-23.

Example 11-23. Robinson uses the *subnets* keyword when redistributing routes into OSPF so that subnets in addition to major network addresses will be redistributed.

```
router eigrp 1
 redistribute ospf 1 metric 1000 100 1 255 1500
 redistribute eigrp 2
 passive-interface FastEthernet0/0
 network 192.168.3.0
!
router eigrp 2
 redistribute ospf 1 metric 1000 100 1 255 1500
 redistribute eigrp 1
 network 192.168.4.0
 network 172.16.0.0
!
router ospf 1
 redistribute eigrp 1 subnets
 redistribute eigrp 2 metric 100 subnets
 default-metric 50 network 192.168.3.33 0.0.0.0 area 0
```

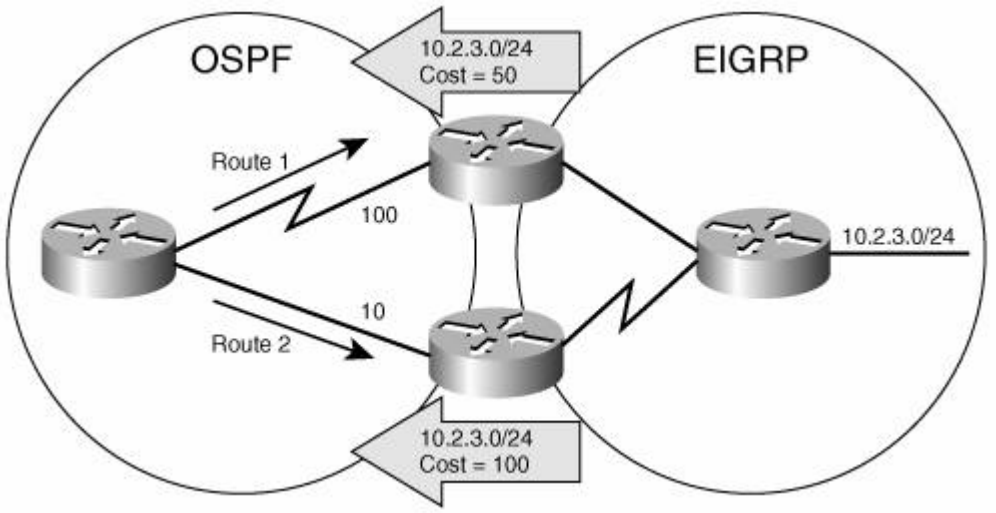
As a result of this change, the subnets in Figure 11-9 are in Hodges' route table (Example 11-24).

Example 11-24. After the *subnets* keyword is added to Robinson's redistribution configuration, Hodges has knowledge of all subnets.

```
Hodges#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 3 subnets, 3 masks
O E2    172.16.2.20/30 [110/100] via 192.168.3.33, 00:00:59, FastEthernet0/0
O E2    172.16.0.0/16 [110/100] via 192.168.3.33, 00:30:20, FastEthernet0/0
O E2    172.16.1.0/24 [110/100] via 192.168.3.33, 00:00:59, FastEthernet0/0
     192.168.4.0/24 is variably subnetted, 3 subnets, 3 masks
O E2    192.168.4.72/29 [110/100] via 192.168.3.33, 00:00:59, FastEthernet0/0
O E2    192.168.4.4/30 [110/100] via 192.168.3.33, 00:01:00, FastEthernet0/0
O E2    192.168.4.0/24 [110/100] via 192.168.3.33, 00:30:22, FastEthernet0/0
C    192.168.1.0/24 is directly connected, Serial0/0
O E2 192.168.2.0/24 [110/50] via 192.168.3.33, 00:08:38, FastEthernet0/0
     192.168.3.0/27 is subnetted, 6 subnets

C       192.168.3.96 is directly connected, Serial0/0
C       192.168.3.64 is directly connected, Serial0/0
C       192.168.3.32 is directly connected, FastEthernet0/0
O E2    192.168.3.192 [110/50] via 192.168.3.33, 00:01:12, FastEthernet0/0
O E2    192.168.3.160 [110/50] via 192.168.3.33, 00:01:12, FastEthernet0/0
O E2    192.168.3.128 [110/50] via 192.168.3.33, 00:01:12, FastEthernet0/0
Hodges#
```

By default, external routes are redistributed into OSPF as type 2 routes. As discussed in Chapter 8, E2 routes include only the external cost of the route. This fact can be important when a single destination is reachable by more than one external route, as shown in Figure 11-10. In this network, one router is redistributing the route to 10.2.3.0/24 with a cost of 50, and the other router is redistributing a different route to the same destination with a cost of 100. If the routes are advertised as E2, the costs of the links within the OSPF domain will not be added. As a result, the router internal to the OSPF domain will choose route 1 to reach 10.2.3.0/24.

Figure 11-10. If the routes to 10.2.3.0/24 are advertised as E2, route 1 will have a cost of 50 and route 2 will have a cost of 100. If the routes are advertised as E1, route 1 will have a cost of 150 and route 2 will have a cost of 110.




If the routes to 10.2.3.0/24 in Figure 11-10 are redistributed as E1, the costs of the links within the OSPF domain will be added to the redistributed costs. As a result, the router internal to the OSPF domain would choose route 2, with a cost of 110 (100 + 10) over route 1, with a cost of 150 (50 + 100).

Robinson, in Figure 11-9, is redistributing EIGRP 1 with a cost of 50 and redistributing EIGRP 2 with a cost of 100. Example 11-24 shows that, at Hodges, the routes to the EIGRP 1 subnets still have a cost of 50, and the routes to the EIGRP 2 subnets still have a cost of 100. The cost of the FastEthernet link between Hodges and Robinson has not been added to the routes.

To redistribute routes into OSPF as E1, the keyword metric-type 1 is added to the redistribution command. In the configuration in Example 11-25, Robinson continues to redistribute EIGRP 1 as E2, but EIGRP 2 is redistributed as E1.

Example 11-25. Robinson configures routes redistributed into OSPF from EIGRP2 as external type 1 routes.

```
router eigrp 1
 redistribute ospf 1 metric 1000 100 1 255 1500
 redistribute eigrp 2
 passive-interface FastEthernet0/0
 network 192.168.3.0
!
router eigrp 2
 redistribute ospf 1 metric 1000 100 1 255 1500
```

```
 redistribute eigrp 1
 network 192.168.4.0
 network 172.16.0.0
!
router ospf 1
 redistribute eigrp 1 subnets
 redistribute eigrp 2 metric 100 metric-type 1 subnets
 default-metric 50 network 192.168.3.33 0.0.0.0 area 0
```

Example 11-26 shows Hodges's route table after Robinson is reconfigured. All the routes to destinations within the EIGRP 1 domain still have a cost of 50, but the routes to destinations within the EIGRP 2 domain now have a cost of 101 (the redistributed cost plus the default cost of 1 for the FastEthernet link between Robinson and Hodges).

### Example 11-26. Robinson's configuration has been changed so that the subnets of 192.168.4.0 and 172.16.0.0 are being advertised as type 1 external routes.

```
Hodges#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 3 subnets, 3 masks
O E1    172.16.2.20/30 [110/101] via 192.168.3.33, 09:01:00, FastEthernet0/0
O E1    172.16.0.0/16 [110/101] via 192.168.3.33, 00:01:02, FastEthernet0/0
O E1    172.16.1.0/24 [110/101] via 192.168.3.33, 09:01:00, FastEthernet0/0
     192.168.4.0/24 is variably subnetted, 3 subnets, 3 masks
O E1    192.168.4.72/29 [110/101] via 192.168.3.33, 00:00:57, FastEthernet0/0
O E1    192.168.4.4/30 [110/101] via 192.168.3.33, 00:01:03, FastEthernet0/0
O E1    192.168.4.0/24 [110/101] via 192.168.3.33, 00:01:03, FastEthernet0/0
C    192.168.1.0/24 is directly connected, Serial0/0
O E2 192.168.2.0/24 [110/50] via 192.168.3.33, 09:01:01, FastEthernet0/0
     192.168.3.0/27 is subnetted, 6 subnets
C       192.168.3.96 is directly connected, Serial0/0
C       192.168.3.64 is directly connected, Serial0/0
C       192.168.3.32 is directly connected, FastEthernet0/0
O E2    192.168.3.192 [110/50] via 192.168.3.33, 09:01:06, FastEthernet0/0
O E2    192.168.3.160 [110/50] via 192.168.3.33, 09:01:06, FastEthernet0/0
O E2    192.168.3.128 [110/50] via 192.168.3.33, 09:01:06, FastEthernet0/0
Hodges#
```

## Case Study: Redistribution and Route Summarization

The Cisco EIGRP, OSPFv2, OSPFv3, and IS-IS implementations have the capability to summarize redistributed routes. This case study examines summarization for EIGRP and OSPF for IPv4; the following two case studies examine OSPFv3 for IPv6 and IS-IS summarization.

The first thing to note is that summarization is useful only if the IP subnet addresses have been planned for summarization. For example, the subnets of 192.168.3.0 within the OSPF domain in Figure 11-9 all fall under the summary address 192.168.3.0/25. The subnets of the same major address within the EIGRP 1 domain all fall under the summary address 192.168.3.128/25. If

subnet 192.168.3.0/27 were to be connected to Podres, that single destination would have to be advertised separately from the summary address. Although advertising such a single destination will have little adverse impact, advertising a large number of subnets outside of the range of the summary address will reduce the benefits of summarization.

The command summary-address specifies a summary address and mask to an OSPF process. Any more-specific subnet addresses that fall within the range of the specified summary address will be suppressed. Note that this command is used only to summarize external routes at ASBRs; summarization of internal OSPF routes at ABRs is accomplished with the area range command, as discussed in Chapter 8.

At Robinson in Figure 11-9, the EIGRP 1 subnets are summarized into the OSPF domain with 192.168.3.128/25, and the EIGRP 2 subnets are summarized with 172.16.0.0/16, as displayed in Example 11-27.

## Example 11-27. Robinson summarizes external addresses advertised into OSPF.

```
router eigrp 1
 redistribute ospf 1 metric 1000 100 1 255 1500
 redistribute eigrp 2
 passive-interface FastEthernet0/0
 network 192.168.3.0
!
router eigrp 2
 redistribute eigrp 1
 network 192.168.4.0
 network 172.16.0.0
!
router ospf 1
 summary-address 192.168.3.128 255.255.255.128
 summary-address 172.16.0.0 255.255.0.0
 redistribute eigrp 1 subnets
 redistribute eigrp 2 metric 100 metric-type 1 subnets
 default-metric 50
 network 192.168.3.33 0.0.0.0 area 0
```

Compare Example 11-28 with Example 11-26. In Example 11-28, Hodges' route table contains the specified summary addresses. The subnet addresses within the summary range have been suppressed at the redistribution point. Notice, also, that no summarization was configured for 192.168.4.0/24, so the subnets of that major address are still in the route table.

## Example 11-28. Robinson is summarizing 192.168.3.128/25 and 172.16.0.0/16, so no more-specific subnets within those ranges appear in the route table of Hodges.

```
Hodges#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
O E1 172.16.0.0/16 [110/101] via 192.168.3.33, 00:11:55, FastEthernet0/0
     192.168.4.0/24 is variably subnetted, 3 subnets, 3 masks
```

```
O E1    192.168.4.72/29 [110/101] via 192.168.3.33, 00:03:56, FastEthernet0/0
O E1    192.168.4.4/30 [110/101] via 192.168.3.33, 00:11:55, FastEthernet0/0
O E1    192.168.4.0/24 [110/101] via 192.168.3.33, 00:11:55, FastEthernet0/0
C    192.168.1.0/24 is directly connected, Serial0/0
O E2 192.168.2.0/24 [110/50] via 192.168.3.33, 00:03:57, FastEthernet0/0
     192.168.3.0/24 is variably subnetted, 4 subnets, 2 masks
C       192.168.3.96/27 is directly connected, Serial0/0
C       192.168.3.64/27 is directly connected, Serial0/0
C       192.168.3.32/27 is directly connected, FastEthernet0/0
O E2    192.168.3.128/25 [110/50] via 192.168.3.33, 00:00:37, FastEthernet0/0
```

Summarization for EIGRP is interface-specific. That is, instead of specifying the summary address and mask under the routing process, they are specified under individual interfaces. This system provides the flexibility to advertise different summary routes out different interfaces of the same process. The command ip summary-address eigrp *process-id* specifies the summary address and mask and the EIGRP process into which the summary is to be advertised.

In the configuration in Example 11-29, Robinson will advertise summary addresses 192.168.3.0/25, 172.16.0.0/16, and 192.168.4.0/24 into EIGRP 1.

## Example 11-29. Robinson summarizes routes advertised into EIGRP 1.

```
interface FastEthernet0/0
 ip address 192.168.3.33 255.255.255.224
!
interface FastEthernet0/1
 ip address 192.168.3.129 255.255.255.224
 ip summary-address eigrp 1 192.168.3.0 255.255.255.128
 ip summary-address eigrp 1 172.16.0.0 255.255.0.0
 ip summary-address eigrp 1 192.168.4.0 255.255.255.0
!
interface Serial0/0.1
 ip address 192.168.4.5 255.255.255.252
 ip summary-address eigrp 2 192.168.3.0 255.255.255.0
!
interface Serial0/0.2
 ip address 172.16.2.21 255.255.255.252
 ip summary-address eigrp 2 192.168.0.0 255.255.0.0
!
router eigrp 1
 redistribute ospf 1 metric 1000 100 1 255 1500
 redistribute eigrp 2
 passive-interface FastEthernet0/0
 network 192.168.3.0
!
router eigrp 2
 redistribute eigrp 1
 network 192.168.4.0
 network 172.16.0.0
!
router ospf 1
 summary-address 192.168.3.128 255.255.255.128
 summary-address 172.16.0.0 255.255.0.0
 redistribute eigrp 1 subnets
 redistribute eigrp 2 metric 100 metric-type 1 subnets
 default-metric 50
 network 192.168.3.33 0.0.0.0 area 0
```

Example 11-30 shows the route table of Podres. As with OSPF summarization, EIGRP summarization suppresses the advertisement of subnets within the summary range. Unlike OSPF, Podres' route table shows that summary routes advertised into EIGRP are not tagged as external routes.

## Example 11-30. The route table of Podres, showing summary routes 192.168.3.0/25, 192.168.4.0/24, and 172.16.0.0/16.

```
Podres#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E  EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o  ODR
       P - periodic downloaded static route
Gateway of last resort is not set
D    172.16.0.0/16 [90/2195456] via 192.168.3.129, 00:00:12, Ethernet0
D    192.168.4.0/24 [90/2195456] via 192.168.3.129, 00:00:12, Ethernet0
D EX 192.168.1.0/24 [170/2611200] via 192.168.3.129, 00:00:12, Ethernet0
C    192.168.2.0/24 is directly connected, Ethernet3
     192.168.3.0/24 is variably subnetted, 6 subnets, 3 masks
D       192.168.3.0/25 [90/284160] via 192.168.3.129, 00:00:12, Ethernet0
D       192.168.3.0/24 is a summary, 00:00:12, Null0
C       192.168.3.192/27 is directly connected, Ethernet2
C       192.168.3.160/27 is directly connected, Ethernet1
D EX    192.168.3.128/25 [170/2611200] via 192.168.3.129, 00:00:13, Ethernet0
C       192.168.3.128/27 is directly connected, Ethernet0
D EX 192.168.0.0/16 [170/284160] via 192.168.3.129, 00:00:13, Ethernet0
Podres#
```

Robinson is advertising EIGRP summary routes of 192.168.3.0/24 to Campanella and 192.168.0.0/16 to Snider. Example 11-31 shows Campanella's route table, and Example 11-32 shows Snider's route table.

## Example 11-31. Campanella's route table after summarization is configured at Robinson.

```
Campanella#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E  EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o  ODR
       P - periodic downloaded static route
Gateway of last resort is not set
D EX 172.16.0.0/16 [170/2681856] via 192.168.4.5, 00:35:12, Serial0.1
     192.168.4.0/24 is variably subnetted, 3 subnets, 3 masks
C       192.168.4.72/29 is directly connected, Ethernet0
C       192.168.4.4/30 is directly connected, Serial0.1
D EX    192.168.4.0/24 [170/2681856] via 192.168.4.5, 00:35:12, Serial0.1
D EX 192.168.1.0/24 [170/3097600] via 192.168.4.5, 00:35:12, Serial0.1
D EX 192.168.2.0/24 [170/2300416] via 192.168.4.5, 00:35:12, Serial0.1
D    192.168.3.0/24 [90/2172416] via 192.168.4.5, 00:35:13, Serial0.1
Campanella#
```

## Example 11-32. Snider's route table after summarization is configured at Robinson.

```
Snider#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 3 subnets, 3 masks
C       172.16.2.20/30 is directly connected, Serial0/0.1
D EX    172.16.0.0/16 [170/2681856] via 172.16.2.21, 00:07:59, Serial0/0.1
C       172.16.1.0/24 is directly connected, Ethernet0/0
D    192.168.0.0/16 [90/2195456] via 172.16.2.21, 00:07:59, Serial0/0.1
Snider#
```

Looking again at Example 11-30, notice the entry for the summary route 192.168.3.128/25. This entry might surprise you because that summary address is advertised into OSPF, not EIGRP. Notice also that the route is marked as an external route, indicating that it was redistributed into the EIGRP domain. What has happened is that the summary route was advertised into OSPF and was then redistributed back into EIGRP from the OSPF domain. Hence, the unexpected entry at Podres. This is the same reason why you see 172.16.0.0/16 and 192.168.4.0/24 as external routes on Campanella. These summary addresses are advertised to Podres, in EIGRP 1, and then redistributed back into EIGRP 2. Snider's route table shows the route to 172.16.0.0/16, but not 192.168.4.0/24, because Snider has an entry for the summarized address 192.168.0.0/16.

Now suppose subnet 192.168.3.192/27 were to become inaccessible. Podres would forward packets destined for that subnet to the less-specific route 192.168.3.128/25. The packet would be sent into the OSPF domain, where you might expect the summary route 192.168.3.128/25 to cause the packet to be sent right back to Podres.

In fact, this situation will not occur.

Robinson's route table (Example 11-33) has numerous entries for summary routes that show interface Null0 as a connected interface. The null interface is a software-only interface to nowherepackets routed to it are dropped. With some exceptions,[2] whenever a router generates a summary address, the router will also create a route for that address that goes to the null interface. If Robinson receives a packet destined for 192.168.3.192/27 and that subnet is no longer reachable, the router will forward the packet to the null interface. The routing loop is broken in one hop.

[2] In some older versions of IOS, OSPF inter-area summarization, for example, does not automatically create a summary route to the null interface. It must be configured statically.

## Example 11-33. Robinson's route table. Because the router is originating many summary routes, there are many entries for summaries whose connected interface is Null0. This is a safeguard against routing loops.

```
Robinson#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
       E1 - OSPF external type 1, E2 - OSPF external type 2, E  EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o  ODR
       P - periodic downloaded static route
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 3 subnets, 3 masks
C       172.16.2.20/30 is directly connected, Serial0/0.2
D       172.16.0.0/16 is a summary, 00:19:44, Null0
D       172.16.1.0/24 [90/2195456] via 172.16.2.22, 00:17:08, Serial0/0.2
     192.168.4.0/24 is variably subnetted, 3 subnets, 3 masks
D       192.168.4.72/29 [90/2172416] via 192.168.4.6, 00:15:17, Serial0/0.1
C       192.168.4.4/30 is directly connected, Serial0/0.1
D       192.168.4.0/24 is a summary, 00:20:53, Null0
O    192.168.1.0/24 [110/74] via 192.168.3.34, 00:21:14, FastEthernet0/0
D    192.168.2.0/24 [90/665600] via 192.168.3.130, 00:19:41, FastEthernet0/1
     192.168.3.0/24 is variably subnetted, 8 subnets, 3 masks
O       192.168.3.96/27 [110/74] via 192.168.3.34, 00:21:15, FastEthernet0/0
O       192.168.3.64/27 [110/74] via 192.168.3.34, 00:21:15, FastEthernet0/0
C       192.168.3.32/27 is directly connected, FastEthernet0/0
D       192.168.3.0/24 is a summary, 00:19:01, Null0
D       192.168.3.0/25 is a summary, 00:19:47, Null0
D       192.168.3.192/27 [90/665600] via 192.168.3.130, 00:19:42, FastEthernet0/1
D       192.168.3.160/27 [90/665600] via 192.168.3.130, 00:19:42, FastEthernet0/1
C       192.168.3.128/27 is directly connected, FastEthernet0/1
D    192.168.0.0/16 is a summary, 00:19:00, Null0
Robinson#
```

Summary routes to null interfaces are very helpful for preventing loops, and their use is described in greater detail in Chapter 12, "Default Routes and On-Demand Routing." However, the redistribution of incorrect routing information should not be allowed to happen at all. Suppose that instead of being one hop away from Robinson, Podres is 10 hops away. The misdirected packet would have to travel a long way before being dropped. This example demonstrates the need to carefully regulate route advertisements when using *mutual redistribution*that is, when two routing protocols are redistributing their routes into each other. In such cases the use of route filters, described in Chapter 13, "Route Filtering," or route maps, as described in Chapter 14, "Route Maps," is essential.

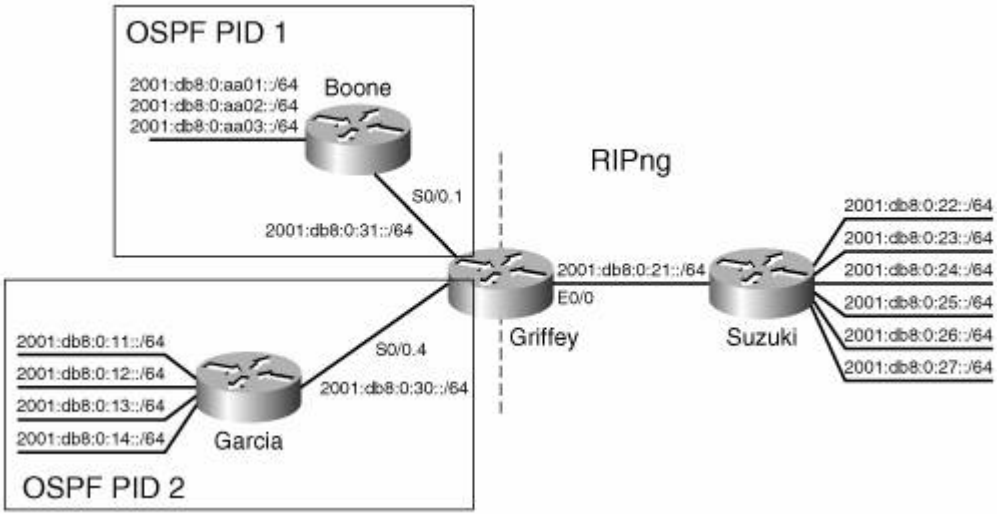
The previous scenario also demonstrates the trade-off of using summarization. Although the size of the route table is reduced, saving memory and processor cycles, route precision is also reduced. As the network grows more complex, that loss of detail increases the possibility of routing errors.

## Case Study: Redistributing OSPFv3 and RIPng

IPv6 is running on the network in Figure 11-11. Router Griffey is the connection point between three distinct networks. Two networks are running OSPFv3, the other is running RIPng. Routes are redistributed between the routing processes by Griffey.

Figure 11-11. IPv6 is being routed with both OSPFv3 and RIPng, and a single router is redistributing between the routing protocols.

[View full size image]




Griffey's configuration is displayed in Example 11-34.

## Example 11-34. Griffey is configured with IPv6, OSPFv3 process 1, OSPFv3 process 2, and RIPng, and is redistributing between the processes.

```
interface Ethernet 0/0
 ipv6 address 2001:db8:0:21::1/64
 ipv6 rip Mariners enable
!
interface Serial 0/0.1 point-to-point
 ipv6 address 2001:db8:0:31::1/64
 ipv6 ospf 1 area 1
!
interface Serial 0/0.4 point-to-point
 ipv6 address 2001:db8:0:30::1/64
 ipv6 ospf 2 area 20
!
ipv6 router ospf 1
 redistribute rip Mariners metric-type 1 tag 4
!
ipv6 router ospf 2
 redistribute rip Mariners
!
ipv6 router rip Mariners
 redistribute ospf 1 metric 5
 redistribute ospf 2 metric 10
```

Griffey is redistributing routes known via the RIPng process called Mariners into OSPFv3 process ID 1 with a metric-type of external, type 1. Also, a tag value of 4 is added to these routes. The tag can later be used to identify routes advertised by this RIPng process. Boone's IPv6 route table (Example 11-35) shows these routes.

## Example 11-35. IPv6 route table shows routes redistributed into OSPF from RIP, with a tag of 4 and external type 1 metrics.

```
Boone#show ipv6 route ospf
IPv6 Routing Table - 26 entries
```

```
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OE1  2001:DB8:0:22::/64 [110/84], tag 4
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE1  2001:DB8:0:23::/64 [110/84], tag 4
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE1  2001:DB8:0:24::/64 [110/84], tag 4
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE1  2001:DB8:0:25::/64 [110/84], tag 4
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE1  2001:DB8:0:26::/64 [110/84], tag 4
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE1  2001:DB8:0:27::/64 [110/84], tag 4
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
Boone#
```

Griffey advertises the RIPng routes into OSPFv3 process ID 2 with the default metric-type (external type 2) and no tag value. Garcia's route table (Example 11-36) shows the redistributed routes.

## Example 11-36. IPv6 route table shows redistributed routes with the default metric-type of external type 2.

```
Garcia#show ipv6 route ospf
IPv6 Routing Table - 26 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OE2  2001:DB8:0:22::/64 [110/20]
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE2  2001:DB8:0:23::/64 [110/20]
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE2  2001:DB8:0:24::/64 [110/20]
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE2  2001:DB8:0:25::/64 [110/20]
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE2  2001:DB8:0:26::/64 [110/20]
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
OE2  2001:DB8:0:27::/64 [110/20]
     via  FE80::207:85FF:FE6B:EA20, Serial0/0.1
Garcia#
```

The subnets keyword is not used with OSPFv3 for IPv6. All IPv6 routing protocols include IPv6 prefixes, which always include the address and the prefix length. Redistribution between protocols includes all prefixes unless filtering is performed to limit the prefixes advertised. Route filtering is discussed in Chapter 13 and Chapter 14.

Griffey also redistributes the routes known via both OSPFv3 processes into RIPng. OSPF 1 routes are advertised with a metric of 5 and OSPF 2 routes are advertised with a metric of 10. Example 11-37 shows Suzuki's IPv6 route table.

## Example 11-37. The RIPng route table shows routes redistributed into RIPng with different metrics.

```
Suzuki#show ipv6 route rip
IPv6 Routing Table - 19 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R   2001:DB8:0:11::/64 [120/11]
     via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R   2001:DB8:0:12::/64 [120/11]
     via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R   2001:DB8:0:13::/64 [120/11]
     via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R   2001:DB8:0:14::/64 [120/11]
     via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R   2001:DB8:0:AA01::/64 [120/6]
     via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R   2001:DB8:0:AA02::/64 [120/6]
     via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R   2001:DB8:0:AA03::/64 [120/6]
     via FE80::207:85FF:FE6B:EA20, Ethernet0/0
Suzuki#
```

The routes from Boone have a metric of 6 and the routes from Garcia have a metric of 11.

Route summarization is now going to be configured on Griffey. RIPng does not have a method for summarizing the routes that are advertised into it when redistributing from another protocol, but Griffey's Ethernet interface connecting to Suzuki can be configured to summarize the RIPng prefixes that are advertised out the interface. OSPFv3 summarizes the RIPng prefixes as they are redistributed into the routing process. Griffey's configuration is displayed in Example 11-38.

## Example 11-38. Griffey summarizes RIPng routes as they are advertised out Ethernet0/0, and also summarizes routes as they are redistributed into OSPFv3.

```
interface Ethernet 0/0
 ipv6 address 2001:db8:0:21::1/64
 ipv6 rip Mariners enable
 ipv6 rip Mariners summary-address 2001:DB8:0:AA00::/62
 ipv6 rip Mariners summary-address 2001:DB8:0:10::/61
!
interface Serial 0/0.1 point-to-point
 ipv6 address 2001:db8:0:31::1/64
 ipv6 ospf 1 area 1
!
interface Serial 0/0.4 point-to-point
 ipv6 address 2001:db8:0:30::1/64
 ipv6 ospf 2 area 20
!
ipv6 router ospf 1
 redistribute rip Mariners metric-type 1 tag 4
 summary-prefix 2001:DB8:0:20::/61 tag 4
!
ipv6 router ospf 2
 redistribute rip Mariners
```

```
 summary-prefix 2001:DB8:0:20::/61
!
ipv6 router rip Mariners
 redistribute ospf 1 metric 5
 redistribute ospf 2 metric 10
```

The new route tables show the summarized routes and show that the more specific routes have been removed (Example 11-39).

## Example 11-39. Route tables include summarized routes but no longer include the more specific routes that fall into the summarized range.

```
Boone#show ipv6 route ospf
IPv6 Routing Table - 21 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OE1  2001:DB8:0:20::/61 [110/84], tag 4
     via FE80::207:85FF:FE6B:EA20, Serial0/0.1
Boone#
```

---

```
Garcia#show ipv6 route ospf
IPv6 Routing Table - 21 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
OE2  2001:DB8:0:20::/61 [110/20]
     via FE80::207:85FF:FE6B:EA20, Serial0/0.1
Garcia#
```

---

```
Suzuki#show ipv6 route rip
IPv6 Routing Table - 19 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R    2001:DB8:0:10::/61 [120/11]
      via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R    2001:DB8:0:20::/61 [120/6]
      via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R    2001:DB8:0:AA00::/62 [120/6]
      via FE80::207:85FF:FE6B:EA20, Ethernet0/0
Suzuki#
```

Notice that the summary route 2001:db8:0:20::/61 is being advertised by Griffey to Suzuki. This is the summary route created for the prefixes advertised into OSPFv3. This summary route is being advertised from OSPF back into RIPng. The routers in the RIPng domain will still be able to route to the more specific prefixes correctly. Suzuki's full IPv6 route table (Example 11-40) shows that the prefixes that are included in the summary 2001:db8:0:20::/61 are directly connected to the router. The router will forward traffic to the more specific prefixes, rather than to the summary. A downside of having this summary re-advertised back into the RIPng domain is that packets destined to an address that falls into the summary range, but does not actually exist

in the network, will get forwarded to Griffey before being dropped. The route can be filtered during route redistribution so that only prefixes known to reside in the OSPFv3 domain are advertised into the RIPng domain. Route filtering during redistribution is discussed in Chapter 13 and Chapter 14.

### Example 11-40. Suzuki's RIPng route table shows a summary route and more specific routes that fall within the summary route's range. The router will forward traffic to the more specific routes when possible.

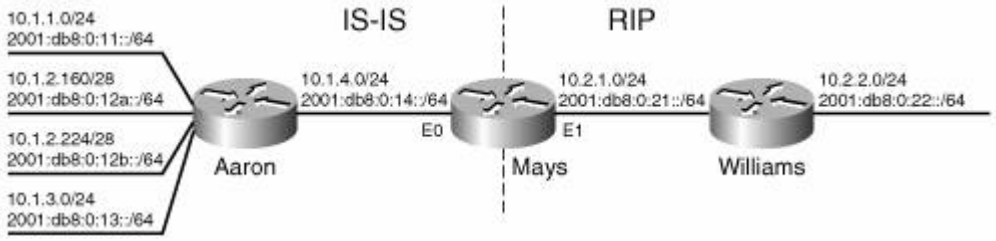
```
Suzuki#show ipv6 route
IPv6 Routing Table - 19 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R    2001:DB8:0:10::/61 [120/11]
      via FE80::207:85FF:FE6B:EA20, Ethernet0/0
R    2001:DB8:0:20::/61 [120/6]
      via FE80::207:85FF:FE6B:EA20, Ethernet0/0
C    2001:DB8:0:21::/64 [0/0]
      via ::, Ethernet0/0
L    2001:DB8:0:21::2/128 [0/0]
      via ::, Ethernet0/0
C    2001:DB8:0:22::/64 [0/0]
      via ::, Serial0/0
L    2001:DB8:0:22::1/128 [0/0]
      via ::, Serial0/0
C    2001:DB8:0:23::/64 [0/0]
      via ::, Serial0/0
L    2001:DB8:0:23::1/128 [0/0]
      via ::, Serial0/0
C    2001:DB8:0:24::/64 [0/0]
      via ::, Serial0/0
L    2001:DB8:0:24::1/128 [0/0]
      via ::, Serial0/0
C    2001:DB8:0:25::/64 [0/0]
      via ::, Serial0/0
L    2001:DB8:0:25::1/128 [0/0]
      via ::, Serial0/0
C    2001:DB8:0:26::/64 [0/0]
      via ::, Serial0/0
L    2001:DB8:0:26::1/128 [0/0]
      via ::, Serial0/0
C    2001:DB8:0:27::/64 [0/0]
      via ::, Serial0/0
L    2001:DB8:0:27::1/128 [0/0]
      via ::, Serial0/0
R    2001:DB8:0:AA00::/62 [120/6]
      via FE80::207:85FF:FE6B:EA20, Ethernet0/0
L    FE80::/10 [0/0]
      via ::, Null0
L    FF00::/8 [0/0]
      via ::, Null0
Suzuki#
```

## Case Study: Redistributing IS-IS and RIP/RIPng

In the network of Figure 11-12, Aaron is running IS-IS, Williams is running RIPv1 for IPv4 and RIPng for IPv6, and Mays is redistributing.

Figure 11-12. Router Mays is redistributing RIP into IS-IS and IS-IS into RIP.

[View full size image]




Mays's IS-IS and RIP IPv4 and IPv6 configuration is displayed in Example 11-41.

Example 11-41. Mays is configured for IS-IS, RIP, and RIPng.

```
interface Ethernet1
 description to Williams
 ip address 10.2.1.1 255.255.255.0
 ipv6 address 2001:db8:0:21::1/64
 ipv6 rip baseball enable
!
interface Ethernet0
 description to Aaron
 ip address 10.1.4.2 255.255.255.0
 ip router isis
 ipv6 address 2001:db8:0:14::2/64
 ipv6 router isis

router rip
 passive-interface Ethernet0
 network 10.0.0.0
 redistribute isis level-1-2 metric 1
!
router isis
 net 01.0001.0000.0c76.5432.00
 redistribute rip metric 0 metric-type internal level-2
 address-family ipv6
  redistribute rip baseball metric 0 metric-type internal level-2
!
ipv6 router rip baseball
 redistribute isis level-1-2 metric 1
```

Routes may be redistributed into IS-IS with either internal or external metrics (internal is the default) and as either level 1 or level 2 routes (level 2 is the default). The metric type determines the base metric value of the redistributed routes. The metric value of an internal metric type will be lower then 64. The metric value of an external metric will be between 64 and 128. If the metric type is internal, the initial metric of the redistributed route will be the number specified by

the metric value (0 in this example), or 0 if no metric is specified. The redistributed route in this example gets added to Mays's IS-IS database entry with a metric of 0. If the metric value of 5 were specified, the entry in Mays's IS-IS database would be 5. An external type of metric starts with a base value of 64. If the metric value of 5 is specified, the metric value for the entry in Mays's IS-IS database would be 64 + 5, or 69. In the example shown, the RIP routes are redistributed as internal, level 2 routes with the default metric of 0. Example 11-42 shows the redistributed routes in Aaron's IPv4 route table, and Example 11-43 shows the redistributed route in the IPv6 route table.

### Example 11-42. Aaron's IPv4 route table shows the redistributed RIP routes.

```
Aaron#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
     10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
C     10.1.3.0/24 is directly connected, Ethernet4
i L2  10.2.1.0/24 [115/10] via 10.1.4.2, Ethernet0
i L2  10.2.2.0/24 [115/10] via 10.1.4.2, Ethernet0
C     10.1.1.0/24 is directly connected, Ethernet1
C     10.1.4.0/24 is directly connected, Ethernet0
C     10.1.2.160/28 is directly connected, Ethernet2
C     10.1.2.224/28 is directly connected, Ethernet3
Aaron#
```

### Example 11-43. Aaron's IPv6 route table shows the redistributed RIP routes.

```
Aaron#show ipv6 route isis
IPv6 Routing Table - 13 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
I2 2001:DB8:0:22::/64 [115/10]
    via FE80::204:C1FF:FE50:E700, Ethernet0
Aaron#
```

The IPv6 route table on Aaron shows that the address connected to Mays, 2001:db8:0:21::/64, is not advertised into IS-IS. An additional command, `redistribute connected`, is added to Mays in Example 11-44 to redistribute the connected addresses, and the resulting IPv6 route table is shown in Example 11-45.

### Example 11-44. *redistribute connected* is added to Mays, so the directly connected IPv6 prefix can be advertised into IS-IS.

```
router isis
 net 01.0001.0000.0c76.5432.00
 redistribute rip metric 0 metric-type internal level-2
```

```
 address-family ipv6
  redistribute rip baseball metric 0 metric-type internal level-2
  redistribute connected metric 0 metric-type internal level-2
!
ipv6 router rip baseball
 redistribute isis level-1-2 metric 1
 redistribute connected metric 1
```

## Example 11-45. Aaron's IPv6 route table shows redistributed RIP and connected addresses.

```
Aaron#show ipv6 route isis
IPv6 Routing Table - 14 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
I2  2001:DB8:0:21::/64 [115/10]
     via FE80::204:C1FF:FE50:E700, Ethernet0
I2  2001:DB8:0:22::/64 [115/10]
     via FE80::204:C1FF:FE50:E700, Ethernet0
Aaron#
```

Because the RIP routes are external to the IS-IS routing domain, it is best to reflect this by having them redistributed into the domain as external routes, as in Example 11-46.

## Example 11-46. Mays configures the routes redistributed from RIP to IS-IS to external routes.

```
router isis
 redistribute rip metric 0 metric-type external level-2
 net 01.0001.0000.0c76.5432.00
 address-family ipv6
  redistribute rip baseball metric 0 metric-type external level-2
  redistribute connected metric 0 metric-type external level-2
!
router rip
 redistribute isis level-1-2 metric 1
 passive-interface Ethernet0
 network 10.0.0.0
!
ipv6 router rip baseball
 redistribute isis level-1-2 metric 1
 redistribute connected metric 1
```

Example 11-47 shows Aaron's IPv4 route table after the change. The only change from Example 11-42 is that the metrics of the redistributed routes have increased to greater than 64, indicating (in this small network) external routes. Example 11-48 displays the IS-IS database entry for Mays. The entries for the redistributed routes are shown as IP-External and IPv6-Ext.

## Example 11-47. The metrics of the routes to 10.2.1.0/24 and 10.2.2.0/24 change to 138 after the routes are advertised as external.

```
Aaron#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B  BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
       10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
C         10.1.3.0/24 is directly connected, Ethernet4
i L2      10.2.1.0/24 [115/74] via 10.1.4.2, Ethernet0
i L2      10.2.2.0/24 [115/74] via 10.1.4.2, Ethernet0
C         10.1.1.0/24 is directly connected, Ethernet1
C         10.1.4.0/24 is directly connected, Ethernet0
C         10.1.2.160/28 is directly connected, Ethernet2
C         10.1.2.224/28 is directly connected, Ethernet3
Aaron#
```

## Example 11-48. Redistributed routes are displayed as IP-External and IPv6-Ext in Mays's IS-IS database.

```
Mays# show isis database detail level-2 Mays.00-00
IS-IS Level-2 LSP Mays.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime     ATT/P/OL
Mays.00-00          * 0x0000005F   0xA0E5        1186             0/0/0
  Area Address: 01.0001
  NLPID:        0xCC 0x8E
  Hostname:     Mays
  IP Address:   10.1.4.2
  IPv6 Address: 2001:DB8:0:14::2
  Metric:10          IS Aaron.00
  Metric:20          IP 10.1.3.0 255.255.255.0
  Metric:64          IP-External 10.2.1.0 255.255.255.0
  Metric:64          IP-External 10.2.2.0 255.255.255.0
  Metric:20          IP 10.1.1.0 255.255.255.0
  Metric:10          IP 10.1.4.0 255.255.255.0
  Metric:20          IP 10.1.2.160 255.255.255.240
  Metric:20          IP 10.1.2.224 255.255.255.240
  Metric:20          IPv6 2001:DB8:0:11::/64
  Metric:20          IPv6 2001:DB8:0:13::/64
  Metric:10          IPv6 2001:DB8:0:14::/64
  Metric:0           IPv6-Ext 2001:DB8:0:21::/64
  Metric:0           IPv6-Ext 2001:DB8:0:22::/64
  Metric:20          IPv6 2001:DB8:0:12A::/64
  Metric:20          IPv6 2001:DB8:0:12B::/64
Mays#
```

Another look at Figure 11-12 shows that both subnets of the RIP domain, IPv4 and IPv6, can be summarized with a single address of 10.2.0.0/16 and 2001:db8:0:20::/62. IPv4 route summarization into IS-IS is configured with the same summary-address command that is used with OSPF. However, the level into which the summary is being sent can also be specified. If the level is not specified, the addresses will be summarized into level 2. As with OSPFv3 for IPv6, IPv6 routes are summarized into IS-IS using the summary-prefix command for the IPv6 address-family. In the configuration in Example 11-49, RIP IPv4 routes are redistributed as level 1 and summarized, and RIPng IPv6 routes are redistributed as level 2 and summarized.

## Example 11-49. Mays summarizes the RIP routes after they have been redistributed into IS-IS.

```
router isis
 summary-address 10.2.0.0 255.255.0.0 level-1
 redistribute rip metric 0 metric-type external level-1
 net 01.0001.0000.0c76.5432.00
 address-family ipv6
  redistribute rip baseball metric 0 metric-type external level-2
  redistribute connected metric 0 metric-type external level-2
  summary-prefix 2001:db8:0:20::/62 level-2
router rip
 redistribute isis level-1-2 metric 1
 passive-interface Ethernet0
 network 10.0.0.0
!
ipv6 router rip baseball
 redistribute isis level-1-2 metric 1
 redistribute connected metric 1
```

Example 11-50 shows the summary route in Aaron's IPv4 route table. Example 11-51 shows Aaron's IPv6 route table. Like OSPF and EIGRP, the summarization causes more-specific routes in the summary range to be suppressed.

## Example 11-50. Aaron's IPv4 route table with a level-1 summary route to the subnets within the RIP domain.

```
Aaron#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
     10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks
i L1    10.2.0.0/16 [115/138] via 10.1.4.2, Ethernet0
C       10.1.3.0/24 is directly connected, Ethernet4
C       10.1.1.0/24 is directly connected, Ethernet1
C       10.1.4.0/24 is directly connected, Ethernet0
C       10.1.2.160/28 is directly connected, Ethernet2
C       10.1.2.224/28 is directly connected, Ethernet3
Aaron#
```

## Example 11-51. Aaron's IPv6 route table with a level-2 summary route to the subnets within the RIP domain.

```
Aaron#show ipv6 route isis
IPv6 Routing Table - 13 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
I2  2001:DB8:0:20::/62 [115/10]
     via FE80::204:C1FF:FE50:E700, Ethernet0
```

```
Aaron#
```

When IS-IS is redistributed into another protocol, the route level to be redistributed must be specified. In the examples shown so far, both level 1 and level 2 routes have been specified to be redistributed into RIP.

## Case Study: Redistributing Static Routes

Example 11-52 shows the route table of Williams in Figure 11-12. Notice that subnets 10.1.2.160/28 and 10.1.2.224/28 are missing; their subnet masks are inconsistent with the 24-bit mask configured on Mays's E1 interface, so the routes have not been included in the RIP updates sent out that interface. This scenario again illustrates the problem of redistributing variably subnetted routes from a classless protocol into a classful protocol, as discussed earlier in this chapter.

### Example 11-52. The routes with subnets other than /24 are not redistributed into the RIP domain.

```
Williams#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
     10.0.0.0/8 is subnetted, 5 subnets
R       10.1.3.0 [120/1] via 10.2.1.2, 00:00:01, Ethernet0
C       10.2.1.0 is directly connected, Ethernet0
R       10.1.1.0 [120/1] via 10.2.1.2, 00:00:02, Ethernet0
C       10.2.2.0 is directly connected, Ethernet1
R       10.1.4.0 [120/1] via 10.2.1.2, 00:00:02, Ethernet0
Williams#
```

A solution to this problem is to summarize the two 28-bit subnets with a single 24-bit address of 10.1.2.0/24. RIP does not have a summarization command, so the way to accomplish summarization is to configure a static route to the summary address and then redistribute that route into RIP, as displayed in Example 11-53.

### Example 11-53. Mays is configured with a static route to create a summarized route for IPv4 subnets. The static route is redistributed into RIP.

```
router isis
 summary-address 10.2.0.0 255.255.0.0 level-1
 redistribute rip metric 0 metric-type external level-1
 net 01.0001.0000.0c76.5432.00
!
router rip
 redistribute static metric 1
 redistribute isis level-1-2 metric 1
 passive-interface Ethernet0
 network 10.0.0.0
!
ip route 10.1.2.0 255.255.255.0 10.1.4.1
```

Example 11-54 shows Williams's route table with the summary route included.

## Example 11-54. Subnets 10.1.2.160/28 and 10.1.2.224/28 have been summarized with the address 10.1.2.0/24.

```
Williams#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
   10.0.0.0/8 is subnetted, 6 subnets
R    10.1.3.0 [120/1] via 10.2.1.2, 00:00:03, Ethernet0
R    10.1.2.0 [120/1] via 10.2.1.2, 00:00:03, Ethernet0
C    10.2.1.0 is directly connected, Ethernet0
R    10.1.1.0 [120/1] via 10.2.1.2, 00:00:03, Ethernet0
C    10.2.2.0 is directly connected, Ethernet1
R    10.1.4.0 [120/1] via 10.2.1.2, 00:00:03, Ethernet0
Williams#
```

As discussed in Chapter 3, a variant of the static route is an entry that points to an outgoing interface instead of a next-hop address. These static routes can also be redistributed, but the configuration is somewhat different. For example, the configuration of Mays can be as displayed in Example 11-55.

## Example 11-55. Mays is configured with a static route pointing out an interface rather then to a next-hop address.

```
router isis
 summary-address 10.2.0.0 255.255.0.0 level-1
 redistribute rip metric 0 metric-type external level-1
 net 01.0001.0000.0c76.5432.00
!
router rip
 redistribute isis level-1-2 metric 1
 passive-interface Ethernet0
 network 10.0.0.0
!
ip route 10.1.2.0 255.255.255.0 Ethernet0
```

Here the static route now points to Mays's E0 interface instead of to the next-hop address 10.1.4.1. Also, the `redistribute static` command is no longer used under the RIP configuration, yet Williams's route table still looks the same as in Example 11-54.

The reason this static route still gets redistributed is that when a static route points to an outgoing interface, the destination is considered by the router to be directly connected (Example 11-56). And because a network statement for 10.0.0.0 appears under the RIP configuration, RIP will advertise this "directly connected" subnet of 10.0.0.0.

## Example 11-56. Mays considers the summary address 10.1.2.0/24 to be directly connected to Ethernet 0.

```
Mays#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
     10.0.0.0/8 is variably subnetted, 8 subnets, 2 masks
i L1    10.1.3.0/24 [115/20] via 10.1.4.1, Ethernet0
S       10.1.2.0/24 is directly connected, Ethernet0
C       10.2.1.0/24 is directly connected, Ethernet1
i L1    10.1.1.0/24 [115/20] via 10.1.4.1, Ethernet0
R       10.2.2.0/24 [120/1] via 10.2.1.1, 00:00:21, Ethernet1
C       10.1.4.0/24 is directly connected, Ethernet0
i       L1 10.1.2.160/28 [115/20] via 10.1.4.1, Ethernet0
i       L1 10.1.2.224/28 [115/20] via 10.1.4.1, Ethernet0
Mays#
```

Suppose Williams receives a packet with a destination address of 10.1.2.5. The summary address of 10.1.2.0/24 will be matched, and the packet will be forwarded to Mays. At Mays, the destination address does not match a more-specific subnet, and therefore will match the static route. Mays will send ARP requests out E0 in an attempt to find host 10.1.2.5 (or a router that will send a Proxy ARP reply). Finding none, the router does not know what to do with the packet. An ICMP Destination Unreachable message will not be sent to the originator.

Recall that when the summarization commands are used, they create an entry in the route table to Null 0.

### Note

Null interfaces should be configured in conjunction with static summary routes.

The same can and should be done with static summary routes (see Example 11-57).

## Example 11-57. Mays's static summary route now points to the null interface.

```
router isis
 summary-address 10.2.0.0 255.255.0.0 level-1
 redistribute rip metric 0 metric-type external level-1
 net 01.0001.0000.0c76.5432.00
!
router rip
 redistribute isis level-1-2 metric 1
 passive-interface Ethernet0
 network 10.0.0.0
!
ip route 10.1.2.0 255.255.255.0 Null0
```

Now any destination address that doesn't find a more-specific match at Mays will be routed to the null interface and dropped, and an ICMP Destination Unreachable message will be sent to the

originator.

PREV NEXT

# Looking Ahead

This chapter touches on several problems that can arise when redistributing routes. Avoiding or correcting trouble in all but the simplest redistribution schemes usually involves the use of route filters, which are discussed in Chapter 13, or route maps, discussed in Chapter 14. Those chapters include examples of more complex redistribution schemes and how to troubleshoot them. First, though, Chapter 12 examines default routeswhich may be thought of as the most generic form of summary routes.

# Summary Table: Chapter 11 Command Review

| Command | Description |
|---|---|
| default-metric *bandwidth delay reliability load mtu* | Specifies a default metric to be associated with routes redistributed into IGRP and EIGRP |
| default-metric *number* | Specifies a default metric to be associated with routes redistributed into RIP and OSPF |
| ip summary-address eigrp *autonomous-system-number address mask* | Configures an EIGRP summary route on an interface |
| redistribute connected | Redistributes all directly connected networks |
| redistribute *protocol* [*process-id*] {level-1\| level-1-2\| level-2} [metric *metric-value*] [metric-type *type-value*] [match{internal\| external 1\| external 2}] [tag *tag-value*] [route-map *map-tag*] [weight *weight*] [subnets] | Configures redistribution into a routing protocol and specifies the source of the redistributed routes |
| summary-address *address mask* {level-1\| level-1-2\| level-2} *prefix mask* [not-advertise] [tag *tag*] | Configures route summarization for IS-IS and OSPF |
| summary-prefix *prefix/length* {level-1\| level-1-2\| level-2} | Configures route summarization for IS-IS for IPv6 |
| summary-prefix *prefix/length*[not-advertise\| tag *tag-value*] | Configures route summarization for OSPFv3 for IPv6 |

# Review Questions

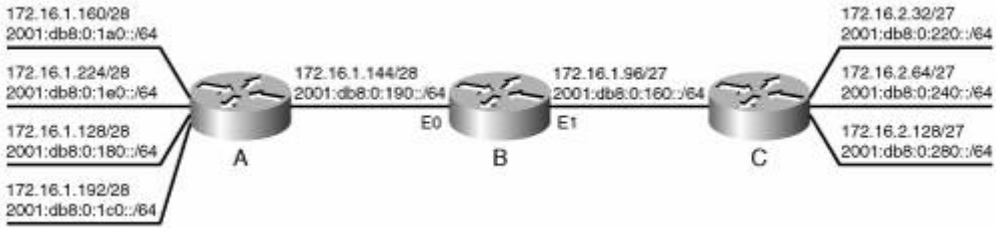
1. From what sources can a route be redistributed?
2. What is the purpose of an administrative distance?
3. How can administrative distances cause problems when redistributing?
4. How can redistribution from a classless to a classful routing protocol cause problems?
5. Which IP IGPs can use the default redistribution metric, and which IGPs must have a metric configured for redistribution to work?
6. What is the difference between using the `metric` keyword with the `redistribute` command and using the `default-metric` command?
7. What is the purpose of the `subnets` keyword when redistributing OSPF?
8. How is the null interface useful when summarizing routes?

# Configuration Exercises

1 Router A in Figure 11-13 is running OSPFv2 and v3, and router C is running RIPv1 and RIPng. Write a configuration for router B that will provide full connectivity for all subnets.

Figure 11-13. The network for Configuration Exercises 1, 2, and 3.

[View full size image]




2 Configure IPv6 summarization on router B in Figure 11-13 wherever possible.

3 Router A in Figure 11-13 is running IS-IS for both IPv4 and IPv6 and router C is running EIGRP for IPv4, and RIPng for IPv6. At router B, all the IS-IS routes are level 1. Configure mutual redistribution at router B by using summarization wherever possible. The EIGRP routes should be advertised into the IS-IS domain as external routes.

# Troubleshooting Exercises

1. In the case study "Redistributing IGRP and RIP," the following configuration is given for router Mantle of Figure 11-8:

```
router rip
 redistribute igrp 1 metric 5
 passive-interface Ethernet1
 network 10.0.0.0
!
router igrp 1
 redistribute connected
 redistribute rip
 default-metric 1000 100 255 1 1500
 passive-interface Ethernet0
 network 10.0.0.0
```

   Will the RIP domain know about the stub network 192.168.10.0/24 by virtue of the fact that it is redistributed into IGRP, which is then redistributed into RIP?

2. In Troubleshooting Exercise 1, if the redistribute rip command is eliminated under the IGRP configuration, will the redistribute connected command be sufficient to advertise the RIP domain?

3. In Example 11-21, why isn't subnet 192.168.3.32/27 tagged as an EIGRP external route?

4. In Example 11-21, there is a summary route to 192.168.3.0. What caused this entry?

5. Given the following configuration, will a level-1 router in area 1 know about the summary route?

   Router A:

```
router isis
 net 01.0001.0000.0c76.5432.00
 address-family ipv6
  redistribute rip baseball metric-type external
  summary-prefix 2001:db8:0:20::/62 level-1
```

# Chapter 12. Default Routes and On-Demand Routing

This chapter covers the following subjects:

- Fundamentals of Default Routes
- Fundamentals of On-Demand Routing
- Configuring Default Routes and ODR

Summarization has been examined in several chapters so far. *Summarization* conserves network resources by reducing the size of route tables and route advertisements. The smaller, simpler route tables can also make management and troubleshooting easier.

A *summary address* is an address that represents several, sometimes many, more-specific addresses. For example, the following four subnets

192.168.200.128/27
192.168.200.160/27
192.168.200.192/27
192.168.200.224/27

can be summarized with the single address 192.168.200.128/25.

When examined in binary, the addresses reveal that the summary address is less specific because it consists of fewer network and subnet bits than the addresses being summarized. So put crudely, it might be said that as more zeros are added to the host space and as fewer network bits are used, more addresses are summarized. Taking this concept to its limit, what if so many zeros are added to the host space that no network bits remain? In other words, what if the summary address consists of 32 zeros and has a prefix length of 0 (0.0.0.0/0)? This address summarizes every possible IPv4 address.

0.0.0.0/0 is the IPv4 default address, and a route to 0.0.0.0/0 is a default route.[1] Similarly, the default IPv6 address ::/0 summarizes every possible IPv6 address. Every other IP address is more specific than the default address, so when a default route exists in a route table, that route will be matched only if a more specific match cannot be made.

[1] This address is used by all the open IP routing protocols. The Cisco IGRP and EIGRP use an actual network address, advertised as an external route.

# Fundamentals of Default Routes

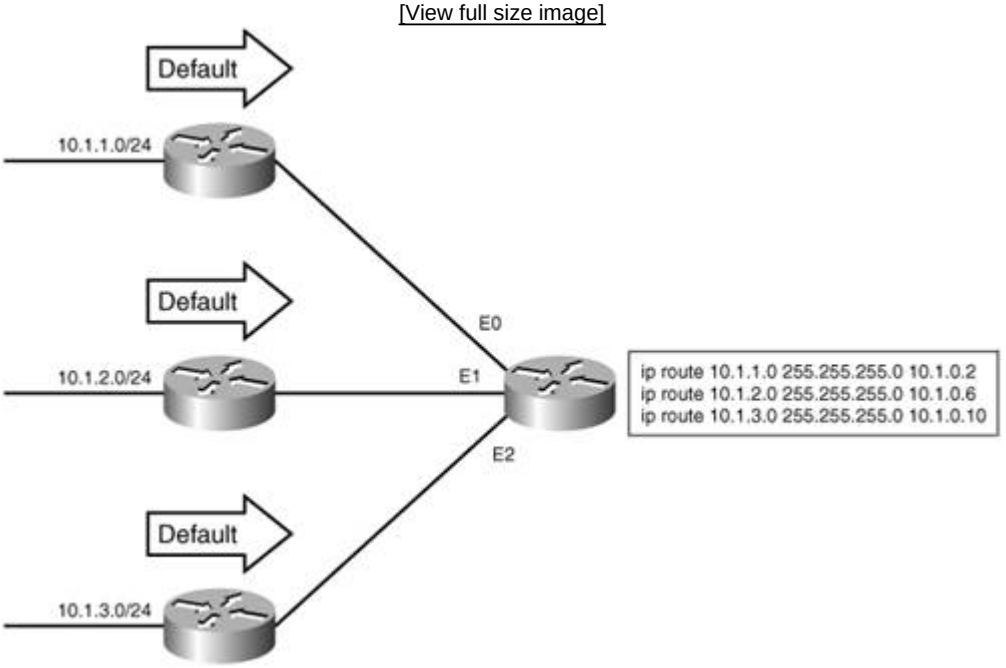
When a router is connected to the Internet, a default route is immensely useful. With a default route, the router needs to only recognize destinations that are internal to its own administrative system. The default route will forward packets destined for any other address to the Internet service provider. This negates the necessity of running Border Gateway Protocol (BGP) with the service provider to learn all of the prefixes in the Internet route tablea table which consists of well over 100,000 prefixes, and might soon be approaching 200,000. In dealing with large route tables, topology changes are an even bigger concern than the demands on memory. In a large network, topology changes will occur more frequently, resulting in increased system activity to advertise and process those changes. Using a default route effectively "hides" the changes of more-specific routes, making the network to which the default points appear more stable from the point of view of the router using the default route.

Default routes are also useful on a smaller scale, within single autonomous systems. The same benefits of decreased memory and processor utilization can be gained in smaller networks, although the benefits decrease as the number of routes decreases.

Default routes are also very useful in hub-and-spoke topologies, such as the one in Figure 12-1. Here, the hub router has a static route to every remote subnet. Entering new static routes in the hub router when a new subnet is brought online is a fairly trivial administrative task, but adding the routes to every spoke router might be much more time-consuming. By using default routes at the spoke routers, only the hub needs entries for every subnet. When a spoke router receives a packet for an unknown destination, it will forward the packet to the hub, which can, in turn, forward the packet to the correct destination.

Figure 12-1. Default routes greatly simplify the administration of static routing in a hub-and-spoke network.

[View full size image]




The spoke routers in Figure 12-1 are more correctly called "stub" routers. A *stub router* has only

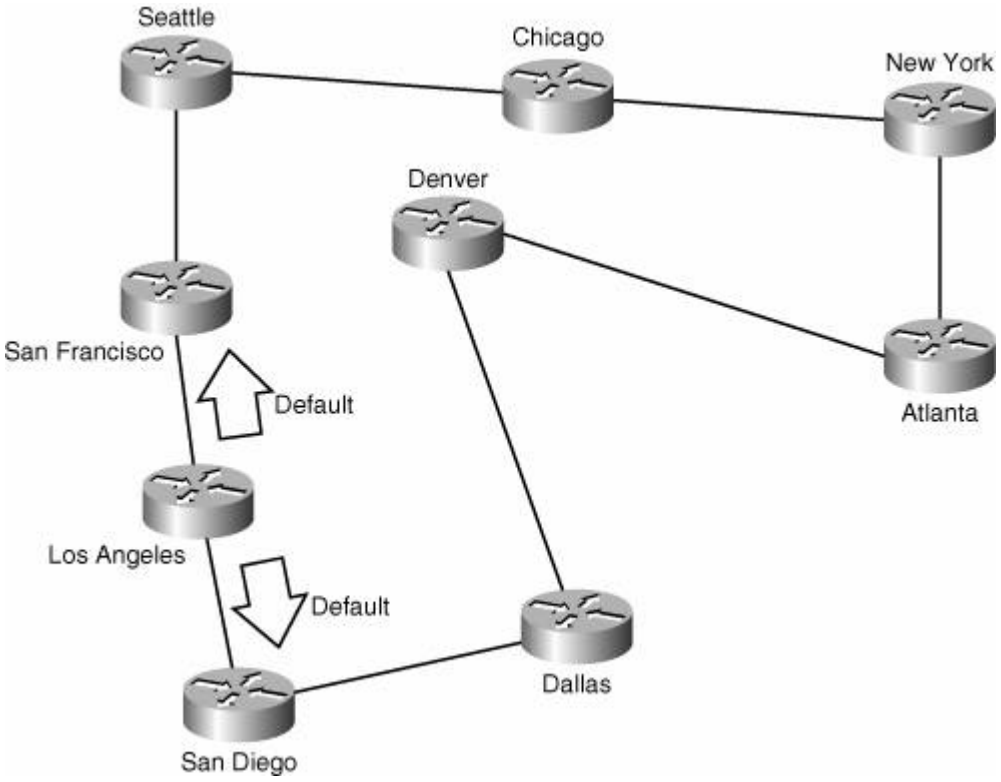
a single connection to another router. The routing decisions become very simple in such a device: The destination is either one of the router's directly connected networks (*stub networks*), or it is reachable via its single neighbor. And if the single neighbor is the only next-hop routing choice, the stub router has little need for a detailed route table. A default route is usually sufficient.

As with other summary routes, the trade-off with default routes is a loss of routing detail. The stub routers in Figure 12-1, for instance, have no way of knowing whether a destination is unreachable. All packets to unknown destinations are forwarded to the hub router, and only then is reachability determined. Packets to nonexistent addresses should be infrequent in a network. If for some reason they are not, a better design choice might be to allow the stub routers to run a routing protocol and learn routes from the hub so that unknown destinations can be determined as soon as possible. The design choice for you to make in a network such as the one in Figure 12-1 is whether it is more economical to forward packets with unknown destinations to the hub router, which can then drop them, or whether it is more economical to run a dynamic routing protocol between the hub and stub routers just to drop packets to unknown destinations at the stub routers. Although the resource and operational costs of running a dynamic routing protocol are usually small, the default route is still more likely to be the best choice.

Another problem with loss of routing detail is shown in Figure 12-2. These routers form a nationwide backbone, and large local networks are connected to each of the backbone routers. The Los Angeles backbone router has default routes pointing to both San Francisco and San Diego. If Los Angeles must forward a packet to Seattle and has only the two default routes, it has no way of knowing that the best route is via San Francisco. Los Angeles might forward the packet to San Diego, in which case the packet will use a small portion of some very expensive bandwidth, and will incur some unnecessary propagation delay, before it belatedly reaches its destination. Using default routes on this backbone is a bad design decision,[2] but it illustrates how hiding route details with a default route can lead to suboptimal routing.

[2] Having each backbone router advertise only a default route into its local network, on the other hand, can be a very good design choice, limiting the size of the local route tables.

Figure 12-2. If the Los Angeles router knows only default routes pointing to San Francisco and San Diego and has no more specific details about the topology behind those two routers, it cannot route efficiently.


Seattle
Chicago
New York
Denver
San Francisco
Default
Atlanta
Los Angeles
Default
Dallas
San Diego

PREV NEXT

# Fundamentals of On-Demand Routing

Although the configuration of static routes is simple in a hub router such as the one in Figure 12-1, many network administrators still see static routes as administratively undesirable. The difficulty is not so much adding routes as new stub networks are brought online, as it is remembering to remove routes when stub networks or stub routers are taken offline. Beginning with IOS 11.2, Cisco offers a proprietary alternative for hub routers called *On-Demand Routing* (ODR).

With ODR, a hub router can automatically discover stub networks while the stub routers still use a default route to the hub. ODR conveys address prefixesthat is, only the network portion of the addressrather than the entire addressso VLSM is supported. And because only minimal route information is traversing the link between the stub and hub routers, bandwidth is conserved.

ODR is not a true routing protocol. It discovers information about stub networks but does not provide any routing information to the stub routers. The link information is conveyed by a data-link protocol and, therefore, does not go further than from the stub router to the hub router. However, as a case study will show, ODR-discovered routes can be redistributed into dynamic routing protocols.

Example 12-1 shows a route table containing ODR entries. The table shows that the administrative distance is 160; the metric of the routes is 1. Because ODR routes are always from a hub router to a stub router, the metric (hop count) will never be more than 1. The routes also show that VLSM is supported.

### Example 12-1. This route table shows several ODR entries.

```
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
    192.168.1.0/24 is variably subnetted, 3 subnets, 2 masks
o      192.168.1.40/30 [160/1] via 192.168.1.37, 00:00:27, Serial0
C      192.168.1.36/30 is directly connected, Serial0
C      192.168.1.192/27 is directly connected, Ethernet1
o   192.168.3.0/24 [160/1] via 192.168.1.37, 00:00:27, Serial0
    192.168.4.0/24 is variably subnetted, 2 subnets, 2 masks
o      192.168.4.48/29 [160/1] via 192.168.1.37, 00:00:27, Serial0
o      192.168.4.128/27 [160/1] via 192.168.1.37, 00:00:27, Serial0
Router#
```

The transport mechanism for ODR routes is *Cisco Discovery Protocol* (CDP), a proprietary data link protocol that gathers information about neighboring network devices.[3] Example 12-2 shows the type of information collected by CDP.

[3] CDP runs not only on routers but also on Cisco switches and access servers.

### Example 12-2. CDP collects information about neighboring Cisco

network devices.

```
Bumble#show cdp neighbors detail
-------------------------
Device ID: P8R1
Entry address(es):
  IP address: 10.131.223.226
Platform: Cisco 2620, Capabilities: Router
Interface: Serial0/0.708, Port ID (outgoing port): Serial0/0.807
Holdtime : 144 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-J1S3-M), Version 12.3(6), RELEASE SOFTWARE (fc3)
Copyright (c) 1986-2004 by Cisco Systems, Inc.
Compiled Wed 11-Feb-04 19:24 by kellythw

advertisement version: 2

-------------------------
Device ID: Blathers
Entry address(es):
  IP address: 192.168.3.2
Platform: cisco 2610, Capabilities: Router
Interface: Serial0/0.1, Port ID (outgoing port): Serial0/0.2
Holdtime : 122 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-J1S3-M), Version 12.3(10a), RELEASE SOFTWARE (fc2
)
Copyright (c) 1986-2004 by Cisco Systems, Inc.
Compiled Fri 22-Oct-04 20:43 by kellythw

advertisement version: 2

-------------------------

Bumble#
-------------------------
```

CDP runs on any media that supports the subnetwork access protocol (SNAP), which means that ODR also depends on SNAP support. Although CDP is enabled by default on all interfaces of all Cisco devices running IOS 10.3 and later, ODR support begins with IOS 11.2. The configuration case study will show that ODR is configured on the hub router only; however, the stub routers must run IOS 11.2 or later for the hub router to discover their attached networks.

PREV NEXT

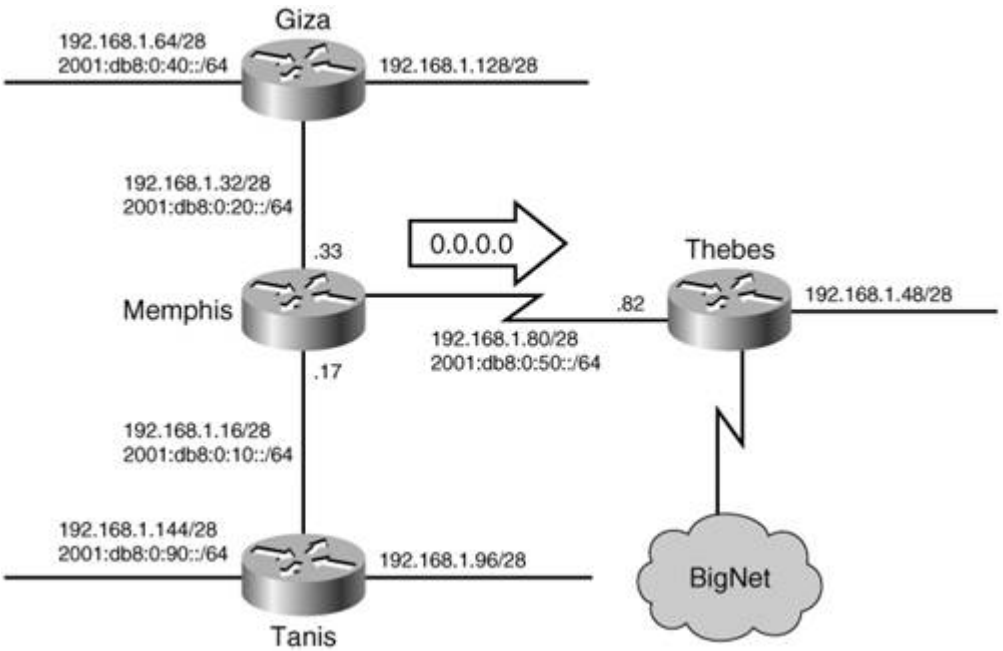
# Configuring Default Routes and ODR

Default routes can be configured either on each router that needs a default route or on one router that in turn advertises the routes to its peers. The case studies of this section examine both methods.

Recall from the discussion of classful route lookups in Chapter 5, "Routing Information Protocol (RIP)," that a router first matches a major network number and then matches the subnet. If a subnet cannot be matched, the packet will be dropped. Classless route lookup is the default behavior on Cisco routers as of IOS 11.3 and later; for earlier IOS versions, lookups can be changed to classless (even for classful routing protocols) with the global command ip classless.

Any router using a default route must perform classless route lookups. Figure 12-3 shows why. In this network, Memphis is speaking a dynamic routing protocol to Tanis and Giza, but is not receiving routes from Thebes. Memphis has a default route pointing to Thebes for routing packets to BigNet. If Memphis receives a packet with a destination address of 192.168.1.50 and is performing classful route lookups, it will first match major network 192.168.1.0, of which it has several subnets in its route table. Memphis will then attempt to find a route for subnet 192.168.1.48/28, but because Memphis is not receiving routes from Thebes, this subnet is not in its route table. The packet will be dropped.

Figure 12-3. Memphis forwards packets to Thebes with a default route. If Memphis uses classful route lookups, subnet 192.168.1.48/28 will be unreachable.




If Memphis is configured with ip classless, it will try to find the most specific match for 192.168.1.48/28 without matching the major network first. Finding no match for this subnet in the route table, it will match the default route and forward the packet to Thebes.

## Case Study: Static Default Routes

The configuration of Memphis in Figure 12-3 is displayed in Example 12-3.

### Example 12-3. Configuration of Router Memphis uses static IPv4 and IPv6 routes to create default routes.

```
interface serial 0/0.1
 ip address 192.168.1.33 255.255.255.240
 ipv6 address 2001:db8:0:20::1/64
 ipv6 rip egypt enable
!
interface serial 0/0.2
 ip address 192.168.1.81 255.255.255.240
 ipv6 address 2001:db8:0:50::1/64
 ipv6 rip egypt enable
!
interface serial 0/0.3
 ip address 192.168.1.17 255.255.255.240
 ipv6 address 2001:db8:0:10::1/64
 ipv6 rip egypt enable
!
router rip
network 192.168.1.0
!
ip classless
ip route 0.0.0.0 0.0.0.0 192.168.1.82
ipv6 route ::/0 2001:DB8:0:50::2
```

The static routes configure the default route addresses of 0.0.0.0 and ::/0 and use a mask that is also 0.0.0.0 (prefix length 0 for IPv6). A common mistake made by people configuring default routes for the first time is to use an all-ones mask instead of an all-zeros mask, such as the following:

```
ip route 0.0.0.0 255.255.255.255 192.168.1.82
```

An all-ones mask would configure a host route to 0.0.0.0, and the only packets that would match this address would be those with a destination address of 0.0.0.0. The all-zeros mask, on the other hand, is a mask made up entirely of "don't care" bits and will match any bit in any position. The beginning of this chapter described the default address as a summary route taken to its extreme so that every bit is summarized with a zero. The mask of the default route is a summary mask taken to its extreme.

Memphis' default route has a next-hop address at Thebes. This next-hop address is the *gateway of last resort*, or the default router. Example 12-4 shows the IPv4 route table at Memphis. The route to 0.0.0.0 is tagged as a candidate default, and the gateway of last resort is indicated at the top of the table. Example 12-5 shows the IPv6 route table.

### Example 12-4. Memphis' IPv4 route table, showing the default route and the gateway of last resort.

```
Memphis#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is 192.168.1.82 to network 0.0.0.0
    192.168.1.0/28 is subnetted, 7 subnets
R      192.168.1.96 [120/1] via 192.168.1.18, 00:00:15, Ethernet0
R      192.168.1.64 [120/1] via 192.168.1.34, 00:00:27, Ethernet1
C      192.168.1.80 is directly connected, Serial0
C      192.168.1.32 is directly connected, Ethernet1
C      192.168.1.16 is directly connected, Ethernet0
R      192.168.1.128 [120/1] via 192.168.1.34, 00:00:27, Ethernet1
R      192.168.1.144 [120/1] via 192.168.1.18, 00:00:15, Ethernet0
S*  0.0.0.0/0 [1/0] via 192.168.1.82
Memphis#
```

Example 12-5. Memphis' IPv6 route table shows the static entry for the default address ::/0.

```
Memphis#show ipv6 route
IPv6 Routing Table - 11 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
S   ::/0 [1/0]
     via 2001:DB8:0:50::2

C   2001:DB8:0:10::/64 [0/0]
     via ::, Serial0/0.3
L   2001:DB8:0:10::1/128 [0/0]
     via ::, Serial0/0.3
C   2001:DB8:0:20::/64 [0/0]
     via ::, Serial0/0.1
L   2001:DB8:0:20::1/128 [0/0]
     via ::, Serial0/0.1
R   2001:DB8:0:40::/64 [120/2]
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.1
C   2001:DB8:0:50::/64 [0/0]
     via ::, Serial0/0.2
L   2001:DB8:0:50::1/128 [0/0]
     via ::, Serial0/0.2

R   2001:DB8:0:90::/64 [120/2]
     via FE80::205:5EFF:FE6B:50A0, Serial0/0.3
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Memphis#
```

The default route now needs to be advertised to the rest of the RIP routers. This is done by redistributing the static route into RIP. Memphis will not advertise the default route to Tanis and Giza unless the static route is redistributed into the RIP protocol.[4] Example 12-6 shows that a redistribution command is added for both IPv4 and IPv6 on the Memphis router.

[4] Before IOS train 12.0T, if a default route was known in the route table, RIP, IGRP, and EIGRP would automatically advertise it to neighbors, without the need to redistribute the static route into the routing protocol.

## Example 12-6. Redistribution commands have been added to Memphis to enable the static default routes to be advertised by RIP.

```
router rip
redistribute static
!
ipv6 router rip egypt
redistribute static
```

OSPF and IS-IS do not use the redistribute command to advertise a default route but can still originate default routes, as shown in a subsequent case study. Example 12-7 and Example 12-8 show the IPv4 and IPv6 route tables of Tanis after the static default routes are redistributed into RIP.

## Example 12-7. The IPv4 route table of Tanis shows that the default route has been learned from Memphis via RIP.

```
Tanis#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is 192.168.1.17 to network 0.0.0.0
     192.168.1.0/28 is subnetted, 9 subnets
C       192.168.1.96 is directly connected, Ethernet1
R       192.168.1.64 [120/2] via 192.168.1.17, 00:00:01, Ethernet0
R       192.168.1.80 [120/1] via 192.168.1.17, 00:00:01, Ethernet0
R       192.168.1.32 [120/1] via 192.168.1.17, 00:00:01, Ethernet0
R       192.168.1.48 [120/2] via 192.168.1.17, 00:00:01, Ethernet0
C       192.168.1.16 is directly connected, Ethernet0
R       192.168.1.224 [120/1] via 192.168.1.17, 00:00:01, Ethernet0
R       192.168.1.128 [120/2] via 192.168.1.17, 00:00:01, Ethernet0
C       192.168.1.144 is directly connected, Ethernet2
R*    0.0.0.0/0 [120/1] via 192.168.1.17, 00:00:02, Ethernet0
Tanis#
```

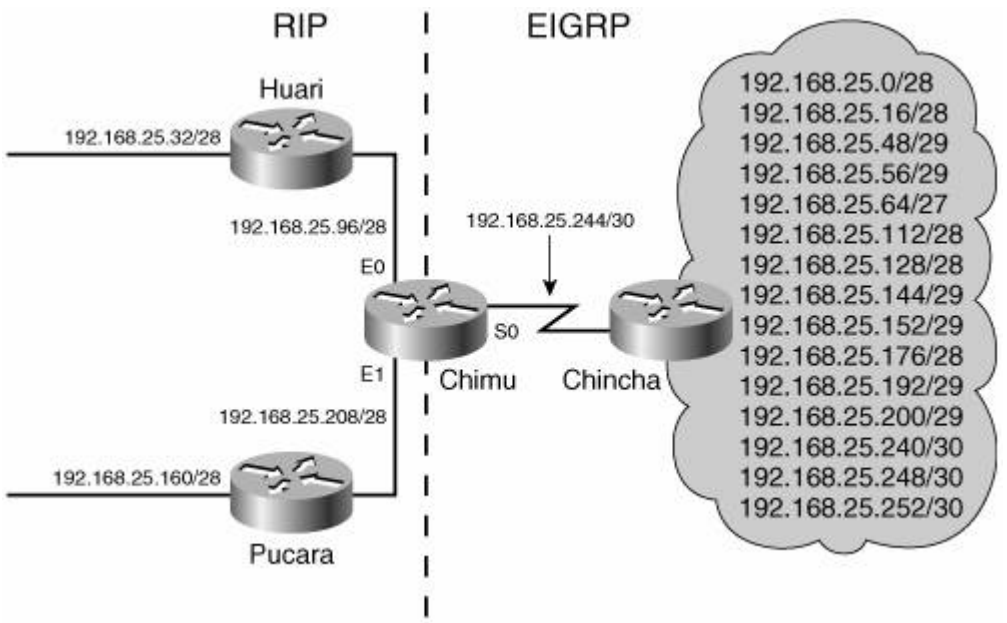
## Example 12-8. The IPv6 route table of Tanis shows that the default route has been learned from Memphis via IPv6 RIP.

```
       Tanis#show ipv6 route
IPv6 Routing Table - 10 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
R   ::/0 [120/2]
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
C   2001:DB8:0:10::/64 [0/0]
     via ::, Serial0/0.1
L   2001:DB8:0:10::2/128 [0/0]
     via ::, Serial0/0.1
R   2001:DB8:0:20::/64 [120/2]
```

```
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
R   2001:DB8:0:40::/64 [120/3]
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
R   2001:DB8:0:50::/64 [120/2]
     via FE80::204:C1FF:FE50:E700, Serial0/0.1
C   2001:DB8:0:90::/64 [0/0]
     via ::, FastEthernet0/0
L   2001:DB8:0:90::1/128 [0/0]
     via ::, FastEthernet0/0
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Tanis#
```

Default routes are also useful for connecting classless routing domains. In Figure 12-4, Chimu is connecting a RIP domain with an EIGRP domain. Although the masks of major network 192.168.25.0 are consistent in the RIP domain, they are variably subnetted in the EIGRP domain. Further, the VLSM scheme does not lend itself to summarization into RIP.

Figure 12-4. A default route enables RIP to route into the variably subnetted EIGRP domain.




Chimu's configuration is displayed in Example 12-9.

Example 12-9. RIP routes are redistributed into EIGRP by Chimu, but a default route, rather than all the EIGRP routes, is advertised into the RIP domain.

```
router eigrp 1
redistribute rip metric 1000 100 255 1 1500
passive-interface Ethernet0
passive-interface Ethernet1
```

```
network 192.168.25.0

!
router rip
passive-interface Serial0
network 192.168.25.0
redistribute static
!
ip classless
ip route 0.0.0.0 0.0.0.0 Null0
```

Chimu has a full set of routes from the EIGRP domain but is not redistributing them into RIP. Instead, Chimu is advertising a default route. The RIP routers will forward packets with unknown destinations to Chimu, which can then consult its route table for a more-specific route into the EIGRP domain.

Chimu's static route is pointing to the null interface rather than a next-hop address. If a packet is forwarded to Chimu with a destination on a nonexistent subnet, such as 192.168.25.224/28, the packet will be dropped instead of being forwarded into the EIGRP domain.

## Case Study: The Default-Network Command

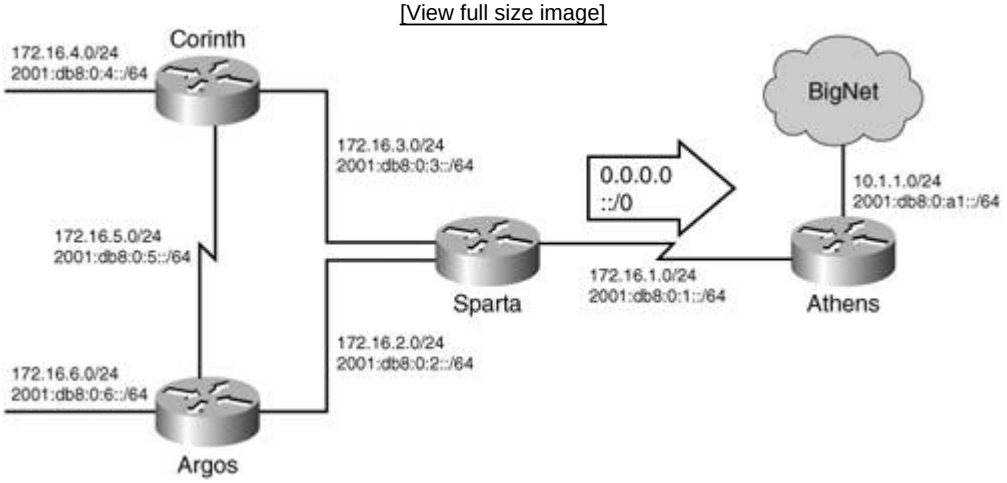
An alternative method of configuring default routes is to use the command ip default-network. This command specifies a network address to be used as a default network. The network might be directly connected to the router, specified by a static route, or discovered by a dynamic routing protocol. The command was first introduced for use with IGRP, which doesn't identify 0.0.0.0 as a default route, so an existing network was flagged as the default instead. Only IGRP, EIGRP, and RIP use this command.

The ip default-network command is a global command and causes any routing protocol that is configured on the router that supports the command to advertise a default route. The default route will be the network specified as an argument to the command if IGRP or EIGRP is used, and it will be 0.0.0.0 with RIP.

The ip default-network command is used with RIP in the configuration of Athens in Figure 12-5. Athens configuration is displayed in Example 12-10.

Figure 12-5. The *default-network* command is used at Athens to generate a default network advertisement.

[View full size image]



## Example 12-10. The *default-network* command can be used by RIP to create a default route.

```
router rip
network 172.16.0.0
!
ip classless
ip default-network 10.0.0.0
```

Example 12-11 shows that network 10.0.0.0 has been tagged as a candidate default route in the Athens route table, but notice that no gateway of last resort is specified. The reason is that Athens is the gateway to the default network. The ip default-network command will cause Athens to advertise a default network, even though no network statement for 10.0.0.0 exists under the RIP configuration (Example 12-12). When using RIP, the ip default-network command configured on Athens causes Athens to advertise 0.0.0.0 as the default network, not the network specified by the ip default-network command.

## Example 12-11. Network 10.0.0.0 is tagged as a candidate default in Athens' route table.

```
Athens#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
* 10.0.0.0/8 is subnetted, 1 subnets
C 10.1.1.0 is directly connected, Ethernet0
  172.16.0.0/16 is subnetted, 6 subnets
R 172.16.4.0 [120/2] via 172.16.1.2, 00:00:12, Serial0
R 172.16.5.0 [120/2] via 172.16.1.2, 00:00:12, Serial0
R 172.16.6.0 [120/2] via 172.16.1.2, 00:00:12, Serial0
C 172.16.1.0 is directly connected, Serial0
R 172.16.2.0 [120/1] via 172.16.1.2, 00:00:12, Serial0
R 172.16.3.0 [120/1] via 172.16.1.2, 00:00:12, Serial0
Athens#
```

## Example 12-12. Sparta's route table shows that Athens is advertising a default route of 0.0.0.0 and that Athens is Sparta's gateway of last resort.

```
Sparta#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is 172.16.1.1 to network 0.0.0.0
  172.16.0.0/24 is subnetted, 6 subnets
R   172.16.4.0 [120/1] via 172.16.3.2, 00:00:14, Ethernet1
R   172.16.5.0 [120/1] via 172.16.3.2, 00:00:14, Ethernet1
```

```
R   172.16.6.0 [120/1] via 172.16.2.2, 00:00:10, Ethernet0
C   172.16.1.0 is directly connected, Serial0
C   172.16.2.0 is directly connected, Ethernet0
C   172.16.3.0 is directly connected, Ethernet1
R* 0.0.0.0/0 [120/1] via 172.16.1.1, 00:00:17, Serial0
Sparta#
```

As with RIP, EIGRP will advertise a default route to neighbors if the static route to 0.0.0.0 is configured, and EIGRP redistributes static routes. EIGRP advertises the redistributed route as an external route See Chapter 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)."

If the routers in Figure 12-5 are configured to run EIGRP using the ip default-network command, Athens' configuration will be as displayed in Example 12-13.

## Example 12-13. The *default-network* command can be used with EIGRP to flag a network as a candidate default route.

```
router eigrp 1
network 10.0.0.0
network 172.16.0.0
!
ip classless
ip default-network 10.0.0.0
```

The ip default-network command remains the same as with RIP, but notice that a network statement for 10.0.0.0 is added to the EIGRP configuration. Since EIGRP sends the actual network address as the default network, that address must be configured to be advertised, as shown in Example 12-14. Compare the route table in Example 12-12 with the table in Example 12-14. RIP flags the route to 0.0.0.0/0 as the default, while EIGRP flags the route to 10.0.0.0/8 as the default network. Because Corinth has learned about the default route from Sparta, that router is Corinth's gateway of last resort. If the link to Sparta fails, Corinth will use Argos as its gateway of last resort.

## Example 12-14. EIGRP uses an actual network address, rather than 0.0.0.0, as the default network. Corinth's route table shows that network 10.0.0.0 is tagged as the default network.

```
Corinth#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is 172.16.3.1 to network 10.0.0.0
D*      10.0.0.0/8 [90/2195456] via 172.16.3.1, 00:02:32, Ethernet0
   172.16.0.0/16 is subnetted, 6 subnets
C     172.16.4.0 is directly connected, Ethernet1
C     172.16.5.0 is directly connected, Serial0

D       172.16.1.0 [90/1811456] via 172.16.3.1, 00:00:17, Ethernet0
D       172.16.6.0 [90/921600] via 172.16.3.1, 00:00:16, Ethernet0
D       172.16.2.0 [90/793600] via 172.16.3.1, 00:00:16, Ethernet0
C     172.16.3.0 is directly connected, Ethernet0
```

Notice that in the configuration of Athens, the ip default-network command is a global

command. It is not associated with a particular routing protocol. Any routing protocol that is configured on the router that can use the ip default-network command will use it. If both RIP and EIGRP are configured on the router, both protocols will advertise a default route, as shown in Corinth's route table in Example 12-15.

### Example 12-15. Corinth's route table shows two candidate default routes when Athens is configured with both RIP and EIGRP and using the *ip default-network* command.

```
Corinth#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is 172.16.3.1 to network 10.0.0.0
D*      10.0.0.0/8 [90/2195456] via 172.16.3.1, 00:02:32, Ethernet0
   172.16.0.0/16 is subnetted, 6 subnets
C     172.16.4.0 is directly connected, Ethernet1
C     172.16.5.0 is directly connected, Serial0

D       172.16.1.0 [90/1811456] via 172.16.3.1, 00:00:17, Ethernet0
D       172.16.6.0 [90/921600] via 172.16.3.1, 00:00:16, Ethernet0
D       172.16.2.0 [90/793600] via 172.16.3.1, 00:00:16, Ethernet0
C     172.16.3.0 is directly connected, Ethernet0
R* 0.0.0.0/0 [120/1] via 172.16.3.1, 00:00:17, Serial0
```

The EIGRP-discovered default network becomes the gateway of last resort because EIGRP has a lower administrative distance.

There is an inherent lack of control in this method of advertising a default network. If multiple routing protocols are configured on the router, such as RIP and EIGRP, and the ip default-network command is used, there is no way to control or limit which routing protocol advertises the default network. If Athens, in Figure 12-5, is running EIGRP for BigNet, and RIP for the rest of the network, and the ip default-network command is configured with the intent of advertising a default route into RIP, Athens will also advertise a default into EIGRP. This will disrupt routing not only for traffic originating in the RIP network and attempting to route to BigNet, but also for traffic within BigNet.

When injecting routes into a routing protocol, it is always best to choose the method that offers the most control to minimize unintended route propagation.

## Case Study: The Default-Information Originate Command

An OSPF ASBR and an IS-IS interdomain router will not automatically advertise a default route into their routing domains, even when one exists. For example, suppose Athens in Figure 12-5 is configured for OSPF and given a static default route into BigNet. Example 12-16 shows Athens's configuration.

### Example 12-16. Athens now routes with OSPF and has a static default route.

```
router ospf 1
network 172.16.0.0 0.0.255.255 area 0
!
```

```
ip classless
ip route 0.0.0.0 0.0.0.0 10.1.1.2
```

Example 12-17 shows the route tables of Athens and Sparta. Although the static route has caused the gateway of last resort to be set at Athens, Sparta has no knowledge of the default route. The default route must be advertised into the OSPF domain in type 5 LSAs, which means that Athens must be an ASBR. Yet so far, nothing in Athens' configuration tells it to perform this function.

## Example 12-17. The OSPF process at Athens does not automatically advertise the default route into the OSPF domain.

```
Athens#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is 10.1.1.2 to network 0.0.0.0
    10.0.0.0 255.255.255.0 is subnetted, 1 subnets
C    10.1.1.0 is directly connected, Ethernet0
    172.16.0.0 is variably subnetted, 6 subnets, 2 masks
O    172.16.5.0 255.255.255.0 [110/138] via 172.16.1.2, 00:04:17, Serial0
O    172.16.4.1 255.255.255.0 [110/75] via 172.16.1.2, 00:04:17, Serial0
O    172.16.6.1 255.255.255.0 [110/75] via 172.16.1.2, 00:04:17, Serial0
C    172.16.1.0 255.255.255.0 is directly connected, Serial0
O    172.16.2.0 255.255.255.0 [110/74] via 172.16.1.2, 00:04:17, Serial0
O    172.16.3.0 255.255.255.0 [110/74] via 172.16.1.2, 00:04:17, Serial0
S* 0.0.0.0 0.0.0.0 [1/0] via 10.1.1.2
Sparta#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
   172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
O     172.16.5.0/24 [110/74] via 172.16.2.2, 00:06:00, Ethernet1
                    [110/74] via 172.16.3.2, 00:06:00, Ethernet0
O     172.16.4.1/24 [110/11] via 172.16.3.2, 00:06:00, Ethernet0
O     172.16.6.1/24 [110/11] via 172.16.2.2, 00:06:00, Ethernet1
C     172.16.1.0/24 is directly connected, Serial0
C     172.16.2.0/24 is directly connected, Ethernet1
C     172.16.3.0/24 is directly connected, Ethernet0
```

The default-information originate command is a specialized form of the redistribute command, causing a default route to be redistributed into OSPF or IS-IS. And like redistribute, the default-information originate command informs an OSPF router that it is an ASBR, or informs an IS-IS router that it is an interdomain router. Also like redistribute, the metric of the redistributed default can be specified, as can the OSPF external metric type and the IS-IS level. To redistribute the default route into the OSPF domain with a metric of 10 and an external metric type of E1, Athens's configuration will be as displayed in Example 12-18.

## Example 12-18. The *default-information originate* command is used to originate a default route at Athens.

```
router ospf 1
network 172.16.0.0 0.0.255.255 area 0
default-information originate metric 10 metric-type 1
!
ip classless
ip route 0.0.0.0 0.0.0.0 10.1.1.2
```

Example 12-19 shows that the default route is now being redistributed into OSPF. The route can also be observed in Sparta's OSPF database (Example 12-20).

### Example 12-19. After *default-information originate* is configured at Athens, the default route is redistributed into the OSPF domain.

```
Sparta#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is 172.16.1.1 to network 0.0.0.0
    172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
O      172.16.5.0/24 [110/74] via 172.16.2.2, 00:14:46, Ethernet0
O      172.16.4.1/32 [110/75] via 172.16.2.2, 00:14:46, Ethernet0
O      172.16.6.1/32 [110/11] via 172.16.2.2, 00:14:46, Ethernet0
C      172.16.1.0/24 is directly connected, Serial0
C      172.16.2.0/24 is directly connected, Ethernet0
C      172.16.3.0/24 is directly connected, Ethernet1
O* E1 0.0.0.0/0 [110/74] via 172.16.1.1, 00:02:55, Serial0
Sparta#
```

### Example 12-20. Like other external routes advertised by an ASBR, the default route is advertised in a type 5 LSA.

```
Sparta#show ip ospf database external
       OSPF Router with ID (172.16.3.1) (Process ID 1)
               Type-5 AS External Link States
Routing Bit Set on this LSA
LS age: 422
Options: (No TOS-capability, No DC)
LS Type: AS External Link
Link State ID: 0.0.0.0 (External Network Number )
Advertising Router: 172.16.1.1
LS Seq Number: 80000002
Checksum: 0x5238
Length: 36
Network Mask: /0
       Metric Type: 1 (Comparable directly to link state metric)
       TOS: 0
       Metric: 10
       Forward Address: 0.0.0.0
       External Route Tag: 1
Sparta#
```

The default-information originate command also will redistribute into OSPF or IS-IS a default

route that has been discovered by another routing process. In the configuration in Example 12-21, the static route to 0.0.0.0 has been eliminated, and Athens is speaking BGP to a router in BigNet.

## Example 12-21. Athens is configured to learn routes via BGP rather then statically.

```
router ospf 1
network 172.16.0.0 0.0.255.255 area 0
default-information originate metric 10 metric-type 1
!
router bgp 65501
network 172.16.0.0
neighbor 10.1.1.2 remote-as 65502
!
ip classless
```

Athens is now learning a route to 0.0.0.0 from its BGP neighbor and will advertise the route into the OSPF domain via type 5 LSAs (Example 12-22).

## Example 12-22. A BGP-speaking neighbor in BigNet is advertising a default route to Athens.

```
Athens#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is 10.1.1.2 to network 0.0.0.0
    10.0.0.0/8 is subnetted, 1 subnets
C      10.1.1.0 is directly connected, Ethernet0
    172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
O IA   172.16.4.1/32 [110/139] via 172.16.1.2, 00:16:45, Serial0
O IA   172.16.5.0/24 [110/138] via 172.16.1.2, 00:16:45, Serial0
O IA   172.16.6.1/32 [110/75] via 172.16.1.2, 00:16:45, Serial0
C      172.16.1.0/24 is directly connected, Serial0
O IA   172.16.2.0/24 [110/74] via 172.16.1.2, 00:16:45, Serial0
O IA   172.16.3.0/24 [110/74] via 172.16.1.2, 00:16:45, Serial0
B* 0.0.0.0/0 [20/0] via 10.1.1.2, 00:12:02
Athens#
```

A benefit of a default route, or any summary route, is that it can add stability to a network. But what if the default route itself is unstable? For example, suppose that the default route advertised to Athens in Example 12-19 is *flapping*, that is, alternating frequently between reachable and unreachable. With each change, Athens must send a new type 5 LSA into the OSPF domain. This LSA will be advertised into all nonstub areas. Although this flooding and reflooding might have minimal impact on system resources, it still might be undesirable to the network administrator. A solution is to use the always keyword.[5] Example 12-23 shows how Athens is configured to always originate a default route, even if the default route is not currently present in Athens's route table.

[5] This keyword is available only under OSPF. It is not supported under IS-IS.

## Example 12-23. Athens will always originate a default route, even if no

default route is currently present in the route table.

```
router ospf 1
network 172.16.0.0 0.0.255.255 area 0
default-information originate always metric 10 metric-type 1
!
router bgp 65501
network 172.16.0.0
neighbor 10.1.1.2 remote-as 65502
!
ip classless
```

With this configuration, Athens will always advertise a default route into the OSPF domain, regardless of whether it actually has a route to 0.0.0.0. If a router within the OSPF domain defaults a packet to Athens and Athens has no default route, it will send an ICMP Destination Unreachable message to the source address and drop the packet.

The always keyword can be used safely when there is only a single default route out of the OSPF domain. If more than one ASBR is advertising a default route, the defaults should be dynamicthat is, the loss of a default route should be advertised. If an ASBR claims to have a default when it doesn't, packets can be forwarded to it instead of to a legitimate ASBR.

The default-information originate works similarly for IPv6. In Figure 12-5, IPv6 is being routed via IS-IS. Athens is configured to originate a default route for IPv6.

Athens's configuration is shown in Example 12-24.

## Example 12-24. A default IPv6 route is originated by Athens for the IS-IS protocol.

```
ipv6 unicast-routing
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0
 ipv6 address 2001:DB8:0:A1::1/64
 ipv6 router isis
!
interface Serial0
 ip address 172.16.1.1 255.255.255.0
 ip router isis
 ipv6 address 2001:DB8:0:1::1/64
 ipv6 router isis
!
router isis
 net 01.0000.00ef.5678.00
 metric-style wide
 address-family ipv6
  multi-topology
  default-information originate
  exit-address-family
```

Athens does not require that the default route be learned from another source before entering the default route into its IS-IS database and advertising it to neighbors. All data destined to unknown IPv6 addresses is forwarded to Athens by the other routers. If Athens does not have a route to the destination in its route table, it will drop the packet. Example 12-25 shows the Argos IS-IS level-2 database entry for Athens. Example 12-26 shows the Argos IPv6 route table.

## Example 12-25. IPv6 default routes are added to the level-2 IS-IS database.

```
Argos#show isis database detail level-2 Athens.00-00
IS-IS Level-2 LSP Athens.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime     ATT/P/OL
Athens.00-00          0x00000088   0xBD29        956              0/0/0
  Area Address: 01
  Topology:     IPv4 (0x0) IPv6 (0x2)
  NLPID:        0xCC 0x8E
  Hostname: Athens
  IP Address:   172.16.1.1
  IPv6 Address: 2001:DB8:0:A1::1
  Metric: 10         IS-Extended Athens.01
  Metric: 10         IS (MT-IPv6) Athens.01
  Metric: 10         IP 172.16.1.0/24
  Metric: 0          IPv6 (MT-IPv6) ::/0
  Metric: 10         IPv6 (MT-IPv6) 2001:DB8:0:1::/64
  Metric: 20         IPv6 (MT-IPv6) 2001:DB8:0:2::/64
  Metric: 20         IPv6 (MT-IPv6) 2001:DB8:0:3::/64
  Metric: 30         IPv6 (MT-IPv6) 2001:DB8:0:4::/64
  Metric: 30         IPv6 (MT-IPv6) 2001:DB8:0:5::/64
  Metric: 30         IPv6 (MT-IPv6) 2001:DB8:0:6::/64
  Metric: 30         IPv6 (MT-IPv6) 2001:DB8:0:20::/64
  Metric: 10         IPv6 (MT-IPv6) 2001:DB8:0:A1::/64
Argos#
```

## Example 12-26. IPv6 default routes are added to the IPv6 route table as IS-IS level-2.

```
Argos#show ipv6 route
IPv6 Routing Table - 14 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
I2  ::/0 [115/20]
     via FE80::204:C1FF:FE50:E700, FastEthernet0/1
I1  2001:DB8:0:1::/64 [115/20]
     via FE80::204:C1FF:FE50:E700, FastEthernet0/1
C   2001:DB8:0:2::/64 [0/0]
     via ::, FastEthernet0/1
L   2001:DB8:0:2::2/128 [0/0]
     via ::, FastEthernet0/1
I1  2001:DB8:0:3::/64 [115/20]
     via FE80::204:C1FF:FE50:E700, FastEthernet0/1
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.2
I1  2001:DB8:0:4::/64 [115/20]
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.2
C   2001:DB8:0:5::/64 [0/0]
     via ::, Serial0/0.2
L   2001:DB8:0:5::1/128 [0/0]
     via ::, Serial0/0.2
C   2001:DB8:0:6::/64 [0/0]
     via ::, FastEthernet0/0
L   2001:DB8:0:6::1/128 [0/0]
     via ::, FastEthernet0/0
```

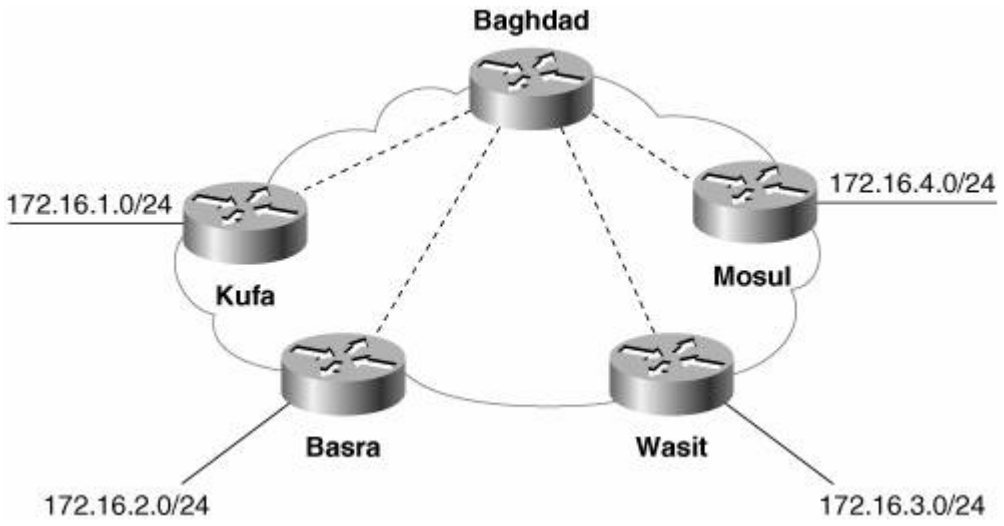
```
I1  2001:DB8:0:20::/64 [115/20]
     via FE80::204:C1FF:FE50:F1C0, Serial0/0.2

I1  2001:DB8:0:A1::/64 [115/30]
     via FE80::204:C1FF:FE50:E700, FastEthernet0/1
L   FE80::/10 [0/0]
     via ::, Null0
L   FF00::/8 [0/0]
     via ::, Null0
Argos#
```

## Case Study: Configuring On-Demand Routing

ODR is enabled with a single command, `router odr`. No networks or other parameters must be specified. CDP is enabled by default; it needs to be enabled only if it has been turned off for some reason. The command to enable the CDP process on a router is `cdp run`; to enable CDP on a specific interface, the command is `cdp enable`.

Figure 12-6 shows a typical hub-and-spoke topology. To configure ODR, the hub router will have the `router odr` command. If all routers are running IOS 11.2 or later and the connecting medium supports SNAP (such as the Frame Relay or PVCs shown), ODR is operational and the hub will learn the stub networks. The only configuration necessary at the stub routers is a static default route to the hub.

Figure 12-6. Hub-and-spoke topologies such as this one are common across Frame Relay networks.




ODR can also be redistributed. If Baghdad in Figure 12-6 needs to advertise the ODR-discovered routes into OSPF, Baghdad's configuration might be as displayed in Example 12-27.

Example 12-27. ODR discovered routes can be redistributed into other IP routing protocols.

```
router odr
!
router ospf 1
```

```
redistribute odr metric 100
network 172.16.0.0 0.0.255.255 area 5
```

PREV NEXT

# Looking Ahead

Configuration and troubleshooting of default routes are trivial tasks in the simple, loop-free networks shown in this chapter. When topologies are more complex, and especially when they include looping paths, the potential for problems with both default routing and with redistribution increases. Chapter 13, "Route Filtering," and Chapter 14, "Route Maps," discuss the tools that are vital to controlling routing behavior in complex topologies.

# Summary Table: Chapter 12 Command Review

| Command | Description |
|---|---|
| cdp enable | Enables CDP on an interface |
| cdp run | Enables CDP globally on a router |
| default-information originate [always] [metric *metric-value*] [metric-type *type-value*]{level-1 \| level-1-2 \| level-2} [route-map *map-name*] | Generates a default route into OSPF and IS-IS routing domains |
| ip classless | Enables classless route lookups so that the router can forward packets to unknown subnets of directly connected networks |
| ip default-network *network-number* | Specifies a network as a candidate route when determining the gateway of last resort |
| ip route *address mask* {*address* \| *interface*} [*distance*] [tag *tag*] [permanent] | Specifies a static route entry |
| router odr | Enables On-Demand Routing |

# Review Questions

1. What is the destination address of IPv4 default routes used by the open protocols?
2. What is the destination prefix/prefix length of IPv6 default routes?
3. How are default routes identified and advertised by EIGRP?
4. Can a static route to 0.0.0.0 be used as the default route on a router running EIGRP?
5. What is a stub router? What is a stub network?
6. What is an advantage of using default routes instead of a full route table?
7. What is an advantage of using a full route table instead of a default route?
8. What data link protocol does On-Demand Routing use to discover routes?
9. What IOS restrictions are placed on ODR?
10. What media restrictions are placed on ODR?

# Chapter 13. Route Filtering

This chapter covers the following subject:

- Configuring Route Filters

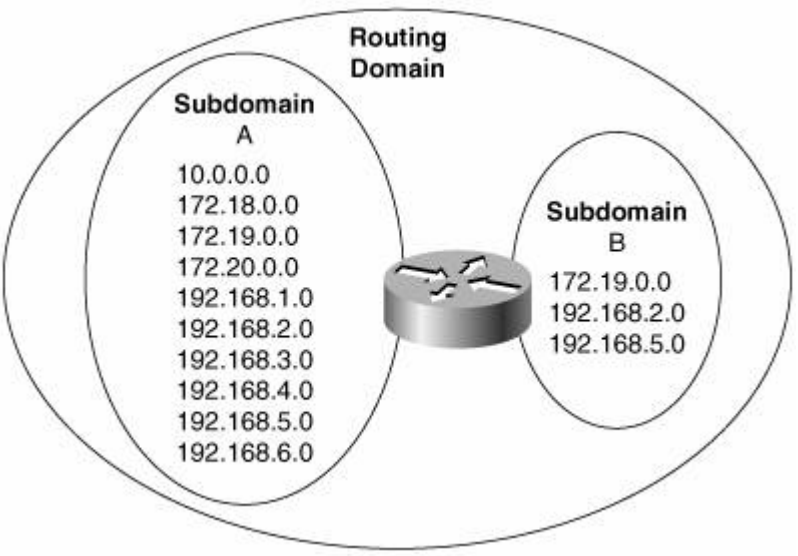
Chapter 11, "Route Redistribution," presents several situations in which redistribution causes unwanted or inaccurate routes to exist in a particular router. For instance, in Figure 11-3 and the associated discussion, one or more routers choose a suboptimal route through a network. The problem in that example is that the routers prefer the lower administrative distance of Interior Gateway routing Protocol (IGRP) to the administrative distance of Routing Information Protocol (RIP). More generally, any time routes to the same destination are being redistributed into a routing domain by more than a single router, the potential for inaccurate routing exists. In some cases, routing loops and black holes might occur.

Example 11-17 shows another example of an unwanted or unexpected route. In this case, the summary route 192.168.3.128/25 is advertised into Open Shortest Path First (OSPF) but is redistributed into the EIGRP domainwhere the summarized subnets exist. This phenomenon, in which a route is advertised in the wrong direction across a redistributing router, is called *route feedback*.

Route filtering enables the network administrator to keep tight control over route advertisements. Any time a router is redistributing routes from one protocol to another, route filters give the network administrator the power to control *what* routes are redistributed. And any time a router is performing *mutual redistribution*the mutual sharing of routes between two or more routing protocolsroute filters should be used to ensure that routes are advertised in only one direction.

Figure 13-1 shows another use for route filters. Here, a routing domain is broken into subdomains, each containing multiple routers. The router connecting the two domains is filtering routes so that the routers in subdomain B know only a subset of the routes in subdomain A. This filtering might be done for security, so that the B routers only recognize authorized subnets. Or the filtering might be a part of a larger traffic-engineering plan, to manage the flow of packets. Or it might be done simply to manage the size of the route tables and updates of the B routers by eliminating unnecessary routes.

Figure 13-1. Route filters can be used to create routing subdomains, into which only some of the routing domain's addresses are advertised.




Yet another common use of route filters is to create a "route firewall." Frequently, corporate divisions or government agencies must be interconnected while they remain under separate administrative control. If you do not have control of all parts of the network, you are vulnerable to misconfigured or even malicious routing. Route filters at the interconnecting routers will ensure that routers accept only legitimate routes. This approach is again a form of security, but in this case, incoming routes, instead of outgoing routes, are regulated.

Whatever the application, route filters are a fundamental building block for creating *routing policies*: A set of rules that govern how packets are forwarded in a network or change the default packet forwarding behavior.

Route filters work by regulating the routes that are entered into, or advertised out of, the route table. They have somewhat different effects on link-state routing protocols than they do on distance-vector routing protocols. A router running a distance-vector protocol advertises routes based on what is in its route table. As a result, a route filter will influence which routes the router advertises to its neighbors.

On the other hand, routers running link-state protocols determine their routes based on information in their link-state database, rather than the advertised route entries of their neighbors. Route filters have no effect on link- state advertisements or the link-state database.[1] As a result, a route filter can influence the route table of the router on which the filter is configured but has no effect on the route entries of neighboring routers. Because of this behavior, route filters are used mostly at redistribution points into link-state domains, such as an OSPF ASBR (autonomous system boundary router), where they can regulate which routes enter or leave the domain. Within the link-state domain, route filters have limited utility.

[1] Remember that a basic requirement of link-state protocols is that all routers in an area must have identical link-state databases. If a route filter blocked some LSAs, this requirement would be violated.

PREV NEXT

# Configuring Route Filters

Route filtering is accomplished by one of the following methods:

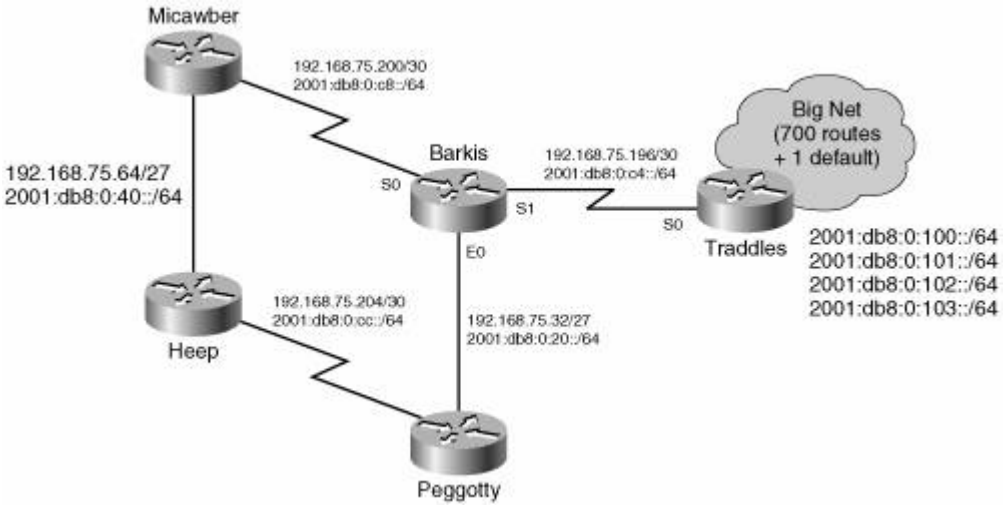
- Filtering specific routes, using the distribute-list command
- Manipulating the administrative distances of routes, using the distance command

## Case Study: Filtering Specific Routes

Figure 13-2 shows a portion of a network running RIPv2 and RIPng. Barkis is providing connectivity to the rest of the Internet via Traddles. In addition to the 700 specific IPv4 routes within BigNet, Traddles is advertising an IPv4 default route to Barkis. Because of the default route, Barkis, Micawber, Peggotty, and Heep do not need to know the other 700 routes in BigNet. So the objective is to configure a filter at Barkis that will accept only the default route from Traddles and reject all other routes. Barkis' configuration is displayed in Example 13-1.

Figure 13-2. A route filter at Barkis will accept only the default route from Traddles and will reject all other BigNet routes.

[View full size image]




Example 13-1. The route filter on Barkis permits the default route and rejects all other addresses.

```
router rip
version 2
network 192.168.75.0
distribute-list 1 in Serial1
!
ip classless
access-list 1 permit 0.0.0.0
```

This route filter examines incoming routes at S1, which is the interface connected to Traddles. It specifies that the only routes accepted by Barkis's RIP process are those permitted by access list 1, and access list 1 specifies that only 0.0.0.0 is permitted.[2] All other routes are implicitly denied by the access list. Example 13-2 shows the resulting route table at Barkis.

[2] Note that no inverse mask is shown. An access list's default inverse mask is 0.0.0.0, which is correct for this configuration.

## Example 13-2. 0.0.0.0 is the only route accepted from Traddles.

```
Barkis#show ip route rip
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is 192.168.75.198 to network 0.0.0.0
    192.168.75.0/24 is variably subnetted, 5 subnets, 2 masks
C      192.168.75.32/27 is directly connected, Ethernet0
R     192.168.75.64/27 [120/1] via 192.168.75.201, 00:00:23, Serial0
C      192.168.75.196/30 is directly connected, Serial1
C      192.168.75.200/30 is directly connected, Serial0
R      192.168.75.204/30 [120/1] via 192.168.75.34, 00:00:13, Ethernet0
R*  0.0.0.0/0 [120/10] via 192.168.75.198, 00:00:03, Serial1
Barkis#
```

IPv6 routes can be filtered in the same way. Barkis accepts only the IPv6 default route from Traddles. Barkis's configuration is displayed in Example 13-3.

## Example 13-3. Barkis filters IPv6 routes to accept only the default route.

```
ipv6 router rip emily
distribute-list prefix-list copperfield in Serial1
!
ipv6 prefix-list copperfield permit ::/0
```

The only difference between the IPv4 and the IPv6 configuration is that the IPv4 distribute-list command references an access list, and the IPv6 distribute-list command references a prefix list. The prefix list permits or denies IPv6 prefixes.

Example 13-4 shows the IPv6 route table at Barkis.

## Example 13-4. ::/0 is the only IPv6 RIP route accepted from Traddles.

```
Barkis#show ipv6 route rip
IPv6 Routing Table - 11 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B  BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
```

```
R   ::/0 [120/2]
     via FE80::205:5EFF:FE6B:50A0, Serial1
R   2001:DB8:0:40::/64 [120/2]
     via FE80::2B0:64FF:FE30:1DE0, Serial0
R   2001:DB8:0:CC::/64 [120/2]
     via FE80::204:C1FF:FE50:F1C0, Ethernet0
Barkis#
```

In Example 13-5, Barkis's configuration is changed to accept 2001:db8:0:100::/64 through 2001:db8:0:103::/64, rather than the default IPv6 route. The prefix list is modified to accept this range of addresses only.

## Example 13-5. Barkis is configured to accept a range of IPv6 addresses into RIPng.

```
ipv6 router rip emily
distribute-list prefix-list copperfield in Serial1
!
ipv6 prefix-list copperfield permit 2001:db8:0:100::/62 le 64
```

A range of IPv6 prefixes is specified by using the ge or the le option to the prefix-list command. ge is "greater than or equal to" and le is "less than or equal to." Barkis's prefix list specifies the prefix 2001:db8:0:100::/62 and any other prefixes that have the same first 62 bits and any value in the bits 63 or 64. This correctly allows 2001:db8:0:100::/64, 2001:db8:0:101::/64, 2001:db8:0:102::/64, and 2001:db8:0:103::/64. Prefix lists are discussed further in Appendix B, "Tutorial: Access Lists."

Of course, advertising 700 routes across a serial link only to discard them at the far end is a waste of bandwidth. So a better configuration would be to place the filter at Traddles, allowing only the default route to be advertised to Barkis, as shown in Example 13-6.

## Example 13-6. Traddles is configured to filter outgoing RIP IPv4 routes.

```
router rip
version 2
network 192.168.63.0
network 192.168.75.0
network 192.168.88.0
distribute-list 1 out Serial0
!
ip classless
access-list 1 permit 0.0.0.0
```

For IPv6, the configuration at Traddles is as displayed in Example 13-7.

## Example 13-7. Traddles is configured to filter outgoing IPv4 RIPng routes.

```
ipv6 router rip emily
distribute-list prefix-list copperfield out Serial0
!
ipv6 prefix-list copperfield permit 2001:db8:0:100::/62 le 64
```

Here, the filter configuration looks almost the same, except that it is filtering outgoing routes instead of incoming routes. Example 13-8 shows that only the default route is being advertised across the serial link from Traddles.

### Example 13-8. The filter at Traddles allows only the default route to be advertised to Barkis.

```
Barkis#debug ip rip
RIP protocol debugging is on
Barkis#
RIP: received v2 update from 192.168.75.198 on Serial1
     0.0.0.0/0 -> 0.0.0.0 in 10 hops
RIP: sending v2 update to 224.0.0.9 via Ethernet0 (192.168.75.33)
     192.168.75.64/27 -> 0.0.0.0, metric 2, tag 0
     192.168.75.196/30 -> 0.0.0.0, metric 1, tag 0
     192.168.75.200/30 -> 0.0.0.0, metric 1, tag 0
     0.0.0.0/0 -> 0.0.0.0, metric 11, tag 0
RIP: sending v2 update to 224.0.0.9 via Serial0 (192.168.75.202)
     192.168.75.32/27 -> 0.0.0.0, metric 1, tag 0
     192.168.75.196/30 -> 0.0.0.0, metric 1, tag 0
     192.168.75.204/30 -> 0.0.0.0, metric 2, tag 0
     0.0.0.0/0 -> 0.0.0.0, metric 11, tag 0
RIP: sending v2 update to 224.0.0.9 via Serial1 (192.168.75.197)
     192.168.75.32/27 -> 0.0.0.0, metric 1, tag 0
     192.168.75.64/27 -> 0.0.0.0, metric 2, tag 0
     192.168.75.200/30 -> 0.0.0.0, metric 1, tag 0
     192.168.75.204/30 -> 0.0.0.0, metric 2, tag 0
RIP: received v2 update from 192.168.75.34 on Ethernet0
     192.168.75.64/27 -> 0.0.0.0 in 2 hops
     192.168.75.204/30 -> 0.0.0.0 in 1 hops
RIP: received v2 update from 192.168.75.201 on Serial0
     192.168.75.64/27 -> 0.0.0.0 in 1 hops
     192.168.75.204/30 -> 0.0.0.0 in 2 hops
```

In both configurations, Barkis will advertise the default route to Micawber and to Peggotty. Neither configuration affects the routes that Barkis advertises to Traddles.

When the distribute-list command is configured under link-state protocols such as OSPF, the out keyword cannot be used in association with an interface.[3] Because link-state protocols do not advertise routes from their route table as do distance-vector protocols, there are no updates to filter. A command such as distribute-list 1 out Serial1 under a link-state protocol is meaningless.

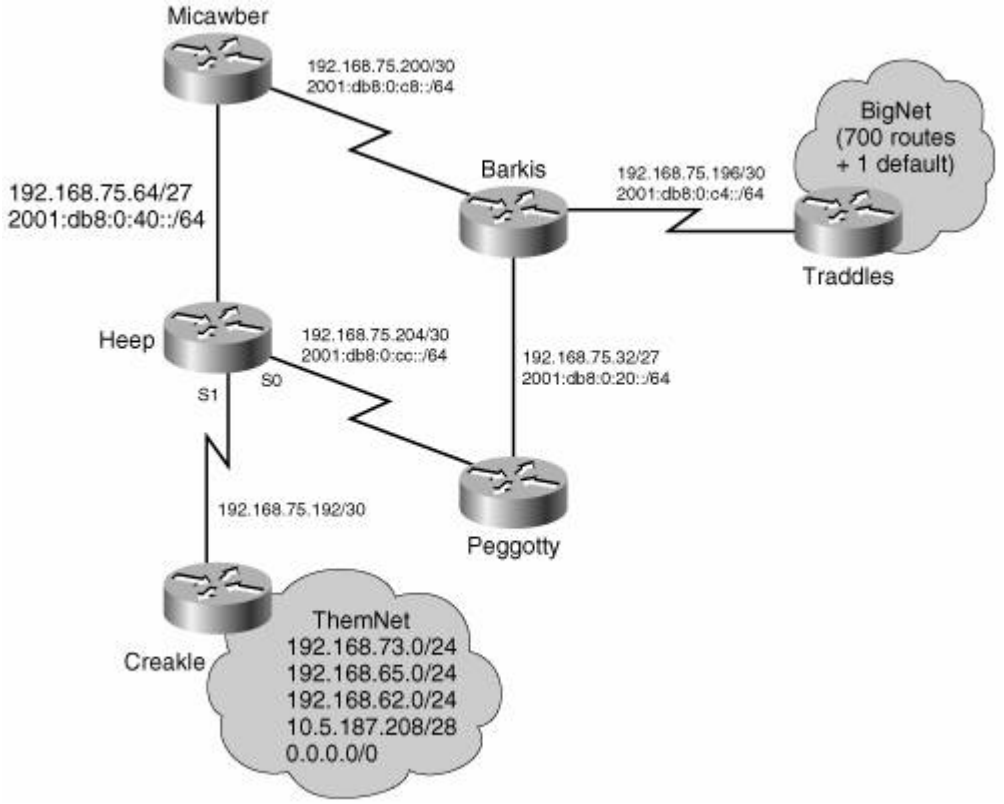
[3] The **out** keyword can be used in association with a routing protocol, as discussed in the next case study.

In Figure 13-3, another group of routers has been connected. This network section, ThemNet, is under separate administrative control, as is the Router Creakle. Because the BigNet administrator has no access to or control over the routers in ThemNet, route filters should be used to minimize the potential of bad routing information being sent into BigNet from Creakle. For example, notice that ThemNet is using a default route (perhaps to access an internal Internet connection). If this default route should be advertised into BigNet, it could cause packets to be misrouted into ThemNet. A black hole would exist.

### Figure 13-3. The network ThemNet is not under the control of the

BigNet administrator.

[View full size image]




To allow only the routes necessary for communication with ThemNet, Heep's configuration is displayed in Example 13-9.

## Example 13-9. Heep's configuration filters RIP routes to and from Creakle.

```
router rip
version 2
network 192.168.75.0
distribute-list 2 out Serial1
distribute-list 1 in Serial1
!
ip classless
access-list 1 permit 192.168.73.0
access-list 1 permit 192.168.65.0
access-list 1 permit 192.168.62.0
access-list 1 permit 10.5.187.208
access-list 2 deny 0.0.0.0
access-list 2 permit any
```

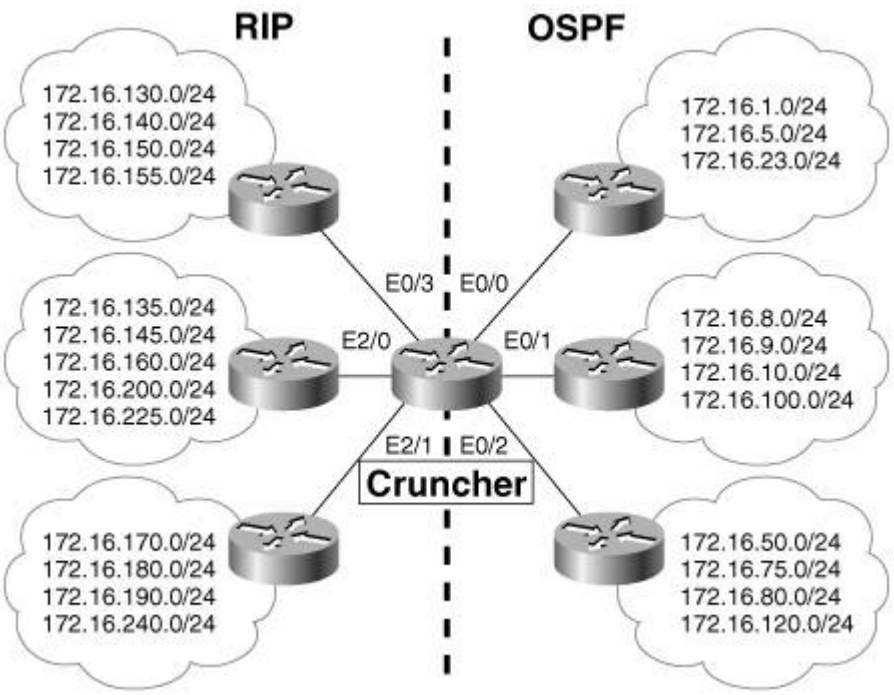
Distribute list 1 allows only the routes specified in access list 1 to be accepted from Creakle. It blocks the default route and any other routes that might be inserted incorrectly into the route tables of ThemNet.

Distribute list 2 is in place to ensure that BigNet is a good neighbor. It blocks the BigNet default route, which would cause problems in ThemNet, and allows all other BigNet routes.

## Case Study: Route Filtering and Redistribution

Any time a router performs mutual redistribution, the potential for route feedback exists. For example, a route from the RIP side in Figure 13-4 can be redistributed into OSPF and, from there, be redistributed back into RIP. Therefore, using route filters to control the direction of route advertisements is a wise approach.

Figure 13-4. Cruncher is redistributing RIP routes into OSPF, and OSPF routes into RIP. Route filters should be used to prevent route feedback.




Cruncher in Figure 13-4 is speaking both RIP and OSPF on several interfaces. Cruncher's configuration is displayed in Example 13-10.

Example 13-10. Cruncher is configured to run both OSPF and RIP, and filters are configured to control which routes are advertised into each protocol.

```
router ospf 25
redistribute rip metric 100
network 172.16.8.254 0.0.0.0 area 25
network 172.16.50.254 0.0.0.0 area 25
distribute-list 3 in Ethernet0/0
distribute-list 3 in Ethernet0/1
distribute-list 3 in Ethernet0/2
!
router rip
version 2
redistribute ospf 25 metric 5
passive-interface Ethernet0/0
passive-interface Ethernet0/1
```

```
passive-interface Ethernet0/2
network 172.16.0.0
distribute-list 1 in Ethernet0/3
distribute-list 1 in Ethernet2/0
distribute-list 1 in Ethernet2/1
!
ip classless
access-list 1 permit 172.16.128.0 0.0.127.255
access-list 3 permit 172.16.0.0 0.0.127.255
```

In this configuration, the logic of the access lists is to permit certain routes and deny all others. The logic can also be reversed to deny certain routes and allow all others, as shown in Cruncher's modified configuration in Example 13-11.

## Example 13-11. Cruncher's access list could be modified to deny specific routes and permit all others.

```
access-list 1 deny 172.16.0.0 0.0.127.255
access-list 1 permit any
access-list 3 deny 172.16.128.0 0.0.127.255
access-list 3 permit any
```

The effect of this second access list configuration is the same as the first. In both cases, routes to destinations in the OSPF domain will not be advertised into OSPF from RIP, and routes to destinations in the RIP domain will not be advertised into RIP from OSPF. However, this second access list configuration needs to be carefully administered to minimize problems in the future. If a new address is added into the RIP domain, for instance, without modifying the route filter, the route will get advertised into OSPF and re-advertised back into RIP, causing a potential route loop. A new deny statement needs to be added to the route filter to control the loop. If the first access list is being used instead of the second, and a new address is added to the RIP domain, it will not be advertised into OSPF until the access lists are modified. No route loops will occur and routing within the RIP domain to the new address will be successful. The second access list configuration is also not as easy to administer, because of the `permit any` at the end. To add new entries to the list, the entire list must first be deleted so that the new entry can be placed before the `permit any`.[4]

[4] In some IOS versions, access-list entries can be added to specific places in the access list using sequence numbers, making the lists easier to administer.

The small access lists shown are possible because the subnet addresses in Figure 13-4 were carefully assigned for easy summarization. The trade-off is a loss of specificity. More precise control over the routes means larger, more precise access lists at the price of increased administrative attention.

An alternative method of configuring route filters at redistribution points is to filter by route process instead of by interface. For example, the configuration in Example 13-12 allows only certain IPv4 routes in Figure 13-4 to be redistributed.

## Example 13-12. Cruncher filters routes at the route process.

```
router ospf 25
redistribute rip metric 100
network 172.16.1.254 0.0.0.0 area 25
network 172.16.8.254 0.0.0.0 area 25
network 172.16.50.254 0.0.0.0 area 25
distribute-list 10 out rip
```

```
!
router rip
version 2
redistribute ospf 25 metric 5
passive-interface Ethernet0/3
passive-interface Ethernet2/0
passive-interface Ethernet2/1
network 172.16.0.0
distribute-list 20 out ospf 25
!
ip classless
access-list 10 permit 172.16.130.0
access-list 10 permit 172.16.145.0
access-list 10 permit 172.16.240.0
access-list 20 permit 172.16.23.0
access-list 20 permit 172.16.9.0
access-list 20 permit 172.16.75.0
```

The route filter under the OSPF configuration allows OSPF to advertise RIP-discovered routes only if access list 10 permits them. Likewise, the filter under the RIP configuration allows RIP to advertise routes discovered by OSPF 25 only if access list 20 permits them. In both cases, the filters have no effect on routes discovered by other protocols. For example, if OSPF were redistributing both RIP and EIGRP routes, the distribute list shown would not apply to EIGRP-discovered routes.

The same configuration can apply to IPv6 prefixes. For RIPng and OSPFv3 for IPv6, the distribute-list references a prefix-list rather then an access-list, and the prefix-list permits or denies IPv6 prefixes rather than IPv4 addresses.

When filtering by route process, only the `out` keyword is allowed. After all, it makes no sense to specify something like `distribute-list 10 in rip` under OSPF. The route has already been entered into the route table by RIP, and OSPF either advertises it ("out") or it does not.

Note that although filtering by routing protocol is useful for specifying which routes will be redistributed, it is not a good method for preventing route feedback. For example, consider the configuration in Example 13-13 for Cruncher in Figure 13-4.

## Example 13-13. Cruncher's configuration filters routes advertised by the OSPF and RIP processes but does not filter routes added into Cruncher's route table.

```
router ospf 25
redistribute rip metric 100
network 172.16.1.254 0.0.0.0 area 25
network 172.16.8.254 0.0.0.0 area 25
network 172.16.50.254 0.0.0.0 area 25
distribute-list 1 out rip
!
router rip
version 2
redistribute ospf 25 metric 5
passive-interface Ethernet0/3
passive-interface Ethernet2/0
passive-interface Ethernet2/1
network 172.16.0.0
distribute-list 3 out ospf 25
!
ip classless
```

```
access-list 1 permit 172.16.128.0 0.0.127.255
access-list 3 permit 172.16.0.0 0.0.127.255
```

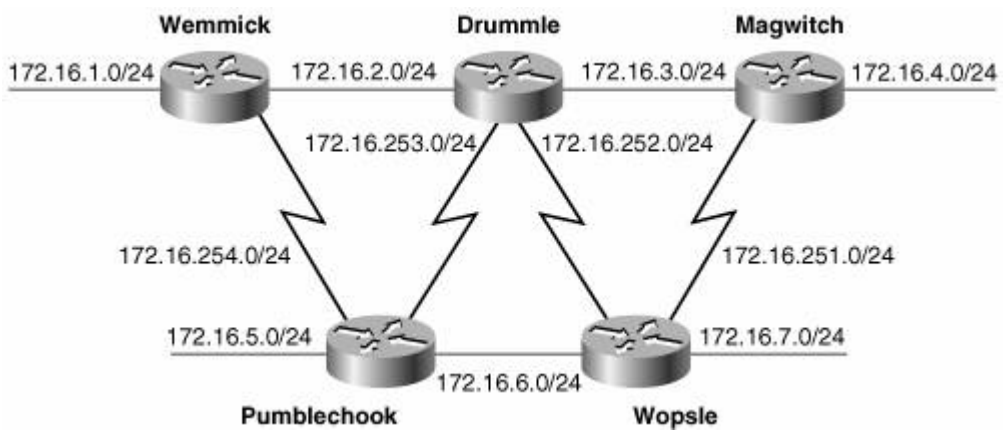
Suppose a route from the RIP domain, such as 172.16.190.0/24, redistributed into the OSPF domain, were to be advertised back to Cruncher because of a misconfigured router. Although the distribute list under the RIP configuration would prevent the route from being advertised back into the RIP domain, the list does nothing to prevent the route from being entered into Cruncher's route table as originating within the OSPF domain. In fact, the filter assumes that the route has already been entered into the table by OSPF. Because OSPF has a lower administrative distance than RIP, Cruncher will prefer the OSPF learned route, and Cruncher will route traffic destined to 172.16.190.0 to the OSPF router rather than to RIP. To prevent route feedback, routes must be filtered as they are incoming on an interface, before they are entered into the route table.

## Case Study: A Protocol Migration

The `distance` command, when used without any optional parameters, specifies the administrative distance to be assigned to routes learned from a particular IPv4 or IPv6 routing protocol. On first consideration, this action might not seem to be a route filtering function, but it is. When multiple routing protocols are running, routes are accepted or rejected into the route table based on their administrative distances.

The network in Figure 13-5 is running RIP, and there is a plan to convert to EIGRP. Several methods exist for conducting such a protocol migration. One option is to turn off the old protocol and turn on the new protocol at each router. Although this option is valid for a small network such as the one in Figure 13-5, the downtime can be impractical in larger networks.

Figure 13-5. These routers are running RIP and are to be converted to EIGRP.




Another option is to add the new protocol without removing the old protocol. If the default administrative distance of the new protocol is lower than that of the old, each router will prefer the routes advertised by the new protocol as they are added. The network will converge to the new protocol as the routers are converted, and after the entire network has converged on the new protocol, the old one can be removed from all routers.

Referring to Table 11-1, the default administrative distance of RIP is 120, and the default administrative distance of EIGRP is 90. If EIGRP is added to each router in addition to RIP, the router will begin preferring the EIGRP routes as its neighbors begin speaking EIGRP. When all RIP routes have disappeared from all route tables, the network has reconverged on EIGRP. The

RIP processes can then be removed from the routers.

The problem with this approach is that the potential for routing loops and black holes exists during the transition period. The five routers in Figure 13-5 can be reconfigured and reconverged in a matter of minutes, so looping might not be as much of a concern here as it would be in a larger network.

A modification of this dual-protocol method is to use the distance command to ensure that the routes of the new protocol are rejected until all routers are ready for conversion. The first step in this procedure is to lower the administrative distance of RIP in all routers, as shown in Example 13-14.

## Example 13-14. Each router in the network in Figure 13-5 is reconfigured to lower the administrative distance of RIP.

```
router rip
network 172.16.0.0
distance 70
```

Note that the administrative distance is relevant only to the routing process of a single router. While RIP is still the only protocol running, the change of administrative distance will have no effect on routing.

Next, each router is revisited, and the EIGRP process is added. The EIGRP configuration in Example 13-15 is added to each router.

## Example 13-15. EIGRP configuration is added to each router in Figure 13-5.

```
router eigrp 1
network 172.16.0.0
!
router rip
network 172.16.0.0
distance 70
```

Because EIGRP has a default administrative distance of 90, the RIP routes are preferred (Example 13-16). No routers are preferring EIGRP, so no network downtime has to be scheduled while the configurations are changed. This approach gives the network administrator time to re-examine the new configurations in each router and to check the EIGRP topology database for accuracy before the conversion.

## Example 13-16. The RIP routes, which have been assigned an administrative distance of 70, are preferred over the EIGRP routes.

```
Drummle#show ip route
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
    172.16.0.0/24 is subnetted, 11 subnets
C      172.16.252.0 is directly connected, Serial0
```

```
C      172.16.253.0 is directly connected, Serial1
R      172.16.254.0 [70/1] via 172.16.2.253, 00:00:16, Ethernet0
                    [70/1] via 172.16.253.253, 00:00:05, Serial1
R      172.16.251.0 [70/1] via 172.16.252.253, 00:00:08, Serial0
                    [70/1] via 172.16.3.253, 00:00:01, Ethernet1
R      172.16.4.0 [70/1] via 172.16.3.253, 00:00:01, Ethernet1
R      172.16.5.0 [70/1] via 172.16.253.253, 00:00:05, Serial1
R     172.16.6.0 [70/1] via 172.16.252.253, 00:00:08, Serial0
                 [70/1] via 172.16.253.253, 00:00:05, Serial1
R      172.16.7.0 [70/1] via 172.16.252.253, 00:00:08, Serial0
R      172.16.1.0 [70/1] via 172.16.2.253, 00:00:17, Ethernet0
C      172.16.2.0 is directly connected, Ethernet0
C      172.16.3.0 is directly connected, Ethernet1
Drummle#
```

The next step in the conversion process is to revisit each router and change the RIP distance back to the default of 120. Network downtime should be scheduled for this step. The RIP routes with the administrative distance of 70 will begin to age out because the new RIP updates will be assigned an administrative distance of 120 (Example 13-17). At 210 seconds, the RIP routes will be declared invalid (Example 13-18), and finally the EIGRP routes will be preferred (Example 13-19).

## Example 13-17. After the RIP administrative distance is changed back to 120, the routes with a distance of 70 begin to age out. Here, all the RIP routes are more than two minutes old.

```
Drummle#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
    172.16.0.0/24 is subnetted, 11 subnets
C      172.16.252.0 is directly connected, Serial0
C      172.16.253.0 is directly connected, Serial1
R     172.16.254.0 [70/1] via 172.16.2.253, 00:02:31, Ethernet0
                   [70/1] via 172.16.253.253, 00:02:18, Serial1
R      172.16.251.0 [70/1] via 172.16.252.253, 00:02:27, Serial0
                    [70/1] via 172.16.3.253, 00:02:32, Ethernet1
R     172.16.4.0 [70/1] via 172.16.3.253, 00:02:32, Ethernet1
R     172.16.5.0 [70/1] via 172.16.253.253, 00:02:19, Serial1
R     172.16.6.0 [70/1] via 172.16.252.253, 00:02:27, Serial0
                 [70/1] via 172.16.253.253, 00:02:19, Serial1
R      172.16.7.0 [70/1] via 172.16.252.253, 00:02:27, Serial0
R      172.16.1.0 [70/1] via 172.16.2.253, 00:02:32, Ethernet0
C      172.16.2.0 is directly connected, Ethernet0
C      172.16.3.0 is directly connected, Ethernet1
```

## Example 13-18. At 3.5 minutes, the RIP routes are declared down.

```
Drummle#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
    172.16.0.0/24 is subnetted, 11 subnets
C      172.16.252.0 is directly connected, Serial0
C      172.16.253.0 is directly connected, Serial1
R      172.16.254.0/24 is possibly down,
         routing via 172.16.253.253, Serial1
R      172.16.251.0/24 is possibly down,
         routing via 172.16.252.253, Serial0
R      172.16.4.0/24 is possibly down,
         routing via 172.16.3.253, Ethernet1
R      172.16.5.0/24 is possibly down,
         routing via 172.16.253.253, Serial1
R      172.16.6.0/24 is possibly down,
         routing via 172.16.253.253, Serial1
R      172.16.7.0/24 is possibly down,
         routing via 172.16.252.253, Serial0
R      172.16.1.0/24 is possibly down,
         routing via 172.16.2.253, Ethernet0
C      172.16.2.0 is directly connected, Ethernet0
C      172.16.3.0 is directly connected, Ethernet1
Drummle#
```

### Example 13-19. The EIGRP routes, with their default distance of 90, replace the RIP routes.

```
Drummle#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
      i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
      U - per-user static route, o - ODR
Gateway of last resort is not set
    172.16.0.0/24 is subnetted, 11 subnets
C      172.16.252.0 is directly connected, Serial0
C      172.16.253.0 is directly connected, Serial1
D      172.16.254.0 [90/2195456] via 172.16.2.253, 00:01:11, Ethernet0
D      172.16.251.0 [90/2195456] via 172.16.3.253, 00:01:06, Ethernet1
D      172.16.4.0 [90/307200] via 172.16.3.253, 00:01:06, Ethernet1
D      172.16.5.0 [90/2195456] via 172.16.253.253, 00:01:11, Serial1
D      172.16.6.0 [90/2185984] via 172.16.252.253, 00:01:11, Serial0
                  [90/2185984] via 172.16.253.253, 00:01:11, Serial1
D      172.16.7.0 [90/2195456] via 172.16.252.253, 00:01:07, Serial0
D      172.16.1.0 [90/307200] via 172.16.2.253, 00:01:11, Ethernet0
C      172.16.2.0 is directly connected, Ethernet0
C      172.16.3.0 is directly connected, Ethernet1
Drummle#
```

Although routing loops and black holes are still a possibility with this method, the conversion should be faster and less prone to human error because the single change of administrative distance is all that is required.

Another advantage of this method is that a back-out, in case of problems, is easy. The RIP processes are still in place on all the routers, so changing the administrative distances back to 70 is all that is needed to return to RIP. After the new EIGRP configuration has been tested and

proven stable, the RIP processes can be deleted from all routers without further service disruptions.

One thing to consider before using the dual-protocol method is the impact that running two protocols concurrently on each router will have on memory and processing. If the utilization of memory or processing or both is averaging above 50 percent to 60 percent, careful lab testing and modeling should be performed before committing to the conversion, to ensure that the routers can handle the additional load. If they can't, a more complex procedure of removing the old protocol before configuring the new protocol might be the only option.

A variant on this procedure is to increase the administrative distance of the new protocol, rather than decreasing the distance of the old protocol, and then lowering the distance of the new protocol at conversion time. However, be sure to enter the `distance` command before any network commands so that the new protocol won't first activate at its default distance.

Looking again at Table 11-1, notice that EIGRP has two default administrative distances: 90 for internal routes and 170 for external routes. Therefore, the `distance` command is also different for EIGRP. For example, to raise the EIGRP distance instead of lowering the RIP distance, the configuration is as displayed in Example 13-20.

Example 13-20. EIGRP's internal administrative distance is raised. This could be done in all routers in Figure 13-5 rather then lowering the RIP administrative distance.

```
router eigrp 1
network 172.16.0.0
distance eigrp 130 170
!
router rip
network 172.16.0.0
```

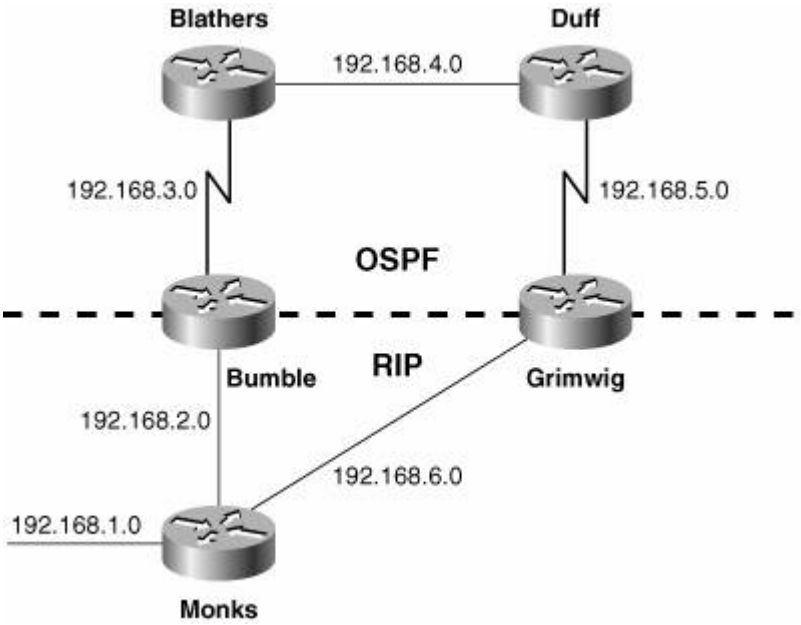
The keyword `eigrp` is added to indicate that EIGRP distances are specified. The administrative distance of internal EIGRP routes is changed to 130, and the distance of external routes is left at 170.

One final note on using the dual-protocol procedure to migrate to a new routing protocol: Be sure you understand the behavior of both protocols. For example, some protocols, such as EIGRP, do not age out their route entries. Therefore, if EIGRP is being replaced, an additional step of the conversion process is to clear the route tables with the `clear ip route *` command after the distances have been changed.

## Case Study: Multiple Redistribution Points

Figure 13-6 shows a network very similar to the one depicted in Figure 11-3. Recall from the associated discussion in Chapter 11 that the problem with multiple redistribution points is that administrative distances can cause routers to choose undesirable paths. In some cases, route loops and black holes can result. For example, Bumble's route table (Example 13-21) shows that it is routing to network 192.168.6.0 through Blathers, rather than using the preferable route through Monks.

Figure 13-6. When mutual redistribution is performed at more than one point, as in this network, administrative distances can cause suboptimal routing, route loops, and black holes.




Example 13-21. The route Bumble is using to reach to network 192.168.6.0 via Blathers (192.168.3.2) involves crossing two serial links and a token ring.

```
Bumble#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
R     192.168.1.0/24 [120/1] via 192.168.2.1, 00:00:00, Ethernet0
C     192.168.2.0/24 is directly connected, Ethernet0
C     192.168.3.0/24 is directly connected, Serial0
O     192.168.4.0/24 [110/70] via 192.168.3.2, 00:05:09, Serial0
O     192.168.5.0/24 [110/134] via 192.168.3.2, 00:05:09, Serial0
O E2 192.168.6.0/24 [110/100] via 192.168.3.2, 00:05:09, Serial0
Bumble#
```

One solution to this problem is to use the distribute-list command to control the source of the routes at the redistribution points. The configurations of Bumble and Grimwig are displayed in Example 13-22 and Example 13-23, respectively.

Example 13-22. Bumble's configuration permits only addresses in the OSPF domain to be accepted into the OSPF process and permits only addresses in the RIP domain to be accepted into the RIP process.

```
router ospf 1
redistribute rip metric 100
network 192.168.3.1 0.0.0.0 area 0
distribute-list 1 in
!
router rip
redistribute ospf 1 metric 2
```

```
network 192.168.2.0
distribute-list 2 in
!
ip classless
access-list 1 permit 192.168.4.0
access-list 1 permit 192.168.5.0
access-list 2 permit 192.168.1.0
access-list 2 permit 192.168.6.0
```

### Example 13-23. Grimwig's configuration permits only addresses in the OSPF domain to be accepted into the OSPF process and permits only addresses in the RIP domain to be accepted into the RIP process.

```
router ospf 1
redistribute rip metric 100
network 192.168.5.1 0.0.0.0 area 0
distribute-list 1 in
!
router rip
redistribute ospf 1 metric 2
network 192.168.6.0
distribute-list 2 in
!
ip classless
access-list 1 permit 192.168.3.0
access-list 1 permit 192.168.4.0
access-list 2 permit 192.168.1.0
access-list 2 permit 192.168.2.0
```

In both configurations, access list 1 allows only addresses within the OSPF domain to be accepted by OSPF, and access list 2 allows only addresses within the RIP domain to be accepted by RIP. Example 13-24 shows Bumble's route table after the route filter has been configured.

### Example 13-24. After the route filters are configured, Bumble is using the preferred route to reach network 192.168.6.0.

```
Bumble#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
R    192.168.1.0/24 [120/1] via 192.168.2.1, 00:00:12, Ethernet0
C    192.168.2.0/24 is directly connected, Ethernet0
C    192.168.3.0/24 is directly connected, Serial0
O    192.168.4.0/24 [110/70] via 192.168.3.2, 00:00:22, Serial0
O    192.168.5.0/24 [110/134] via 192.168.3.2, 00:00:22, Serial0
R    192.168.6.0/24 [120/1] via 192.168.2.1, 00:00:13, Ethernet0
Bumble#
```

The problem with this configuration is that it eliminates the redundancy inherent in multiple redistribution points. In Example 13-25, Bumble's Ethernet link has been disconnected. Because routes to the RIP networks are being filtered in OSPF, all those routes are now unreachable.

## Example 13-25. When Bumble's Ethernet link fails, the RIP networks become unreachable. The route filters prevent OSPF from entering the alternative routes into the route table.

```
Bumble#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state to down
Bumble#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
C    192.168.3.0/24 is directly connected, Serial0
O    192.168.4.0/24 [110/70] via 192.168.3.2, 00:06:45, Serial0
O    192.168.5.0/24 [110/134] via 192.168.3.2, 00:06:45, Serial0
Bumble#
```

A better approach for IPv4 is to use two forms of the distance command to set preferred routes.[5] The configurations of Bumble and Grimwig are displayed in Example 13-26 and Example 13-27, respectively.

[5] IPv6 routing protocols also support the **distance** command to change the administrative distances. The IPv6 form of the command only has the option of changing the distance value. There are currently no options for specifying neighbor addresses or prefix-lists.

## Example 13-26. Bumble uses the distance configuration command to alter the administrative distance of IP routes.

```
router ospf 1
redistribute rip metric 100
network 192.168.3.1 0.0.0.0 area 0
distance 130
distance 110 0.0.0.0 255.255.255.255 1
!
router rip
redistribute ospf 1 metric 2
network 192.168.2.0
distance 130
distance 120 192.168.2.1 0.0.0.0 2
!
ip classless
access-list 1 permit 192.168.4.0
access-list 1 permit 192.168.5.0
access-list 2 permit 192.168.1.0
access-list 2 permit 192.168.6.0
```

## Example 13-27. Grimwig uses the distance command to alter the administrative distances of IP routes.

```
router ospf 1
redistribute rip metric 100
network 192.168.5.1 0.0.0.0 area 0
distance 130
```

```
distance 110 0.0.0.0 255.255.255.255 1
!
router rip
redistribute ospf 1 metric 2
network 192.168.6.0
distance 130
distance 120 192.168.6.1 0.0.0.0 2
!
ip classless
access-list 1 permit 192.168.3.0
access-list 1 permit 192.168.4.0
access-list 2 permit 192.168.1.0
access-list 2 permit 192.168.2.0
```

The first `distance` command in both configurations sets the default distance of both OSPF and RIP routes to 130. The second `distance` command sets a different distance according to the advertising router specified and the referenced access list. For instance, Grimwig's RIP process assigns a distance of 120 to routes advertised by Monks (192.168.6.1) that are permitted by access list 2. All other routes are given a distance of 130. Notice that an inverse mask is used with the address of the advertising router.

OSPF is more problematic. The address of the advertising router is not necessarily the interface address of the next-hop router. Rather, it is the Router ID of the router originating the LSA from which the route is calculated. Therefore, the address and inverse mask for the `distance` command under OSPF is 0.0.0.0 255.255.255.255, specifying any router.[6] OSPF routes from all routers that are permitted by access list 1 are assigned a distance of 110; all other routes are given a distance of 130.

[6] The same "any" address can be used with RIP. A specific address was used for demonstration purposes.

The result appears in Example 13-28. The first route table shows that Grimwig is routing to all networks in the OSPF domain via Duff and to all networks in the RIP domain via Monks. The normal distances of 110 for OSPF and 120 for RIP are assigned to the routes. Next, Grimwig's Ethernet connection fails. The second route table shows that all networks are now reachable via Duff. The routes to the networks in the RIP domain have a distance of 130. When the Ethernet connection is restored, the RIP advertisements from Monks, with a distance of 120, will replace the OSPF advertisements with a distance of 130.

## Example 13-28. Grimwig's route table before and after its Ethernet link to Monks is disconnected.

```
Grimwig#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
R    192.168.1.0/24 [120/1] via 192.168.6.1, 00:00:19, Ethernet0
R    192.168.2.0/24 [120/1] via 192.168.6.1, 00:00:19, Ethernet0
O    192.168.3.0/24 [110/134] via 192.168.5.2, 00:15:06, Serial0
O    192.168.4.0/24 [110/70] via 192.168.5.2, 00:15:06, Serial0
C    192.168.5.0/24 is directly connected, Serial0
C    192.168.6.0/24 is directly connected, Ethernet0
Grimwig#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0, changed state to down
Grimwig#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
O E2 192.168.1.0/24 [130/100] via 192.168.5.2, 00:00:08, Serial0
O E2 192.168.2.0/24 [130/100] via 192.168.5.2, 00:00:08, Serial0
O    192.168.3.0/24 [110/134] via 192.168.5.2, 00:16:23, Serial0
O    192.168.4.0/24 [110/70] via 192.168.5.2, 00:16:23, Serial0
C    192.168.5.0/24 is directly connected, Serial0
O E2 192.168.6.0/24 [130/100] via 192.168.5.2, 00:00:08, Serial0
Grimwig#
```

If one of the serial links in Figure 13-6 fails, the opposite will happen. The networks within the OSPF domain will become reachable through the RIP domain, again with a distance of 130 (Example 13-29). However, unlike the speedy reconvergence of OSPF, the RIP side will take several minutes to reconverge. This slow reconvergence is due to RIP's split horizon at Monks. That router will not advertise the OSPF routes to Bumble or Grimwig until the same routes have ceased to be advertised from one of those routers and the existing routes have timed out.

## Example 13-29. Grimwig's route table before and after its serial link to Duff is disconnected.

```
Grimwig#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
R    192.168.1.0/24 [120/1] via 192.168.6.1, 00:00:04, Ethernet0
R    192.168.2.0/24 [120/1] via 192.168.6.1, 00:00:04, Ethernet0
O    192.168.3.0/24 [110/134] via 192.168.5.2, 00:00:12, Serial0
O    192.168.4.0/24 [110/70] via 192.168.5.2, 00:00:12, Serial0
C    192.168.5.0/24 is directly connected, Serial0
C    192.168.6.0/24 is directly connected, Ethernet0
Grimwig#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed state to down
%LINK-3-UPDOWN: Interface Serial0, changed state to down
Grimwig#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route
Gateway of last resort is not set
R    192.168.1.0/24 [120/1] via 192.168.6.1, 00:00:07, Ethernet0
R    192.168.2.0/24 [120/1] via 192.168.6.1, 00:00:07, Ethernet0
R    192.168.3.0/24 [130/3] via 192.168.6.1, 00:00:07, Ethernet0
R    192.168.4.0/24 [130/3] via 192.168.6.1, 00:00:07, Ethernet0
R    192.168.5.0/24 is possibly down, routing via 192.168.6.1, Ethernet0
C    192.168.6.0/24 is directly connected, Ethernet0
Grimwig#
```

The solution to this problem is to disable split horizon on Monks's two Ethernet interfaces with the command `no ip split-horizon`. Although disabling split horizon is trading some loop protection for shorter reconvergence times, it is a good trade in this case. The distance-based route filters at Bumble and Grimwig will prevent all multi-hop loops, and the split horizon

function on the same two routers' Ethernet interfaces will break single-hop loops.

There is another option that can assist in filtering addresses between routing protocols that are mutually redistributing routes. This is route tagging. As routes are being redistributed into a routing protocol, the routes can be tagged. The tag is a number that the administrator sets, that will uniquely identify the set of routes that have been redistributed at this router. The tag can be used at another router to filter these routes from being redistributed back into the originating routing protocol. Tags are set using route maps and are discussed further in Chapter 14, "Route Maps."

## Case Study: Using Administrative Distances to Set Router Preferences

Suppose policy states that Monks in Figure 13-6 will use Grimwig as its primary gateway to the OSPF domain and route through Bumble only if Grimwig becomes unavailable. Presently, Monks is performing equal-cost load balancing between Grimwig and Bumble to reach the OSPF networks (Example 13-30).

### Example 13-30. Monks sees all of the networks in the OSPF domain as equidistant through Bumble and Grimwig.

```
Monks#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
C    192.168.1.0/24 is directly connected, Ethernet2
C    192.168.2.0/24 is directly connected, Ethernet0
R    192.168.3.0/24 [120/2] via 192.168.2.2, 00:00:26, Ethernet0
                    [120/2] via 192.168.6.2, 00:00:23, Ethernet1
R    192.168.4.0/24 [120/2] via 192.168.2.2, 00:00:26, Ethernet0
                    [120/2] via 192.168.6.2, 00:00:23, Ethernet1
R    192.168.5.0/24 [120/2] via 192.168.2.2, 00:00:26, Ethernet0
                    [120/2] via 192.168.6.2, 00:00:23, Ethernet1
C    192.168.6.0/24 is directly connected, Ethernet1
Monks#
```

Monks can be configured to prefer Grimwig by lowering the distance of all routes from that router, as in Example 13-31.

### Example 13-31. Monks is configured to lower the administrative distance for all routes advertised from Grimwig.

```
router rip
network 192.168.1.0
network 192.168.2.0
network 192.168.6.0
distance 100 192.168.6.2 0.0.0.0
```

Here, the distance command makes no reference to an access list. All routes advertised by Grimwig (192.168.6.2) will be assigned an administrative distance of 100. All other routes (meaning routes from Bumble) will be assigned the default RIP administrative distance of 120.

Therefore, Grimwig's routes will be preferred.

Example 13-32 shows the results. The first route table shows that Monks is now routing to Grimwig only. After a failure of the connection to Grimwig, the bottom route table shows that Monks has switched to Bumble (192.168.2.2).

## Example 13-32. Monks's route table before and after its Ethernet link to Grimwig is disconnected.

```
Monks#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
C    192.168.1.0/24 is directly connected, Ethernet2
C    192.168.2.0/24 is directly connected, Ethernet0
R    192.168.3.0/24 [100/2] via 192.168.6.2, 00:00:12, Ethernet1
R    192.168.4.0/24 [100/2] via 192.168.6.2, 00:00:12, Ethernet1
R    192.168.5.0/24 [100/2] via 192.168.6.2, 00:00:12, Ethernet1
C    192.168.6.0/24 is directly connected, Ethernet1
Monks#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet1, changed state to down
Monks#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
C    192.168.1.0/24 is directly connected, Ethernet2
C    192.168.2.0/24 is directly connected, Ethernet0
R    192.168.3.0/24 [120/2] via 192.168.2.2, 00:00:00, Ethernet0
R    192.168.4.0/24 [120/2] via 192.168.2.2, 00:00:00, Ethernet0
R    192.168.5.0/24 [120/2] via 192.168.2.2, 00:00:00, Ethernet0
R    192.168.6.0/24 [120/2] via 192.168.2.2, 00:00:00, Ethernet0
Monks#
```

PREV NEXT

# Looking Ahead

Route filters are very useful tools for controlling the behavior of your network. In large networks, route filters are nearly indispensable. Yet for all their utility, they still can only allow or disallow routes. Chapter 14, introduces route maps, powerful tools that not only identify routes but also actively modify them.

# Summary Table: Chapter 13 Command Review

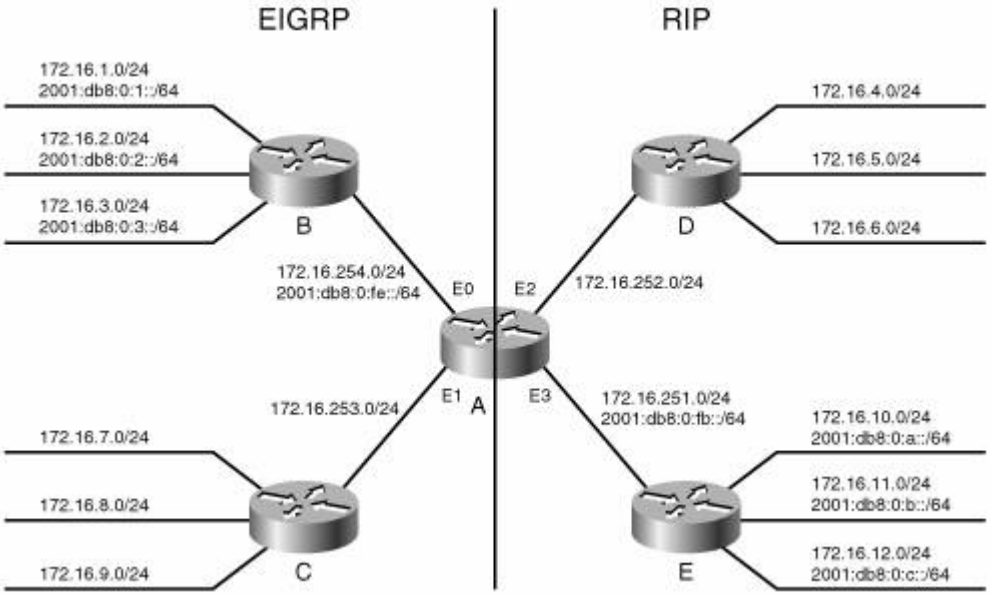
| Command | Description |
|---|---|
| access-list *access-list-number*{deny\| permit} *source*[*source-wildcard*] | Defines a line of a standard IP access list |
| distance *weight* [*address mask*[*access-list-number*\| *name*]] | Defines an administrative distance other than the default |
| distance eigrp *internal-distance external-distance* | Defines the administrative distances other than the default of internal and external EIGRP routes |
| distance ospf{[intra-area *dist1*] [inter-area *dist2*] [external *dist3*]} | Defines the administrative distances other than the default of intra-area, inter-area, and external routes |
| distribute-list{*access-list-number*\| *name*} in [*interface-name*] | Filters the IPv4 routes in incoming updates |
| distribute-list{*access-list-number*\| *name*} out [*interface-name*\| *routing-process*\| *autonomous-system-number*] | Filters the IPv4 routes in outgoing updates |
| distribute-list prefix-list *list_name*[in\| out] | Filters the IPv6 routes in incoming or outgoing updates |
| ipv6 prefix-list *list_name* [seq *num*] {deny\| permit} *prefix/length*[ge *length* \| le *length*] | Defines a line of an IPv6 prefix list, specifying a prefix or group of prefixes to permit or deny |
| redistribute *protocol*[*process-id*] {level-1\| level-1-2\| level-2} [metric *metric-value*] [metric-type *type-value*] [match{internal\| external 1\| external 2}] [tag *tag-value*] [route-map *map-tag*] [weight *weight*] [subnets] | Configures IPv4 redistribution into a routing protocol and specifies the source of the redistributed routes |

# Configuration Exercises

1 The routing configuration for Router A in Figure 13-7 is displayed in Example 13-33.

Figure 13-7. The network for Configuration Exercises 1 through 4.

[View full size image]




Configure a route filter at A that will prevent subnet 172.16.12.0/24 from being known by any router other than E.

Example 13-33. Router A's configuration for the network shown in Figure 13-7.

```
router rip
redistribute eigrp 1 metric 3
passive-interface Ethernet0
passive-interface Ethernet1
network 172.16.0.0
!
router eigrp 1
redistribute rip metric 10000 1000 255 1 1500
passive-interface Ethernet2
passive-interface Ethernet3
network 172.16.0.0
```

2 Configure a route filter at A in Figure 13-7 that will prevent D from learning about subnet 172.16.10.0/24.

3 Configure a route filter at A in Figure 13-7 that will allow only subnets 172.16.2.0/24, 172.16.8.0/24, and 172.16.9.0/24 to be advertised into the RIP domain.

4 Configure a route filter at A in Figure 13-7 that will prevent B from learning about any of the subnets in the RIP domain.

5 OSPFv3 for IPv6 is added to Routers A, B and C. RIPng is added to Routers A, D, and E. IPv6 prefixes 2001:db8:0:1::/64, 2001:db8:0:2::/64, and 2001:db8:0:3::/64 are connected to Router B. Prefixes 2001:db8:0:a::/64, 2001:db8:0:b::/64, and 2001:db8:0:c::/64 are connected to Router E. Configure a route filter at A in Figure 13-7 that prevents the prefixes 2001:db8:0:a::/64 and 2001:db8:0:b::/64 from being advertised to D.

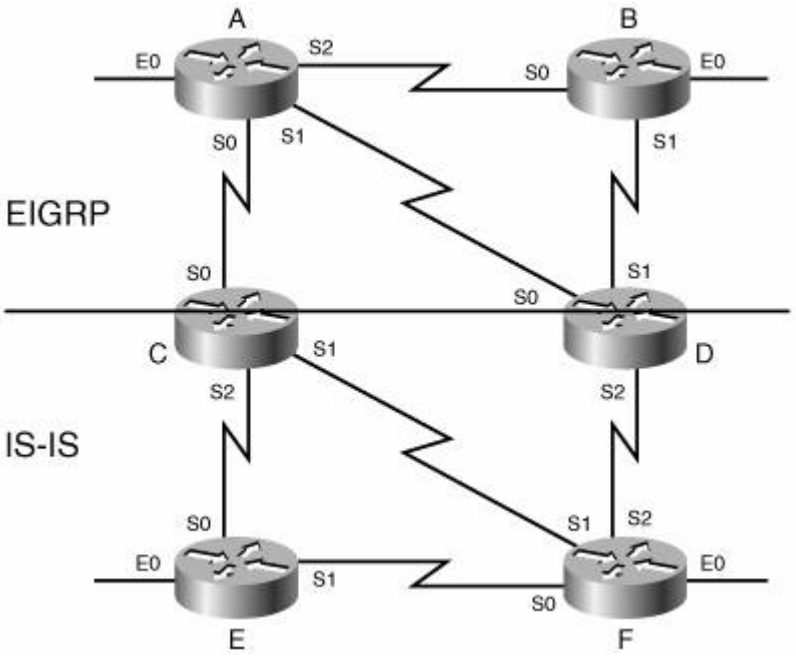
6 Table 13-1 shows the interface addresses for all routers in Figure 13-8. Routers A and B are running EIGRP, and Routers E and F are running IS-IS. C and D are redistributing. Configure distance commands for C and D that will prevent loops and route feedback but will still allow redundant paths.

Table 13-1. The interface addresses of the routers in Figure 13-8.

| Router | Interface | Address | Mask |
|---|---|---|---|
| A | E0 | 192.168.1.1 | 255.255.255.0 |
| | S0 | 192.168.10.254 | 255.255.255.252 |
| | S1 | 192.168.10.249 | 255.255.255.252 |
| | S2 | 192.168.10.245 | 255.255.255.252 |
| B | E0 | 192.168.2.1 | 255.255.255.0 |
| | S0 | 192.168.10.246 | 255.255.255.252 |
| | S1 | 192.168.10.241 | 255.255.255.252 |
| C | S0 | 192.168.10.253 | 255.255.255.252 |
| | S1 | 192.168.10.234 | 255.255.255.252 |
| | S2 | 192.168.10.225 | 255.255.255.252 |
| D | S0 | 192.168.10.250 | 255.255.255.252 |
| | S1 | 192.168.10.242 | 255.255.255.252 |
| | S2 | 192.168.10.237 | 255.255.255.252 |
| E | E0 | 192.168.4.1 | 255.255.255.0 |
| | S0 | 192.168.10.226 | 255.255.255.252 |
| | S1 | 192.168.10.229 | 255.255.255.252 |
| F | E0 | 192.168.3.1 | 255.255.255.0 |
| | S0 | 192.168.10.230 | 255.255.255.252 |

| Router | Interface | Address | Mask |
|---|---|---|---|
| | S1 | 192.168.10.233 | 255.255.255.252 |
| | S2 | 192.168.10.238 | 255.255.255.252 |

Figure 13-8. The network for Configuration Exercises 6 through 8.




7 Using the distance command, configure Router D in Figure 13-8 to accept EIGRP routes only from Router A. If the link to A fails, D should not accept routes from Router B, although D should still advertise routes to B.

8 Remove the configuration added to D in Configuration Exercise 7. Configure Router C in Figure 13-8 to route to all destinations, including the networks and subnets of the IS-IS domain, via Router A. C should route through E and F only if the link to A fails.

PREV NEXT

# Troubleshooting Exercises

1 A router has the configuration in Example 13-34.

Example 13-34. A router configuration attempting to filter the default route.

```
router eigrp 1
network 10.0.0.0
distribute-list 1 in Ethernet5/1
!
access-list 1 deny 0.0.0.0 255.255.255.255
access-list 1 permit any
```

The intention is to deny the incoming default route at interface E5/1 and to permit all other routes incoming on that interface. However, no routes are being accepted on E5/1. What is wrong?

2 Grimwig in Figure 13-6 has the configuration in Example 13-35.

Example 13-35. Grimwig's configuration for troubleshooting Exercise 2.

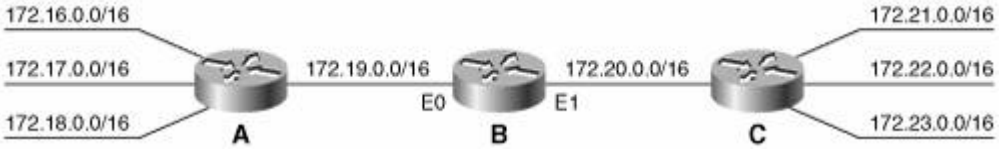
```
router ospf 1
redistribute rip metric 100
network 192.168.5.1 0.0.0.0 area 0
distance 255
distance 110 0.0.0.0 255.255.255.255 1
!
router rip
redistribute ospf 1 metric 2
network 192.168.6.0
distance 255
distance 120 192.168.6.1 0.0.0.0 2
!
ip classless
access-list 1 permit 192.168.3.0
access-list 1 permit 192.168.4.0
access-list 2 permit 192.168.1.0
access-list 2 permit 192.168.2.0
```

What effect will this configuration have on the routes at Grimwig?

3 The routers in Figure 13-9 are running OSPF. Router B has the configuration in Example 13-36.

Figure 13-9. The network for Troubleshooting Exercises 3 and

4.




Example 13-36. Router B's OSPF configuration from Figure 13-9.

```
router ospf 50
network 0.0.0.0 255.255.255.255 area 1
distribute-list 1 in
!
access-list 1 deny 172.17.0.0
access-list 1 permit any
```

The intention is to prevent Routers B and C from having a route entry for network 172.17.0.0. This plan seems to be working at Router B, but Router C still has an entry for 172.17.0.0. Why?

4 The routers in Figure 13-9 are running RIP. Router B has the configuration in Example 13-37.

Example 13-37. Router B's RIP configuration.

```
router rip
network 172.19.0.0
network 172.20.0.0
distribute-list 1 out Ethernet0
distribute-list 2 out Ethernet1
!
access-list 1 permit 172.18.0.0
access-list 2 permit 172.22.0.0
```

The intention is to advertise only network 172.22.0.0 to Router A and to advertise only network 172.18.0.0 to Router C. However, A and C have no RIP entries in their route table. What is wrong?

PREV NEXT

# Chapter 14. Route Maps

This chapter covers the following subjects:

- Basic Uses of Route Maps
- Configuring Route Maps

Route maps are similar to access lists; they both have criteria for matching the details of certain packets and an action of permitting or denying those packets. Unlike access lists, though, route maps can add to each "match" criterion a "set" criterion that actually changes the packet in a specified manner, or changes route information in a specified manner. Route maps also allow you more options for matching a given packet. Altogether, route maps are a powerful tool for creating customized routing policies in your network.

# Basic Uses of Route Maps

Route maps can be used for both redistribution and for policy routing. Although redistribution has been covered extensively in previous chapters, this chapter introduces the topic of policy routing. And most commonly, they are an essential tool in large-scale Border Gateway Protocol (BGP) implementations. The use of route maps to implement BGP routing policies is outside the scope of this volume, and is covered in "Routing TCP/IP," Volume II (CCIE Professional Development) (1-57870-089-2).

*Policy routes* are nothing more than sophisticated static routes. Whereas static routes forward a packet to a specified next hop based on the destination address of the packet, policy routes can forward a packet to a specified next hop based on the source of the packet or other fields in the packet header. Policy routes can also be linked to extended IP access lists so that routing might be based on such things as protocol types and port numbers. Like a static route, a policy route influences the routing only of the router on which it is configured.

Figure 14-1 shows an example of a typical policy routing application. AbnerNet is connected to two Internet service providers via router Dogpatch. AbnerNet's corporate policy dictates that some users' Internet traffic should be sent via ISP 1 and other users' Internet traffic should be sent via ISP 2. If either ISP should become unavailable, the traffic normally using that provider would be sent to the other provider. A policy route at Dogpatch can distribute Internet traffic in accordance with local policy. The distribution of traffic might be based on subnet, specific user, or even user applications.

Figure 14-1. Policy routing allows traffic from AbnerNet to be routed to one of its two Internet service providers based on parameters such as source address, source/destination address combinations, packet size, or application-level ports.

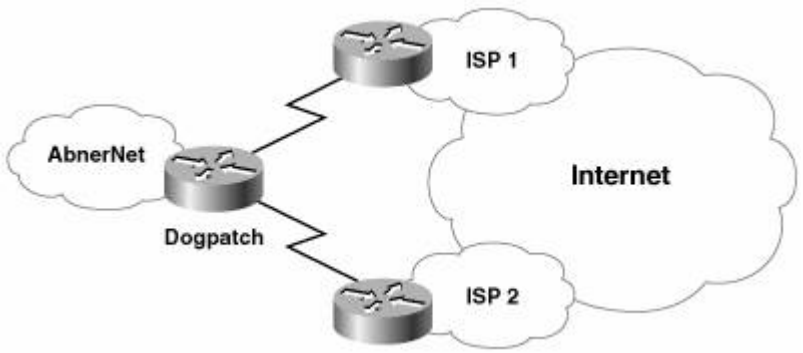
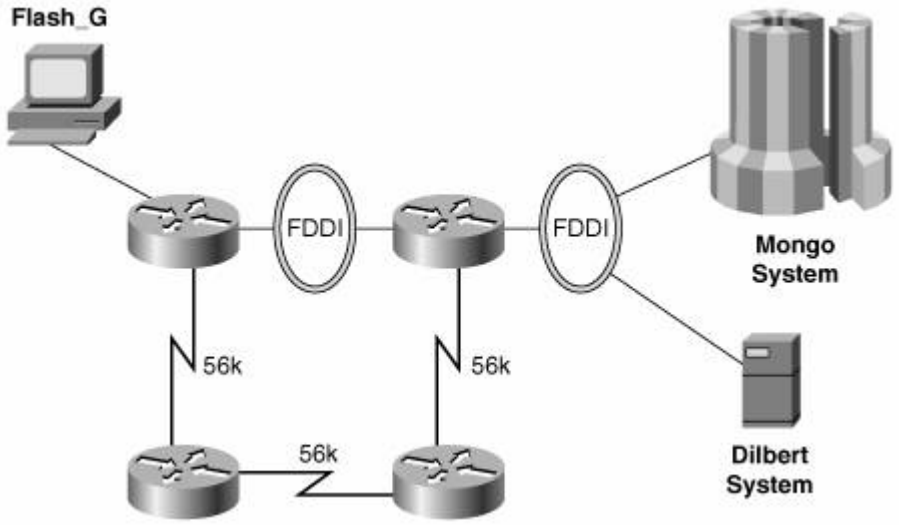



Figure 14-2 shows another use for policy routing. One of the systems on the right watches for invasion forces from the planet Mongo, while the other stores back copies of *Dilbert* comic strips. Policy routes can be configured to route the critical traffic from the Mongo System to Flash_G over the FDDI (Fiber Distributed Data Interface) link and to route the lower-priority Dilbert traffic over the 56K linksor vice versa.

Figure 14-2. Policy routing allows high-priority traffic from the Mongo System to be routed over the FDDI link while low-priority traffic from

the Dilbert System is routed over the 56K links.




Tables 14-1 and 14-2 show the match and set commands that can be used with redistribution, and Tables 14-3 and 14-4 show the match and set commands that can be used with policy routing.

Table 14-1. *match* commands that can be used with redistribution.

| Command | Description |
|---|---|
| match interface *type number* [...*type number*] | Matches routes that have their next hop out one of the interfaces specified |
| match ip address {*access-list-number*\| *name*} [...*access-list-number*\| *name*] | Matches routes that have a destination address specified by one of the access lists |
| match ip next-hop{*access-list-number*\| *name*} [...*access-list-number*\| *name*] | Matches routes that have a next-hop router address specified by one of the access lists |
| match ip route-source{*access-list-number*\| *name*} [...*access-list-number*\| *name*] | Matches routes that have been advertised by routers at the addresses specified in the access lists |
| match metric *metric-value* | Matches routes with the specified metric |
| match route-type {internal\| external[type-1\| type-2] \| level-1\| level-2} | Matches OSPF, EIGRP, or IS-IS routes of the specified type |
| match tag *tag-value* [...*tag-value*] | Matches routes with the specified tags |

Table 14-2. *set* commands that can be used with redistribution.

| Command | Description |
|---|---|
| set level{level-1\| level-2\| level-1-2\| stub-area\| backbone} | Sets the IS-IS level or the OSPF area into which a matched route is to be redistributed |

| Command | Description |
| --- | --- |
| set metric{*metric-value*\| *bandwidth delay reliability loading mtu*} | Sets the metric value for a matched route |
| set metric-type{internal\| external\| type-1\| type-2} | Sets the metric type for a matched route being redistributed into IS-IS or OSPF |
| set next-hop *next-hop* | Sets the next-hop router address for a matched route |
| set tag *tag-value* | Sets a tag value for a matched route |

Table 14-3. *match* commands that can be used with policy routing.

| Command | Description |
| --- | --- |
| match ip address {*access-list-number*\| *name*} [...*access-list-number*\| *name*] | Matches a packet with the characteristics specified in the standard or extended access lists |
| match length *min max* | Matches the Layer 3 length of a packet |

Table 14-4. *set* commands that can be used with policy routing.

| Command | Description |
| --- | --- |
| set default interface *type number* [...*type number*] | Sets the outgoing interface for matched packets when there is no explicit route to the destination |
| set interface *type number* [...*type number*] | Sets the outgoing interface for matched packets when there is an explicit route to the destination |
| set ip default next-hop *ip-address* [...*ip-address*] | Sets the next-hop router address for matched packets when there is no explicit route to the destination |
| set ip next-hop *ip-address* [...*ip-address*] | Sets the next-hop router address for matched packets when there is an explicit route to the destination |
| set ip precedence *precedence* | Sets the precedence bits in the Type of Service field of matched IP packets |
| set ip tos *type-of-service* | Sets the TOS bits in the Type of Service field of matched packets |

# Configuring Route Maps

Like access lists (see Appendix B, "Tutorial: Access Lists"), route maps by themselves affect nothi they must be "called" by some command. The command will most likely be either a policy routing command or a redistribution command. Policy routing will send packets to the route map, whereas redistribution will send routes to the route map. The case studies in this section demonstrate the u route maps for both redistribution and policy routing.

Route maps are identified by a name. For example, the route map in Example 14-1 is named Haga

### Example 14-1. A route map named Hagar is defined in this configuratior

```
route-map Hagar permit 10
match ip address 110
set metric 100
```

Each route map statement has a "permit" or "deny" action and a sequence number. This route ma shows a permit action and a sequence number of 10. These settings are the defaultsthat is, if no a or sequence number is specified when the route map is configured, the route map will default to a and a sequence number of 10.

The sequence number allows the identification and editing of multiple statements. Consider the configuration steps in Example 14-2.

### Example 14-2. Route map Hagar is modified in this configuration.

```
route-map Hagar 20
 match ip address 111
 set metric 50
route-map Hagar 15
 match ip address 112
 set metric 80
```

Here, a second and third set of route map statements, each with their own match and set statem have been added to route map Hagar. Notice that a sequence number of 20 was configured first a a sequence number of 15. In the final configuration, the IOS has placed statement 15 before 20 e though it was entered later, as shown in Example 14-3.[1]

[1] Notice also that no action was specified in the configuration steps, so the default **permit** appears in the final configuration.

### Example 14-3. IOS places the commands in sequential order.

```
route-map Hagar permit 10
 match ip address 110
 set metric 100
!
route-map Hagar permit 15
 match ip address 112
 set metric 80
!
```

```
route-map Hagar permit 20
 match ip address 111
 set metric 50
```

The sequence numbers also allow for the elimination of individual statements. For example, the statement

```
Linus(config)#no route-map Hagar 15
```

deletes statement 15 and leaves the other statements intact, as shown in Example 14-4.

## Example 14-4. Route-map Hagar after the match/set statements associ with sequence 15 have been removed.

```
route-map Hagar permit 10
 match ip address 110
 set metric 100
!
route-map Hagar permit 20
 match ip address 111
 set metric 50
```

Be careful when editing route maps. In this example, if no route-map Hagar had been typed, wi specifying a sequence number, the entire route map would have been deleted. Likewise, if no seqı numbers had been specified when the additional match and set statements were added, they wou have simply changed statement 10.[2]

[2] Depending upon your IOS version, trying to change a match or set route-map statement without specifying a sequence nun result in an IOS message "% Please specify the entry by its sequence," rather than assuming that sequence number 10 is to b changed.

A packet or route is passed sequentially through route map statements. If a match is made, any s statements are executed and the permit or deny action is executed. As with access lists, processin when a match is made and the specified action is executed; the route or packet is not passed to subsequent statements. Consider the route map in Example 14-5.

## Example 14-5. Route-map Sluggo.

```
route-map Sluggo permit 10
 match ip route-source 1
 set next-hop 192.168.1.5
!
route-map Sluggo permit 20
 match ip route-source 2
 set next-hop 192.168.1.10
!
route-map Sluggo permit 30
 match ip route-source 3
 set next-hop 192.168.1.15
```

If a route does not match statement 10, it will be passed to statement 20. If a match is made at statement 20, the set command will be executed and the route will be permitted. The matched ro not be passed on to statement 30.

The behavior of a "deny" action depends on whether the route map is being used for policy routing redistribution. If a route map is being used for redistribution and a route matches a statement wit deny action, the route will not be redistributed. If the route map is being used for policy routing a packet matches a statement with a deny action, the packet is not policy routed but is passed back normal routing process for forwarding.

Again as with access lists, there must be a default action for the route map to take in the event th route or packet passes through every statement without a match. An implicit deny exists at the er every route map. Routes that pass through a redistribution route map without a match are not redistributed, and packets that pass through a policy route map without a match are sent to the n routing process.

If no match statement is configured under a route map statement, the default action is to match everything.

Each map statement might have multiple match and set statements, as shown in Example 14-6.

Example 14-6. Route map Garfield contains multiple match and set statements for the map statement with sequence number 10.

```
route-map Garfield permit 10
match ip route-source 15
match interface Serial0
set metric-type type-1
set next-hop 10.1.2.3
```

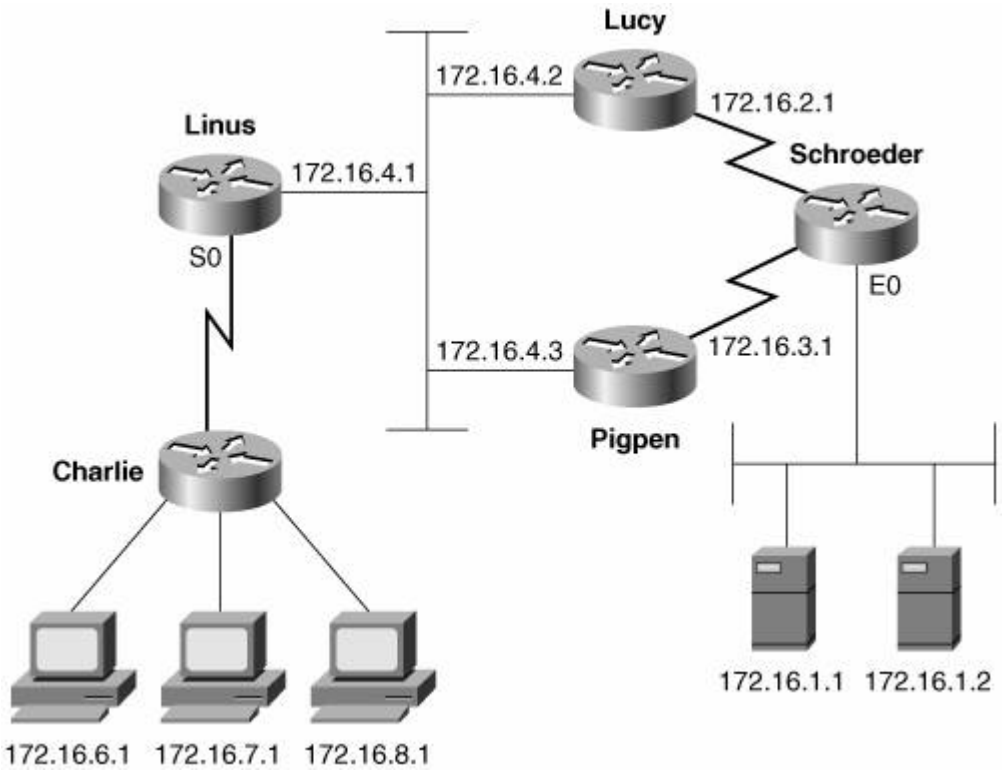
In a case such as this, every match statement must be matched for the set statements to be exe

## Case Study: Policy Routing

Policy routing is defined with the command ip policy route-map. The command is configured on interface and affects incoming packets only.

Suppose a policy were to be implemented on Linus in Figure 14-3 such that traffic from subnet 172.16.6.0/24 would be forwarded to Lucy and traffic from subnet 172.16.7.0/24 would be forwar Pigpen. Linus's configuration is displayed in Example 14-7.

Figure 14-3. Policy routes can be configured at Linus to route some pac through Lucy and other packets through Pigpen.




Example 14-7. Linus's policy routing configuration.

```
interface Serial0
 ip address 172.16.5.1 255.255.255.0
 ip policy route-map Sally
!
access-list 1 permit 172.16.6.0 0.0.0.255
access-list 2 permit 172.16.7.0 0.0.0.255
!
route-map Sally permit 10
 match ip address 1
 set ip next-hop 172.16.4.2
!
route-map Sally permit 15
 match ip address 2
 set ip next-hop 172.16.4.3
```

The policy routing command on S0 sends incoming packets to route map Sally. Statement 10 of rc map Sally uses access list 1 to identify source addresses from subnet 172.16.6.0/24. If a match is the packet is forwarded to Lucy, whose next-hop interface address is 172.16.4.2. If no match is m the packet is sent to statement 15. That statement uses access list 2 to match source addresses fı subnet 172.16.7.0/24. If a match is made at that statement, the packet is forwarded to Pigpen (172.16.4.3). Any packets that do not match statement 15, such as packets sourced from subnet 172.16.8.0/24, are routed normally. Example 14-8 shows the results of the policy route.[3]

[3] Note that the **debug ip packet** command references an access list 5. This access list permits only the subnets connected t Charlie so that uninteresting traffic is not displayed by the debug function.

Example 14-8. The policy route configured on Linus's S0 interface route packets from subnet 172.16.6.0/24 to Lucy (172.16.4.2) and routes pac from subnet 172.16.7.0/24 to Pigpen (172.16.4.3). Packets from subne

172.16.8.0/24, which do not match the policy route, are routed normall
(load balancing between Lucy and Pigpen).

```
Linus#debug ip packet 5
IP packet debugging is on for access list 5
Linus#
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.6.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.6.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.6.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.6.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.8.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.8.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.8.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.8.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forwar
```

Suppose Lucy's Ethernet interface fails. Linus would still attempt to forward packets from 172.16.6
Lucy's IP address. Because a match is made in the route map, even if the specified next-hop inter
were down, no other matches would be attempted nor would the packet be routed normally. To fo
Linus to verify the availability of the next-hop address before attempting to forward the packet, us
command `set ip next-hop verify-availability`. Linus will search its CDP neighbors table to verify
the next-hop address is listed. If it is not, the policy is rejected and the packet is forwarded norma
Example 14-9 shows the output of the commands `debug ip policy` and `debug arp` while Lucy's E
interface is down. Packets are matched and policy routed, even while the router is attempting to A
the next-hop address, without receiving an ARP reply. The packets are dropped.

Example 14-10 shows the output of the `debug ip policy` command with the addition of the `set ip hop verify-availability` command in the IP policy configuration. The example shows Linus's oper
with Lucy's Ethernet interface still down. The packets are matched by the policy, but the policy is r
because the next-hop address is no longer in Linus's CDP table. Since the policy is rejected, the pa
are routed using the normal method.

Example 14-9. *debug ip policy* and *debug arp* on Linus shows packets ar
attempting to be routed according to the configured policy, even if the n
hop specified by the policy is unavailable.

```
Linus#debug arp
ARP packet debugging is on
Linus#debug ip policy
Policy routing debugging is on
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, policy match
IP: route map Sally, item 10, permit
IP: s=172.16.6.1 (Serial0), d=172.16.2.1 (Ethernet0), len 100, policy routed
IP: Serial0 to Ethernet0 172.16.4.2
IP ARP: sent req src 172.16.4.1 0004.c150.e700,
                 dst 172.16.4.2 0000.0000.0000 Ethernet0
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, policy match
IP: route map Sally, item 10, permit
IP: s=172.16.6.1 (Serial0), d=172.16.2.1 (Ethernet0), len 100, policy routed
IP: Serial0 to Ethernet0 172.16.4.2
IP ARP: sent req src 172.16.4.1 0004.c150.e700,
                 dst 172.16.4.2 0000.0000.0000 Ethernet0
IP ARP: creating incomplete entry for IP address: 172.16.4.2 interface Ethernet0
```

```
IP ARP: sent req src 172.16.4.1 0004.c150.e700,
                 dst 172.16.4.2 0000.0000.0000 Ethernet0
IP ARP throttled out the ARP Request for 172.16.4.2
```

Example 14-10. *debug ip policy* on Linus with *set ip next-hop verify-availability* configured, shows the policy is rejected (the next hop is not Linus's CDP neighbor table) and packets are routed normally (not policy routed).

```
Linus#debug ip policy
Policy routing debugging is on

IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy rejected - normal
forwarding
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy rejected - normal
forwarding
```

Example 14-11 shows the output of the same debug command, debug ip policy, after Lucy's Eth interface is backed up. The output shows that the packets are successfully policy routed.

Example 14-11. *debug ip policy* on Linus with *set ip next-hop verify-availability* configured. The packets are policy routed when the next hop verified.

```
Linus#
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, g=172.16.4.2, len 100, FIB policy routed
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, g=172.16.4.2, len 100, FIB policy routed
```

Standard IP access lists are used when policy routing by source address only. To route by both sou and destination, an extended IP access list is used. The configuration in Example 14-12 causes pa from any subnet to host 172.16.1.1 to be forwarded to Lucy, whereas packets from host 172.16.7 host 172.16.1.2 are forwarded to Pigpen. All other packets are routed normally.

Example 14-12. Policy route maps can reference extended IP access-list specify a source and destination address pair to match, as shown in this configuration.

```
interface Serial0
ip address 172.16.5.1 255.255.255.0
ip policy route-map Sally
!
access-list 101 permit ip any host 172.16.1.1
access-list 102 permit ip host 172.16.7.1 host 172.16.1.2
!
route-map Sally permit 10
match ip address 101
set ip next-hop 172.16.4.2
!
route-map Sally permit 15
```

```
match ip address 102
set ip next-hop 172.16.4.3
```

Route map Sally is again used, except the match statements now reference access lists 101 and 1
Example 14-13 shows the results.

### Example 14-13. Packets from any host to host 172.16.1.1 match statem 10 of route map Sally and are forwarded to Lucy. Packets from host 172.16.7.1 to host 172.16.1.2 are forwarded to Pigpen. Packets from an address on subnet 172.16.7.0/24 to host 172.16.1.2 are not matched by Sally and are routed normally.

```
Linus#debug ip packet 5
IP packet debugging is on for access list 5
Linus#
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.7.1 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forwar
IP: s=172.16.7.254 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forwa
IP: s=172.16.7.254 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.2, len 60, forwa
IP: s=172.16.7.254 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forwa
IP: s=172.16.7.254 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.2, len 60, forwa
```

Next, suppose your policy states that FTP traffic from the servers on subnet 172.16.1.0/24 should forwarded to Lucy and that Telnet traffic from the same servers should be forwarded to Pigpen. Th allows the bulk FTP traffic and the bursty, interactive Telnet traffic to be segregated on the two se links from Schroeder. Schroeder will have the configuration in Example 14-14.

### Example 14-14. Schroeder's policy route configuration forwarding FTP a Telnet traffic.

```
interface Ethernet0
ip address 172.16.1.4 255.255.255.0
ip policy route-map Rerun
!
access-list 105 permit tcp 172.16.1.0 0.0.0.255 eq ftp any
access-list 105 permit tcp 172.16.1.0 0.0.0.255 eq ftp-data any
access-list 106 permit tcp 172.16.1.0 0.0.0.255 eq telnet any
!
route-map Rerun permit 10
match ip address 105
set ip next-hop 172.16.2.1
!
route-map Rerun permit 20
match ip address 106
set ip next-hop 172.16.3.1
```

Access lists 105 and 106 are examining not only the source and destination addresses, but also th source port. In Example 14-15, the detail option is used with debug ip packet to allow observati

the packet types being forwarded by Schroeder. An access list 10 limits the displayed packets to tl
from 172.16.1.1 to 172.16.6.1.

### Example 14-15. FTP packets (TCP ports 20 and 21) are being forwarded Lucy, whereas Telnet packets (TCP port 23) with the same source and destination addresses are forwarded to Pigpen. Echo Reply packets (ICM type 0), which do not find a match in the policy route, are routed norma

```
Schroeder#debug ip packet detail 10
IP packet debugging is on (detailed) for access list 10
Schroeder#
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1064, forwa
    TCP src=20, dst=1047, seq=3702770065, ack=591246297, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 64, forwarc
    TCP src=21, dst=1046, seq=3662108731, ack=591205663, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1476, forwa
    TCP src=20, dst=1047, seq=3702771089, ack=591246297, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forwarc
    TCP src=23, dst=1048, seq=3734385279, ack=591277873, win=14332 ACK
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 52, forwarc
    TCP src=23, dst=1048, seq=3734385279, ack=591277873, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forwarc
    TCP src=23, dst=1048, seq=3734385291, ack=591277876, win=14332 ACK
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 60, forwarc
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forwarc
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 60, forwarc
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forwarc
    ICMP type=0, code=0
```

The purpose of segregating bulk and interactive traffic, as demonstrated in the last example, is so the small packets characteristic of interactive traffic do not become delayed by the large packets characteristic of bulk traffic. The problem with the approach in this example is that if many types c must be segregated, the access lists identifying the traffic by destination port might become prohi large.

If the objective is to segregate small packets from large packets, the length of the packet can be matched, as shown in Example 14-16.

### Example 14-16. Schroeder's policy route configuration forwards traffic b on packet length.

```
interface Ethernet0
ip address 172.16.1.4 255.255.255.0 ip policy route-map Woodstock !
route-map Woodstock permit 20
match length 1000 1600
set ip next-hop 172.16.2.1
!
route-map Woodstock permit 30
match length 0 400
set ip next-hop 172.16.3.1
```

Here the `match length` statement specifies a minimum and a maximum packet size. Statement 2

the route map causes all packets between 1000 and 1600 octets in length to be routed across the link to Lucy. Statement 30 causes all packets up to 400 octets in length to be routed across the se to Pigpen. Packets between 400 and 1000 octets are routed normally.

Example 14-17 shows the results of the new route map. Again there are FTP, Telnet, and Echo Re packets from 172.16.1.2 to 172.16.6.1, but now the packets are routed according to their size ins their addresses and ports.

## Example 14-17. Packets of 1000 octets or larger are routed to Lucy, whe packets of 400 octets or less are routed to Pigpen. Any packets betweer and 1000 octets are routed normally.

```
Schroeder#debug ip packet detail 10
IP packet debugging is on (detailed) for access list 10
Schroeder#
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1476, forwa
    TCP src=20, dst=1063, seq=1528444161, ack=601956937, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1476, forwa
    TCP src=20, dst=1063, seq=1528442725, ack=601956937, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1476, forwa
    TCP src=20, dst=1063, seq=1528444161, ack=601956937, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 840, forwar
    TCP src=20, dst=1063, seq=1528445597, ack=601956937, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forward
    TCP src=21, dst=1062, seq=1469372904, ack=601897901, win=14329 ACK
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 54, forward
    TCP src=21, dst=1062, seq=1469372904, ack=601897901, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forward
    TCP src=21, dst=1062, seq=1469372918, ack=601897901, win=14335 ACK FIN
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 44, forward
    TCP src=23, dst=1064, seq=1712116521, ack=602140570, win=14335 ACK SYN
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 43, forward
    TCP src=23, dst=1064, seq=1712116522, ack=602140570, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forward
    TCP src=23, dst=1064, seq=1712116525, ack=602140573, win=14332 ACK
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 52, forward
    TCP src=23, dst=1064, seq=1712116525, ack=602140573, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
```

The policy routes demonstrated so far affect packets entering the router from a particular interfac what about packets generated by the router itself? These can also be policy routed, with the comn local policy route-map. Unlike the ip policy route-map command, which is configured on an interface, this command is configured globally on the router.

To apply the previously demonstrated policy to packets generated by Schroeder, the configuration displayed in Example 14-18.

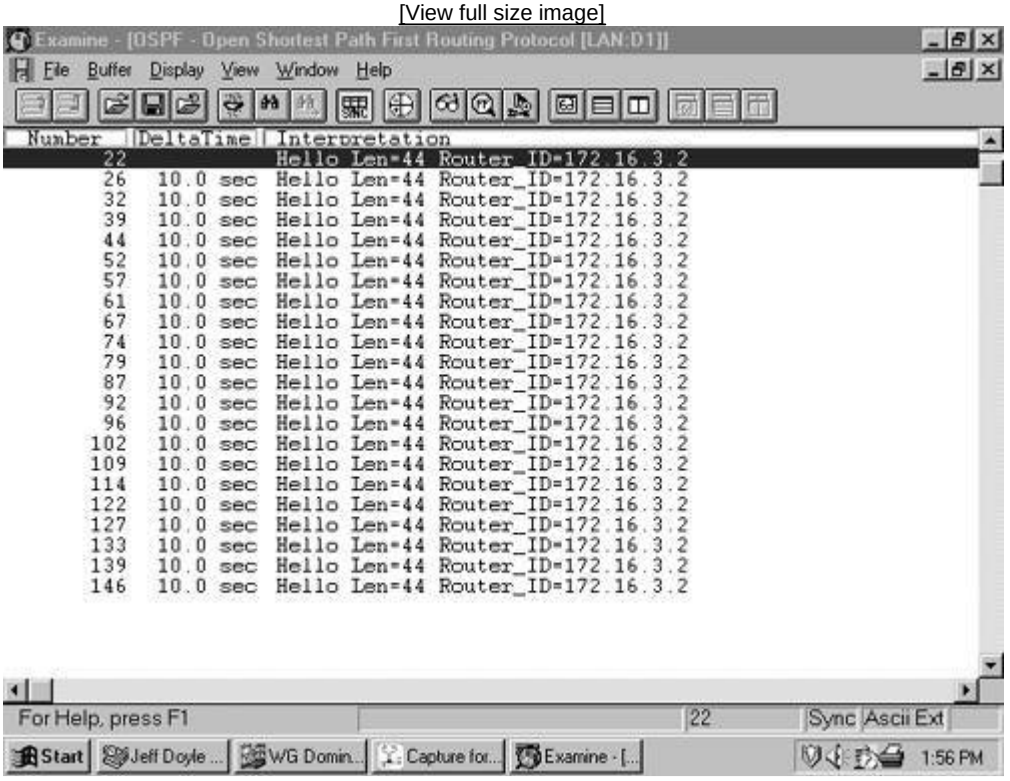
## Example 14-18. Route policy configuration for packets generated by Schroeder.

```
interface Ethernet0
 ip address 172.16.1.4 255.255.255.0
 ip policy route-map Woodstock
!
ip local policy route-map Woodstock
!
access-list 120 permit ip any 172.16.1.0 0.0.0.255
access-list 120 permit ospf any any
!
route-map Woodstock permit 10
 match ip address 120
!
route-map Woodstock permit 20
 match length 1000 1600
 set ip next-hop 172.16.2.1
!
route-map Woodstock permit 30
 match length 0 400
 set ip next-hop 172.16.3.1
```

Of particular interest is statement 10. This statement does not have a set statement, but merely p packets that match access list 120. Access list 120, in turn, permits all packets destined for subne 172.16.1.0/24 and all OSPF packets. Without the first line of the access list, some packets originat Schroeder and destined for subnet 172.16.1.0/24 would be forwarded to the wrong interface by statement 20 or 30. Figure 14-4 shows why the second line of the access list is necessary. The len Schroeder's OSPF Hellos is 44 octets. If statement 10 were not included, the OSPF Hellos would al statement 30 and be forwarded to Pigpen, breaking the adjacency between Lucy and Schroeder. E matching statement 10, the OSPF packets are permitted with no changes and are forwarded norm

Figure 14-4. The length of the OSPF Hello packets is seen in this analy capture.

[View full size image]

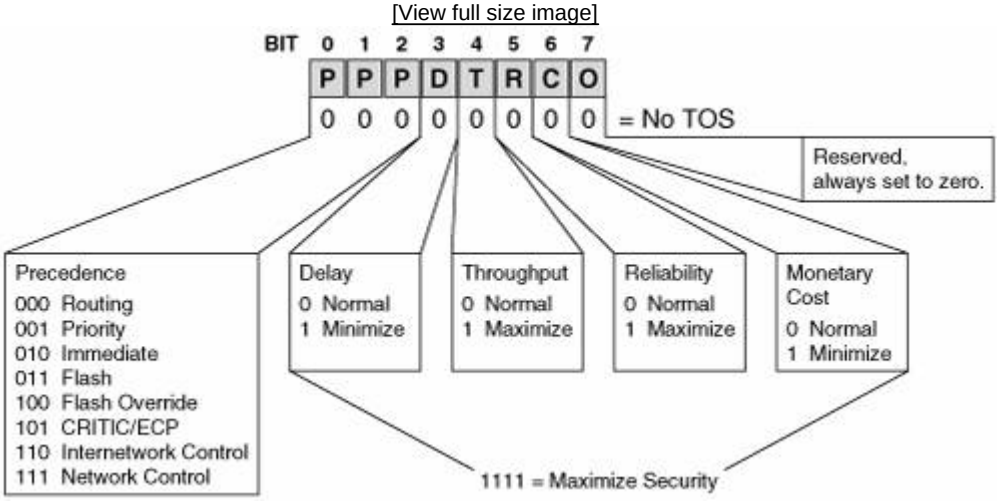


## Case Study: Policy Routing and Quality of Service Routing

Although quality of service (QoS) routing is outside the scope of this volume, it must be noted her policy routing can be an integral part of QoS. Policy routing in conjunction with QoS is done by set Precedence or the Type of Service (TOS) bits of the TOS field in the IP headers of packets as they router's interface. Figure 14-5 shows the bits of the TOS field. Although the TOS bits are seldom u modern networks, the Precedence bits have found new life in QoS applications. The TOS bits are u influence the path a router selects for a packet, whereas the Precedence bits are used to prioritize packets within a router.

Figure 14-5. The Precedence and TOS bits of the Type of Service field o IP header.

[View full size image]




The Precedence bits are set by using the set ip Precedence statement within a route map. The Precedence might be set by specifying the decimal equivalent of the three Precedence bits or by u keywords. Table 14-5 shows the decimal numbers and the keywords that can be used.

Table 14-5. Precedence values and keywords used with the *set ip precedence* command.

| Bits | Number | Keyword |
|---|---|---|
| 000 | 0 | routine |
| 001 | 1 | priority |
| 010 | 2 | immediate |
| 011 | 3 | flash |
| 100 | 4 | flash-override |

| Bits | Number | Keyword |
|---|---|---|
| 101 | 5 | critical |
| 110 | 6 | internet |
| 111 | 7 | network |

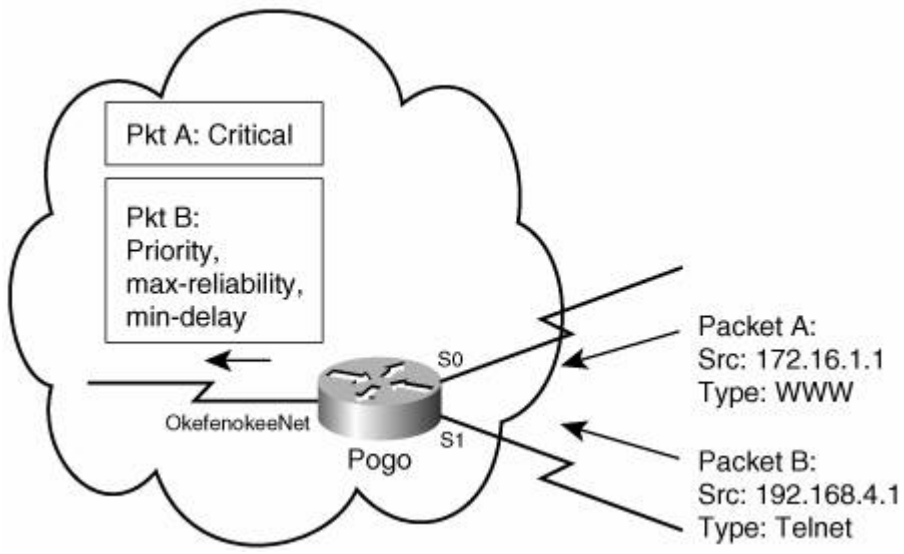
The TOS bits are set by using the set ip tos statement. Like the Precedence statement, the argun the statement might be a number or a keyword, as shown in Table 14-6. Unlike Precedence, you r use a combination of TOS values. For example, specifying a TOS value of 12 (1100b) means *minin delay* and *maximum throughput*. Only a single keyword can be used, so to set a combination of TC values, a number must be specified.

Table 14-6. TOS values and keywords used with the *set ip tos* command.

| Bits | Number (015) | Keyword |
|---|---|---|
| 0000 | 0 | normal |
| 0001 | 1 | min-monetary-cost |
| 0010 | 2 | max-reliability |
| 0100 | 4 | max-throughput |
| 1000 | 8 | min-delay |

Figure 14-6 shows an example of how policy routes can be used for QoS routing.

Figure 14-6. Policy routes can be used to set the Precedence or TOS bi packets entering a network. The routers within the network can then n QoS decisions based on the setting of these bits.




Here, router Pogo is at the "edge" of the Internet OkefenokeeNet. By configuring policy routes on serial links, the Precedence or TOS bits of incoming packets can be changed so that IP traffic is div into several traffic classes. See Example 14-19 for instance.

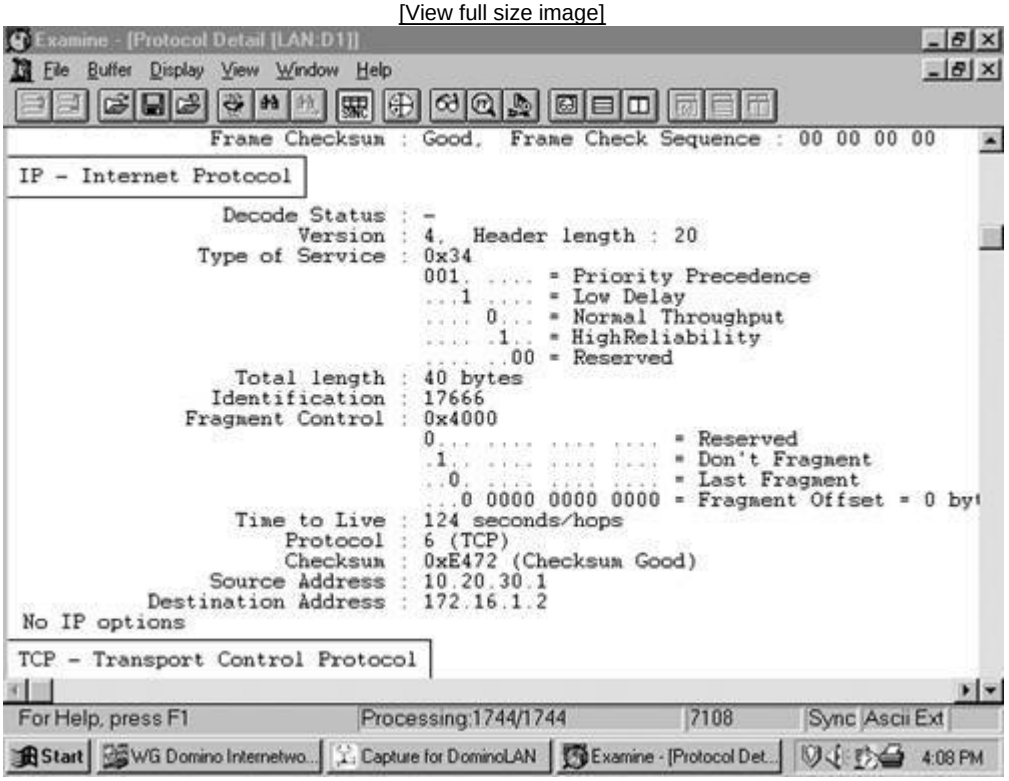
## Example 14-19. Pogo's configuration sets Precedence and TOS bits.

```
interface Serial0
ip address 10.1.18.67 255.255.255.252
ip policy route-map Albert
!
interface Serial1
ip address 10.34.16.83 255.255.255.252
ip policy route-map Albert
!
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 110 permit tcp any eq www any
!
route-map Albert permit 10
match ip address 1 110
set ip precedence critical
!
route-map Albert permit 20
set ip tos 10
set ip precedence priority
```

Statement 10 says that if packets match both access lists 1 and 110, the Precedence will be set to *critical*. Notice that statement 20 has no match statement. This statement will match any packets haven't been matched by statement 10. There are also two set statements under statement 20. T statements will set the TOS bits to *minimum delay* and *maximum reliability* and will set the Precec bits to *priority*. Figure 14-7 shows a capture of a packet from somewhere inside OkefenokeeNet, w has been modified by the route map at Pogo.

## Figure 14-7. Pogo's policy route has set the Precedence bits of this pack priority (001b) and the TOS bits to minimum delay and maximum relial (1010b).

[View full size image]

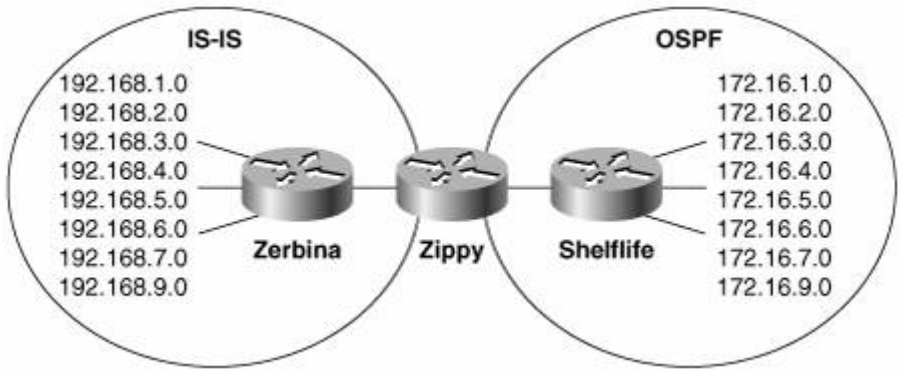



After the Precedence or TOS bits have been set in packets entering the network, the routers withir Internet can make QoS decisions based in part or wholly on the class of service these bits define. I example, priority, custom, or weighted fair queuing might be configured to prioritize traffic accord the Precedence or TOS bits. In some implementations, Precedence can be used with congestion avoidance mechanisms such as Weighted Random Early Detection (WRED). Or a crude Class-of-Se routing can be implemented by configuring access lists that permit or deny packets across certain based on the setting of their Precedence or TOS bits.

## Case Study: Route Maps and Redistribution

A route map can be used with redistribution for both IPv4 and IPv6 by adding a call to the route m the redistribute command. Figure 14-8 shows a network in which IPv4 IS-IS and OSPF routes ar mutually redistributed at router Zippy. Of the network and subnet addresses listed in the illustratic the ones whose third octet is odd-numbered are to be redistributed.

Figure 14-8. The OSPF and IS-IS routes are being mutually redistribu Route maps can be used with the *redistribute* command as simple ro filters, or they can be used to modify characteristics of the redistribu routes.




Zippy's configuration is displayed in Example 14-20.

## Example 14-20. Zippy is configured to redistribute only addresses with a odd numbered third octet.

```
router ospf 1
 redistribute isis level-1 metric 20 subnets route-map Griffy
 network 172.16.10.2 0.0.0.0 area 5
!
router isis
 redistribute ospf 1 metric 25 route-map Toad metric-type internal level-2
 net 47.0001.1234.5678.9056.00
!
access-list 1 permit 192.168.2.0
access-list 1 permit 192.168.4.0
access-list 1 permit 192.168.6.0
access-list 2 permit 172.16.1.0
access-list 2 permit 172.16.3.0
access-list 2 permit 172.16.5.0
access-list 2 permit 172.16.7.0
access-list 2 permit 172.16.9.0
!
route-map Griffy deny 10
 match ip address 1
!
route-map Griffy permit 20
!
route-map Toad permit 10
 match ip address 2
```

Route maps Griffy and Toad perform the same functions, but with different logic. Griffy uses negat logic, identifying the routes that should not be redistributed, and Toad uses positive logic, identifyi routes that should be redistributed.

Statement 10 of Griffy denies any routes that are permitted by access list 1 (the addresses with a third octet). Because the addresses with odd-numbered third octets do not find a match at statem they are passed to statement 20. Statement 20 has no match statement, so the default is to mat everything. Statement 20 has a permit action, so the odd routes are permitted. The result is show Example 14-21.

## Example 14-21. The only destinations within the IS-IS domain that are contained in Shelflife's route table are those with an odd-numbered thir

octet.

```
Shelflife#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
O  E2 192.168.9.0 [110/20] via 172.16.10.2, 00:24:46, Ethernet0
O  E2 192.168.1.0 [110/20] via 172.16.10.2, 00:24:46, Ethernet0
O  E2 192.168.3.0 [110/20] via 172.16.10.2, 00:24:46, Ethernet0
O  E2 192.168.5.0 [110/20] via 172.16.10.2, 00:24:47, Ethernet0
O  E2 192.168.7.0 [110/20] via 172.16.10.2, 00:24:47, Ethernet0
      172.16.0.0 255.255.255.0 is subnetted, 9 subnets
C        172.16.9.0 is directly connected, Serial0
C        172.16.10.0 is directly connected, Ethernet0
O        172.16.4.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O        172.16.5.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O        172.16.6.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O        172.16.7.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O        172.16.1.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O        172.16.2.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O        172.16.3.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
Shelflife#
```

Route map Toad has a single statement that permits routes that have been permitted by access li
(addresses with an odd third octet). The addresses with an even third octet do not find a match at
list 2. The default route map statement when redistributing is to deny all routes, so the addresses
are not matched by access list 2 are not redistributed. Example 14-22 shows the results of route r
Toad.

Example 14-22. The only destinations within the OSPF domain that are contained in Zerbina's route table are those with an odd-numbered thir octet.

```
Zerbina#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
C    192.168.9.0/24 is directly connected, Serial0
C    192.168.10.0/24 is directly connected, Ethernet0
i L1 192.168.1.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.2.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.3.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.4.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.5.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.6.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.7.0/24 [115/15] via 192.168.9.2, Serial0
     172.16.0.0/24 is subnetted, 5 subnets
i L2    172.16.9.0 [115/35] via 192.168.10.2, Ethernet0
i L2    172.16.5.0 [115/35] via 192.168.10.2, Ethernet0
i L2    172.16.7.0 [115/35] via 192.168.10.2, Ethernet0
i L2    172.16.1.0 [115/35] via 192.168.10.2, Ethernet0
i L2    172.16.3.0 [115/35] via 192.168.10.2, Ethernet0
```

```
Zerbina#
```

Other configurations will achieve the same ends. For instance, route map Toad will have the same with the access list in Example 14-23.

### Example 14-23. An alternate configuration for route-map Toad on Zippy

```
access-list 2 deny 172.16.2.0
access-list 2 deny 172.16.4.0
access-list 2 deny 172.16.6.0
access-list 2 permit any
```

Although route maps work fine as simple route filters, their strength lies in their ability to change routes in various ways. Consider the configuration in Example 14-24 of Zippy in Figure 14-8.

### Example 14-24. Zippy's route-map configuration sets the metric type, m and level of certain redistributed routes.

```
router ospf 1
 redistribute isis level-1 metric 20 subnets route-map Griffy
 network 172.16.10.2 0.0.0.0 area 5
!
router isis
 redistribute ospf 1 metric 25 route-map Toad metric-type internal level-2
 net 47.0001.1234.5678.9056.00
!
ip classless
access-list 1 permit 192.168.2.0
access-list 1 permit 192.168.4.0
access-list 1 permit 192.168.6.0
access-list 2 permit 172.16.9.0
access-list 2 permit 172.16.5.0
access-list 2 permit 172.16.7.0
access-list 2 permit 172.16.1.0
access-list 2 permit 172.16.3.0
!
route-map Griffy permit 10
 match ip address 1
 set metric-type type-1
!
route-map Griffy permit 20
!
route-map Toad permit 10
 match ip address 2
 set metric 15
 set level level-1
!
route-map Toad permit 20
```

Statement 10 of route map Griffy permits routes to the addresses in access list 1 and redistributes into OSPF as type 1 external routes. Statement 20 permits all other routes, which will be redistrib with the default external type 2. Example 14-25 shows the results.

Example 14-25. The routes to destinations in the IS-IS domain are E1 if third octet of the address is even and E2 if the third octet is odd.

```
Shelflife#show ip route
Codes:C -connected,S -static,I -IGRP,R -RIP,M -mobile,B -BGP
      D -EIGRP,EX -EIGRP external,O -OSPF,IA -OSPF inter area
      E1 -OSPF external type 1,E2 -OSPF external type 2,E -EGP
      i -IS-IS,L1 -IS-IS level-1,L2 -IS-IS level-2,*-candidate default
Gateway of last resort is not set
O E2 192.168.9.0 [110/20 ] via 172.16.10.2,,00:13:43,Ethernet0
O E2 192.168.1.0 [110/20 ] via 172.16.10.2,,00:13:43,Ethernet0
O E1 192.168.2.0 [110/30 ] via 172.16.10.2,,00:13:43,Ethernet0
O E2 192.168.3.0 [110/20 ] via 172.16.10.2,,00:13:44,Ethernet0
O E1 192.168.4.0 [110/30 ] via 172.16.10.2,,00:13:44,Ethernet0
O E2 192.168.5.0 [110/20 ] via 172.16.10.2,,00:13:44,Ethernet0
O E1 192.168.6.0 [110/30 ] via 172.16.10.2,,00:13:44,Ethernet0
O E2 192.168.7.0 [110/20 ] via 172.16.10.2,,00:13:44,Ethernet0
     172.16.0.0 255.255.255.0 is subnetted,9 subnets
C       172.16.9.0 is directly connected,Serial0
C       172.16.10.0 is directly connected,Ethernet0
O       172.16.4.0 [110/159 ] via 172.16.9.2,,15:49:29,Serial0
O       172.16.5.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.6.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.7.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.1.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.2.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.3.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
Shelflife#
```

Statement 10 of route map Toad permits routes to addresses in access list 2 and redistributes the IS-IS as level 1 routes. Their metric is set to 15. Statement 20 permits all other routes, which will redistributed as level 2 and with a metric of 25, as specified by the redistribute command under IS configuration (see Example 14-26).

Example 14-26. The routes to destinations in the OSPF domain are L2 if third octet of the address is even and L1 if the third octet is odd. The "o are redistributed with a metric of 15, and the "evens" are redistributed a metric of 25 (10 is added for the hop from Zippy to Zerbina).

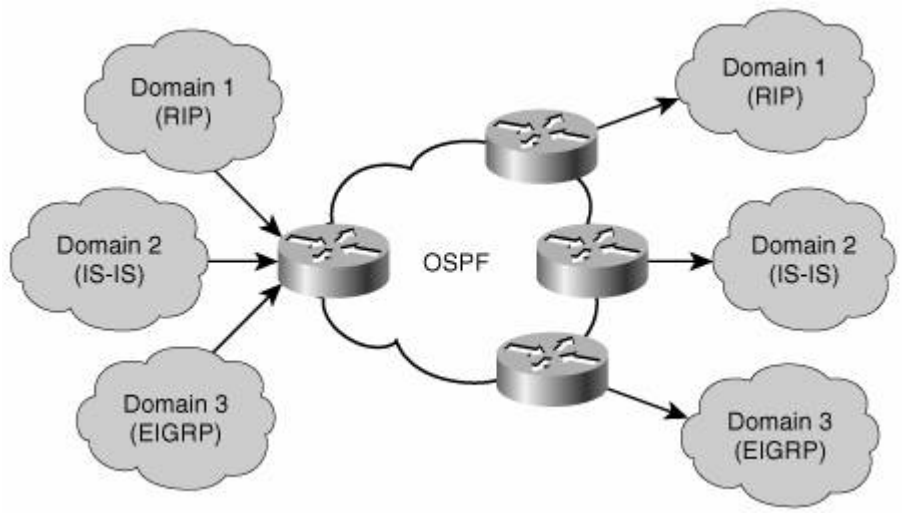
```
Zerbina#show ip route
Codes:C -connected,S -static,I -IGRP,R -RIP,M -mobile,B -BGP
      D -EIGRP,EX -EIGRP external,O -OSPF,IA -OSPF inter area
      N1 -OSPF NSSA external type 1,N2 -OSPF NSSA external type 2
      E1 -OSPF external type 1,E2 -OSPF external type 2,E -EGP
      i -IS-IS,L1 -IS-IS level-1,L2 -IS-IS level-2,*-candidate default
      U -per-user static route,o -ODR
Gateway of last resort is not set
C    192.168.9.0/24 is directly connected,Serial0
C    192.168.10.0/24 is directly connected,Ethernet0
i L1 192.168.1.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.2.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.3.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.4.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.5.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.6.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.7.0/24 [115/15 ] via 192.168.9.2,,Serial0
     172.16.0.0/24 is subnetted,8 subnets
```

```
i L1    172.16.9.0 [115/25 ] via 192.168.10.2,,Ethernet0
i L2    172.16.4.0 [115/35 ] via 192.168.10.2,,Ethernet0
i L1    172.16.5.0 [115/25 ] via 192.168.10.2,,Ethernet0
i L2    172.16.6.0 [115/35 ] via 192.168.10.2,,Ethernet0
i L1    172.16.7.0 [115/25 ] via 192.168.10.2,,Ethernet0
i L1    172.16.1.0 [115/25 ] via 192.168.10.2,,Ethernet0
i L2    172.16.2.0 [115/35 ] via 192.168.10.2,,Ethernet0
i L1    172.16.3.0 [115/25 ] via 192.168.10.2,,Ethernet0
Zerbina#
```

## Case Study: Route Tagging

Figure 14-9 shows a situation in which routes from several routing domains, each running a separ routing protocol, are being redistributed into a single transit domain running OSPF. On the other s the OSPF cloud, the routes must be redistributed back into their respective domains. Route filters used at the egress points from the OSPF cloud into each domain to permit only the routes that bel that domain. However, if each domain has many routes or if the routes within the domain change frequently, the route filters can become difficult to manage.

Figure 14-9. Routes from each of the three domains on the left are redistributed into a transit network running OSPF. On the right, the ro for each domain must be redistributed back into their original domai




Another way of handling this problem is to tag the routes at their ingress points to the OSPF trans domain with a tag that is unique to each domain. At the egress points, the routes can be redistribu their tags instead of by specific addresses. The routing protocol of the transit network does not necessarily use the tag, but merely conveys it to and from its external networks. RIPv2, EIGRP, Integrated IS-IS, and OSPF all support route tags. BGP also supports route tags. Tags are not sup by RIPv1. A case study in this section shows how a transit network running OSPF can use the rout

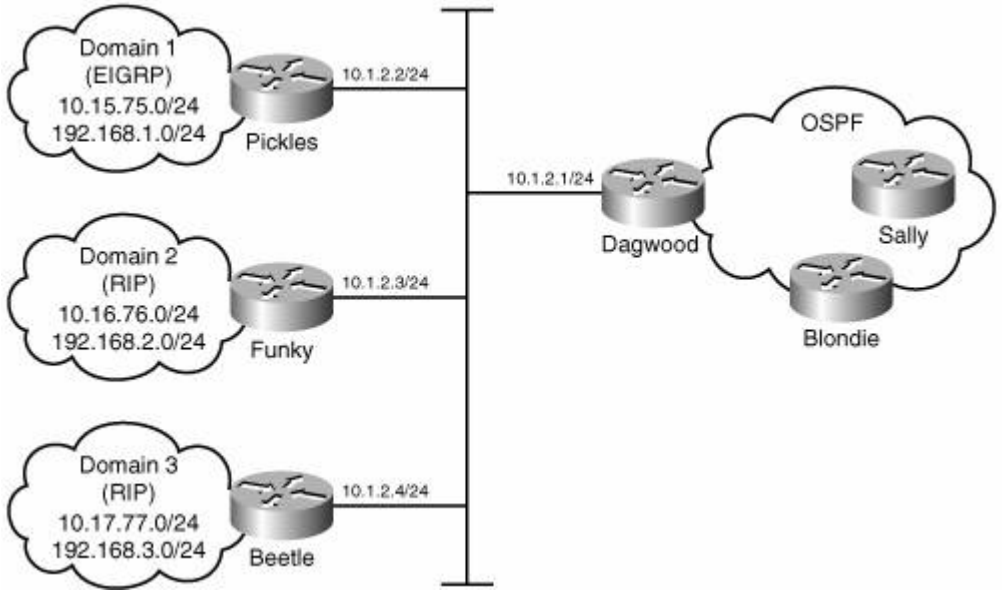
A re-examination of the packet formats in Chapter 6, "RIPv2, RIPng, and Classless Routing," Chap "Enhanced Interior Gateway Routing Protocol (EIGRP)," Chapter 8, "OSPFv2," and Chapter 10, "Integrated IS-IS" show that RIPv2 messages support 16-bit tags, IS-IS Inter-Domain Routing Pr Information TLVs support 16-bit tags, and EIGRP external route TLVs and OSPF type 5 LSAs suppo bit tags. These tags are expressed as decimal numbers, so tags carried by RIPv2 and IS-IS will be between 0 and 65,535, and tags carried by EIGRP and OSPF will be between 0 and 4,294,967,295

In Figure 14-10, router Dagwood is accepting routes from three different routing domains and

redistributing them into a domain running OSPF. The objective here is to tag the routes from each domain so that their source domain might be identified within the OSPF cloud. Routes from domai have a tag of 1, domain 2 will have a tag of 2, and so on.

Figure 14-10. Dagwood is configured to tag the routes from each of the routing domains as they are redistributed into OSPF.

[View full size image]




Dagwood's configuration is displayed in Example 14-27.

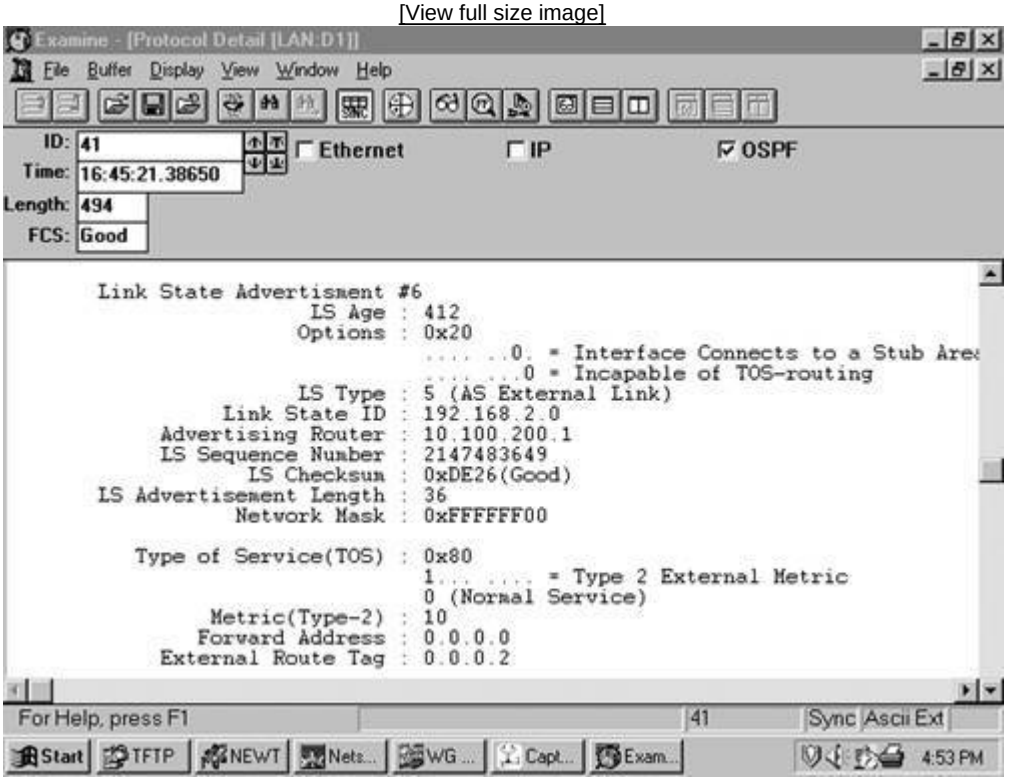
Example 14-27. Dagwood is configured to tag routes as they are redistributed into OSPF from RIP and EIGRP.

```
router ospf 1
  redistribute eigrp 1 metric 10 subnets tag 1
  redistribute rip metric 10 subnets route-map Dithers
  network 10.100.200.1 0.0.0.0 area 0
!
router rip
  network 10.0.0.0
!
router eigrp 1
  network 10.0.0.0
!
access-list 1 permit 10.1.2.3
access-list 2 permit 10.1.2.4
!
route-map Dithers permit 10
  match ip route-source 1
  set tag 2
!
route-map Dithers permit 20
  match ip route-source 2
  set tag 3
```

First, notice the redistribute eigrp command under OSPF. Dagwood is accepting routes from onl EIGRP domain, so the tag can be set to 1 directly on the redistribute command. However, routes being learned from two RIP domains. Here a route map is needed. Route map Dithers sets the tag RIP routes to either 2 or 3, depending on whether the route was learned from Funky (10.1.2.3) or (10.1.2.4). Figure 14-11 shows an LSA advertising one of the RIP-learned routes, with the route t to 2.

Figure 14-11. This type 5 LSA is advertising network 192.168.2.0, which domain 2, within the OSPF domain. The route tag is shown on the last

[View full size image]




The route tags can also be observed in the OSPF link state database (see Example 14-28).

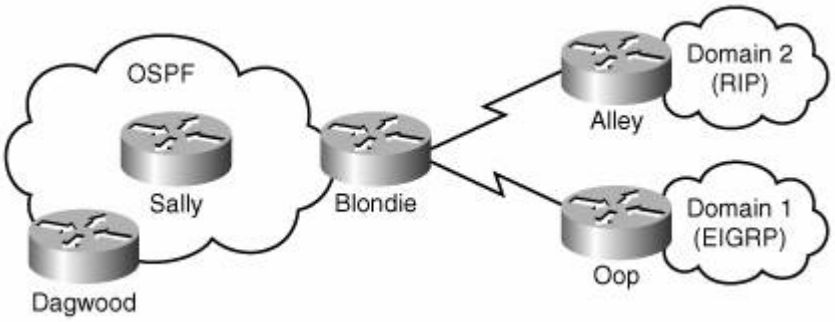
Example 14-28. The OSPF link state database indicates the tags that we set for each of the external routes by Dagwood's redistribution processe

```
Blondie#show ip ospf database
       OSPF Router with ID (10.100.200.2)(Process ID 1)
               Router Link States (Area 0)
Link ID         ADV Router      Age     Seq#        Checksum   Link count
10.100.200.3    10.100.200.3    671     0x80000003  0x00A137   4
10.100.200.2    10.100.200.2    39      0x80000002  0x6FF5   3
10.100.200.1    10.100.200.1    40      0x80000033  0x33E1   3
               Net Link States (Area 0)
Link ID         ADV Router      Age     Seq#        Checksum
10.100.200.1    10.100.200.1    40      0x80000001  0xB0A7
               AS External Link States
Link ID         ADV Router     Age      Seq#        Checksum  Tag
192.168.2.0     10.100.200.1   641      0x80000028  0x904D   2
10.17.77.0      10.100.200.1   642      0x80000028  0xC817   3
192.168.3.0     10.100.200.1   642      0x80000028  0x9744   3
```

```
10.15.75.0      10.100.200.1  642      0x80000028  0xD213   1
10.1.2.0        10.100.200.1  642      0x80000028  0xA19B   1
10.16.76.0      10.100.200.1  642      0x80000028  0xCD15   2
192.168.1.0     10.100.200.1  644      0x80000028  0x8956   1
10.100.200.0    10.100.200.1  644      0x80000028  0x6EA4   1
Blondie#
```

In Figure 14-12, Blondie must redistribute only domain 2 routes to Alley and only domain 1 routes Oop. Because the routes were tagged as they entered the OSPF transit domain, this is easily done Blondie's configuration is shown in Example 14-29.

Figure 14-12. Blondie is using route maps to redistribute routes accordi their route tag.




Example 14-29. Blondie is configured to use route maps to redistribute routes with tag 1 to EIGRP and tag 2 to RIP.

```
router ospf 1
  network 10.100.200.2 0.0.0.0 area 0
!
router rip
  redistribute ospf 1 match external 2 route-map Daisy
  passive-interface Ethernet0
  passive-interface Serial1
  network 10.0.0.0
  default-metric 5
!
router eigrp 1
  redistribute ospf 1 match external 2 route-map Herb
  passive-interface Ethernet0
  passive-interface Serial0
  network 10.0.0.0
  default-metric 10000 1000 255 1 1500
!
route-map Daisy permit 10
  match tag 2
!
route-map Herb permit 10
  match tag 1
```

Example 14-30 shows the resulting routes at Alley and Oop. One drawback to the use of route tag filter routes is that there is no way to filter routes by interface. For example, if Blondie had to send

to both domain 2 and domain 3, which both run RIP, route maps could not be configured to send s routes to one RIP process and other routes to another RIP process. The routes would have to be fi by address with distribute-list commands.

### Example 14-30. The route tables of Alley and Oop in Figure 14-12 show results of the redistribution configuration at Blondie.

```
Alley#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

   10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
C    10.1.3.0/24 is directly connected, Serial0
R    10.1.5.4/30 [120/1] via 10.1.3.1, 00:00:25, Serial0
R    10.1.4.0/24 [120/1] via 10.1.3.1, 00:00:25, Serial0
R    10.16.76.0/24 [120/5] via 10.1.3.1, 00:00:25, Serial0
R    10.100.200.2/32 [120/1] via 10.1.3.1, 00:00:25, Serial0
R 192.168.2.0/24 [120/5] via 10.1.3.1, 00:00:25, Serial0
Alley#
```

---

```
Oop#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks
D       10.1.3.0/24 [90/2681856] via 10.1.4.1, 00:21:36, Serial0
D EX    10.1.2.0/24 [170/2425856] via 10.1.4.1, 00:08:22, Serial0
D       10.1.5.4/30 [90/2681856] via 10.1.4.1, 00:22:40, Serial0
C       10.1.4.0/24 is directly connected, Serial0
D EX    10.15.75.0/24 [170/2425856] via 10.1.4.1, 00:04:56, Serial0
D       10.100.200.2/32 [90/2297856] via 10.1.4.1, 00:22:40, Serial0
D EX 192.168.1.0/24 [170/2425856] via 10.1.4.1, 00:04:56, Serial0
Oop#
```

## Case Study: Filtering Tagged Routes Out of OSPF Route Table

The network running OSPF in Figure 14-10 and Figure 14-12 is a transit network. If devices within transit area do not need to send packets to any of the other domains, there is no need to maintair addresses of those domains in the OSPF route tables. The tags that were added to the routes, alor distribute lists and route maps, can be applied to the OSPF routers to prevent the addresses from added to the route tables, while not affecting the entries in the link-state database.

Sally, a router wholly within the OSPF domain, has been modified with the configuration in Exampl

31.

## Example 14-31. Sally's configuration uses tags to filter routes from the route table.

```
router ospf 1
 network 10.100.200.1 0.0.0.0 area 0
 network 10.1.5.0 0.0.0.255 area 0
 distribute-list route-map Charlie in
!
route-map Charlie deny 10
 match tag 1 2 3
!
route-map Charlie permit 20
```

The route map Charlie denies addresses that are marked with tags 1, 2 or 3. These addresses are omitted from the IP route table by the distribute-list in command. Any other address is added to route table, permitted by the route map sequence 20. The distribute list is not applied to Blondie a Dagwood, the edge routers that are performing redistribution. If an address is not in the route tab even if it is in the OSPF LSA database, it will not be redistributed into another routing protocol. Ex 14-32 shows Sally's IP route table and OSPF LSA database.

## Example 14-32. The OSPF addresses marked with tags are filtered out o IP route table. The addresses still exist in the OSPF LSA database.

```
Sally#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
C      10.1.5.4/30 is directly connected, Serial0/0.2
C      10.1.5.0/30 is directly connected, Serial0/0.1
O      10.100.200.2/32 [110/65] via 10.1.5.6, 00:17:24, Serial0/0.2
C      10.100.200.3/32 is directly connected, Loopback0
O      10.100.200.1/32 [110/65] via 10.1.5.1, 00:17:24, Serial0/0.1
```

---

```
Sally#show ip ospf database

            OSPF Router with ID (10.100.200.3) (Process ID 1)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#     Checksum Link count
10.100.200.1    10.100.200.1    1183        0x80000004 0x003756 3
10.100.200.2    10.100.200.2    1181        0x80000004 0x00E1A1 3
10.100.200.3    10.100.200.3    1177        0x8000000A 0x00933E 4

                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
```

```
10.1.2.0        10.100.200.1    94          0x80000003 0x00EB76 1
10.15.75.0      10.100.200.1    603         0x80000002 0x001FEC 1
10.16.76.0      10.100.200.1    346         0x80000002 0x001AEE 2
10.16.77.0      10.100.200.1    353         0x80000002 0x001FEE 2
192.168.1.0     10.100.200.1    603         0x80000002 0x00D530 1
192.168.2.0     10.100.200.1    346         0x80000002 0x00DC27 2
192.168.3.0     10.100.200.1    353         0x80000002 0x00E427 2
Sally#
```

None of the tagged routes, tag 1, 2, or 3, exist in Sally's IP route table.

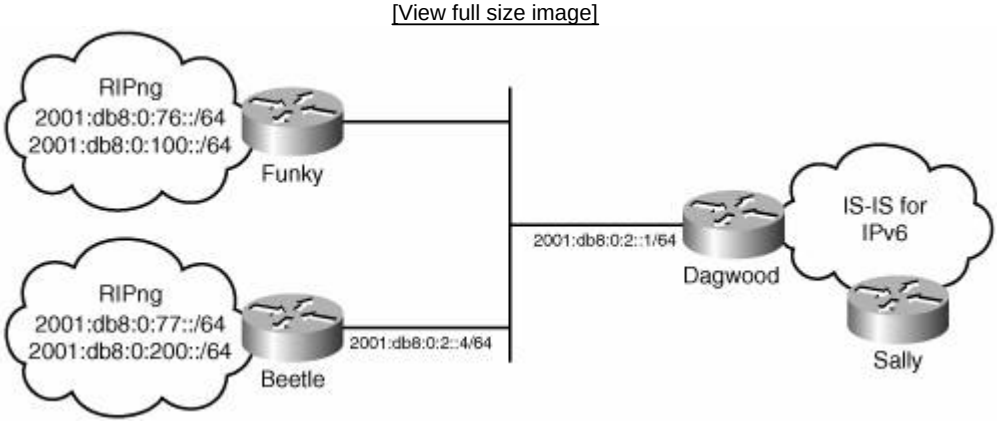
## Case Study: IPv6 Redistribution with Route Maps

IPv6 routing protocols also support redistribution of routes using route maps. The configuration is identical to IPv4.

Figure 14-13 shows the addition of IPv6 addresses and routing protocols to the network shown in 14-10. Funky and Beetle are each running RIPng. Dagwood is redistributing IPv6 prefixes between and IS-IS. Only IPv6 prefixes from Beetle are redistributed into IS-IS, and prefix 2001:db8:0:77: set with a metric of 10 while prefix 2001:db8:0:200::/64 has a metric of 100.

Figure 14-13. IPv6 has been added to the network in Figure 14-10. IP prefixes are redistributed between RIPng and IS-IS.

[View full size image]




Dagwood's configuration is displayed in Example 14-33.

Example 14-33. Dagwood's IPv6 configuration.

```
Interface ethernet 0/0
 ipv6 address 2001:db8:0:2::1/64
 ipv6 rip domain3 enable
!
interface serial 0/0.1 point-to-point
 ipv6 address 2001:db8:0:5::1/64
 ipv6 router isis
!
ipv6 router rip domain3
!
```

```
router isis
 net 00.0001.0000.5678.ef01.00
 Address-family ipv6
  Redistribute rip domain3 route-map Beetlefilter
!
route-map Beetlefilter permit 10
 match ipv6 route-source prefix-list 1
 match ipv6 address prefix-list 3
 set metric 10
!
route-map Beetlefilter permit 20
 match ipv6 route-source prefix-list 1
 match ipv6 address prefix-list 2
 set metric 100
!
ipv6 prefix-list 1 permit 2001:db8:0:2::4/128
ipv6 prefix-list 2 permit 2001:db8:0:200::/64
ipv6 prefix-list 3 permit 2001:db8:0:77::/64
```

IPv6 prefix-lists are used to match the source of route information or to match specific addresses redistribution. Statement 10 of Beetlefilter specifies that the route source must equal Beetle's IPv6 address and the prefix must equal 2001:db8:0:77::/64 for the metric to be set to 10. If the route or the prefix does not match, statement 20 is executed. If the source is Beetle and the prefix is 2001:db8:0:200::/64, the metric is set to 100. If the source is not Beetle, or the prefix is not one ones specified, the route is not redistributed.

Sally's IS-IS database (Example 14-34) shows the redistributed prefixes.

## Example 14-34. The redistributed routes are seen in Sally's IS-IS datab

```
Sally#show isis database detail Dagwood-00.00

IS-IS Level-1 LSP Dagwood.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Dagwood.00-00         0x00000004   0xCA8B        938               0/0/0
  Area Address: 00.0001
  NLPID:        0x8E
  Hostname: Dagwood
  IPv6 Address: 2001:DB8:0:5::1
  Metric: 10         IPv6 2001:DB8:0:5::/64
  Metric: 10         IS Sally.00

IS-IS Level-2 LSP Dagwood.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime     ATT/P/OL
Dagwood.00-00         0x00000008   0x28D7        1089             0/0/0
  Area Address: 00.0001
  NLPID:        0x8E
  Hostname: Dagwood
```

PREV NEXT

# Looking Ahead

This chapter concludes this book's in-depth look at routing TCP/IP with respect to interior gateway protocols. If you are preparing to become a CCIE, you certainly will want to know this book's topics thoroughly before taking the exam. Use the end-of-chapter questions and exercises to test your level of understanding and preparedness. And if you haven't studied routing TCP/IP with respect to Exterior Gateway Protocols, doing so is a next logical step in your preparation.

# Summary Table: Chapter 14 Command Review

| Command | Description |
|---|---|
| access-list *access-list-number* {deny \| permit} *source* [*source-wildcard*] | Defines a line of a standard IP access list |
| access-list *access-list-number* {deny \| permit} *protocol source source-wildcard destination destination-wildcard* [precedence *precedence*] [tos *tos*] [log] | Defines a line of an extended IP access list |
| ip local policy route-map *map-tag* | Defines a policy route for packets originated by the router itself |
| ip policy route-map map-tag | Defines a policy route for packets transiting the router |
| match interface *type number* [...*type number*] | Matches routes that have their next hop out one of the interfaces specified |
| match ip address{*access-list-number*\| *name*} [...*access-list-number*\| *name*] | Matches routes that have a destination address specified by one of the access lists |
| match ip next-hop{*access-list-number*\| *name*} [...*access-list-number*\| *name*] | Matches routes that have a next-hop router address specified by one of the access lists |
| match ip route-source{*access-list-number*\| *name*} [...*access-list-number*\| *name*] | Matches routes that have been advertised by routers at the addresses specified in the access lists |
| match ipv6 address prefix-list{*name*} | Matches routes that have a destination address specified by the prefix-list |
| match ipv6 next-hop prefix-list{*name*} | Matches routes that have a next-hop router address specified by the prefix-lists |
| match ipv6 route-source prefix-list{*name*} | Matches routes that have been advertised by routers at the addresses specified in the prefix-list |
| match length *min max* | Matches the level 3 length of a packet |
| match metric *metric-value* | Matches routes with the specified metric |
| match route-type{internal\| external[type-1\| type-2] \| level-1\| level-2} | Matches OSPF, EIGRP, or IS-IS routes of the specified type |
| match tag *tag-value* [...*tag-value*] | Matches routes with the specified tags |
| ipv6 prefix-list *list-name* [seq *seq-number*] {deny *ipv6-prefix / prefix-length* \| permit *ipv6-prefix / prefix-length* \| description *text*} [ge *ge-value*] [le *le-value*] | Defines an IPv6 prefix list |
| redistribute *protocol*[*process-id*] {level-1\| level-1-2\| level-2} [metric *metric-value*] [metric-type *type-value*] [match{internal\| external 1\| external 2}] [tag *tag-value*] [route-map *map-tag*] [weight *weight*] [subnets] | Configures redistribution into a routing protocol and specifies the source of the redistributed routes |

| Command | Description |
|---|---|
| set level{level-1\| level-2\| level-1-2\| stub-area \| backbone} | Sets the IS-IS level or the OSPF area into which a matched route is to be redistributed |
| set default interface *type number* [...*type number*] | Sets the outgoing interface for matched packets when there is no explicit route to the destination |
| set interface *type number* [...*type number*] | Sets the outgoing interface for matched packets when there is an explicit route to the destination |
| set ip default next-hop *ip-address* [...*ip-address*] | Sets the next-hop router address for matched packets when there is no explicit route to the destination |
| set ip next-hop *ip-address* [...*ip-address*] | Sets the next-hop router address for matched packets when there is an explicit route to the destination |
| set ip precedence *precedence* | Sets the precedence bits in the Type of Service field of matched IP packets |
| set ip tos *type-of-service* | Sets the TOS bits in the Type of Service field of matched packets |
| set metric{*metric-value*\| *bandwidth delay reliability loading mtu*} | Sets the metric value for a matched route |
| set metric-type{internal\| external\| type-1\| type-2} | Sets the metric type for a matched route being redistributed into IS-IS or OSPF |
| set next-hop *next-hop* | Sets the next-hop router address for a matched route |
| set tag *tag-value* | Sets a tag value for a matched route |

PREV NEXT

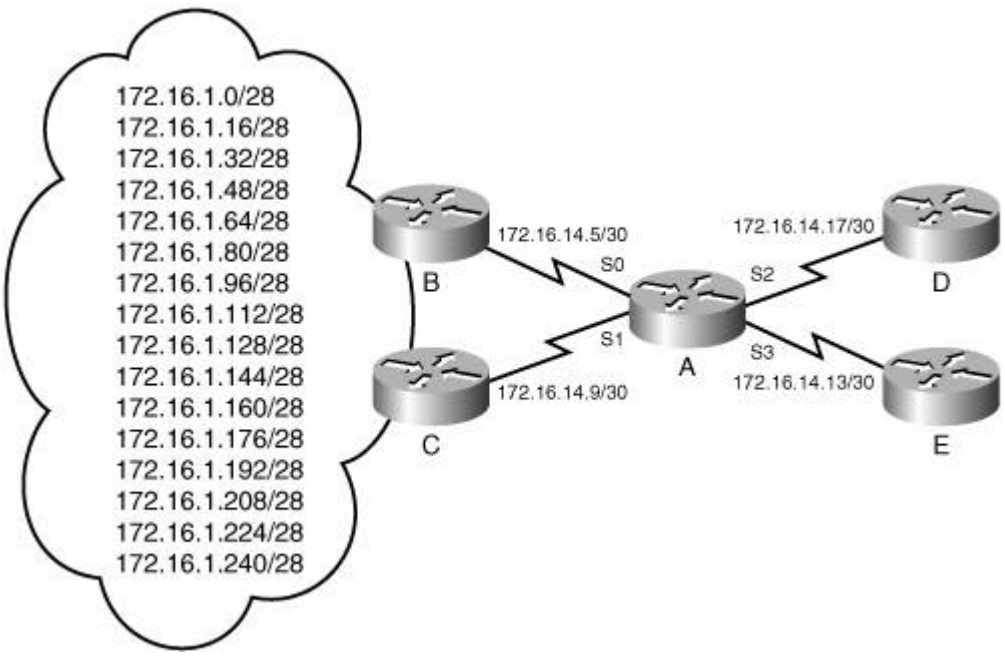
# Review Questions

1. How are route maps similar to access lists? How are they different?
2. What are policy routes?
3. What are route tags?
4. In what way do route tags affect routing protocols?

# Configuration Exercises

1 Configure policy routes for Router A in Figure 14-14 that forward packets from subnets 172.16.1.0/28 through 172.16.1.112/28 to Router D and forward packets from subnets 172.16.1.128/28 through 172.16.1.240/28 to Router E.

Figure 14-14. The network for Configuration Exercises 1 through 3.



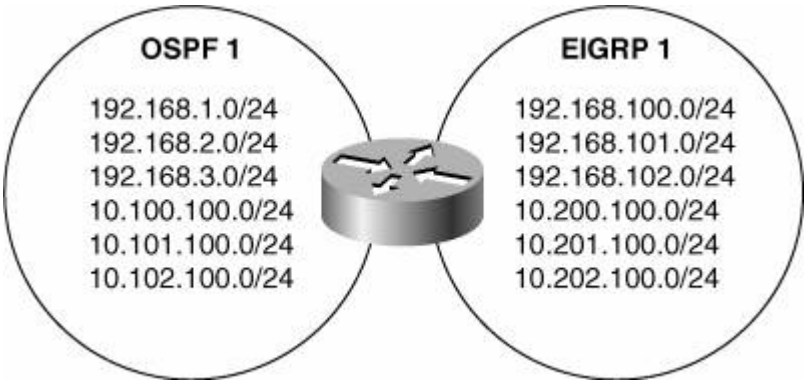

2 Configure policy routes for Router A in Figure 14-14 so that packets from subnets 172.16.1.64/28 through 172.16.1.112/28 are forwarded to Router D if they are received from Router C. If packets from the same subnets are received from Router B, forward them to Router E. All other packets should be forwarded normally.

3 Configure policy routes for Router A in Figure 14-14 that will forward any packets destined for subnets 172.16.1.0/28 through 172.16.1.240/28, sourced from an SMTP port, to Router C. Route any other UDP packets destined for the same subnets to Router B. No other packets should be forwarded to Routers C or B by either the policy routes or the normal routing protocol.

4 The OSPF and EIGRP configuration for the router in Figure 14-15 is

```
router eigrp 1
network 192.168.100.0
!
router ospf 1
network 192.168.1.0 0.0.0.255 area 16
```

Figure 14-15. The router for Configuration Exercises 4 and 5.




Configure the router to redistribute internal EIGRP routes into OSPF as E1 routes with a metric of 10 and to redistribute external EIGRP routes into OSPF as E2 routes with a metric of 50. Of the networks and subnets shown in the EIGRP domain, all should be redistributed except 10.201.100.0/24.

5 Configure the router in Figure 14-15 to redistribute internal OSPF routes into EIGRP with a lower delay than external OSPF routes. Allow only the three Class C networks shown in the OSPF domain to be redistributed.

PREV NEXT

# Troubleshooting Exercise

1 Given the following configuration:

```
interface TokenRingl
ip address 192.168.15.254 255.255.255.0
ip policy route-map Ex1
!
access-list 1 permit 192.168.0.0 0.0.255.255
access-list 101 permit host 192.168.10.5 any eq telnet
!
route-map Ex1 permit 5
match ip address 1
set ip next-hop 192.168.16.254
!
route-map Ex1 permit 10
match ip address 101
set ip next-hop 192.168.17.254
```

The intention is to policy route all packets whose source address prefix is 192.168.0.0 to 192.168.255.255. The exception is that packets originating from the Telnet port of host 192.168.10.5 should be forwarded to 192.168.17.254. There are two errors in this configuration that are preventing the policy route from working correctly. What are they?

# Part IV: Appendixes

Appendix A Tutorial: Working with Binary and Hex

Appendix B Tutorial: Access Lists

Appendix C CCIE Preparation Tips

Appendix D Answers to Review Questions

Appendix E Solutions to Configuration Exercises

Appendix F Solutions to Troubleshooting Exercises

# Appendix A. Tutorial: Working with Binary and Hex

The best way to gain an understanding of binary and hexadecimal numbering is to begin by examining the decimal numbering system. The decimal system is a base 10 numbering system. (The root *deci* means ten.) *Base 10* means that there are 10 digits with which to represent numbers: 0 through 9. Most likely, we work in base 10 because our ancient ancestors began counting their cattle and childrenand enemieson their fingers (in fact, the word *digit* means finger).

The use of place values allows the representation of large numbers with a few digits, such as the 10 decimal digits. The place values of all numbering systems begin at the right, with the base raised to the power of 0. Reading to the left, each place value is the base raised to a power that is one more than the power of the previous place value:

$$B^4 \quad B^3 \quad B^2 \quad B^1 \quad B^0$$

In base 10, the first five place values are

$$10^4 \quad 10^3 \quad 10^2 \quad 10^1 \quad 10^0$$

The first two place values are easy to calculate for any base. Any number raised to the power of 0 is 1; so $10^0 = 1$. Any number raised to the power of 1 is simply that number; so $10^1 = 10$. The third place value is easy to calculate. Simply multiply the second place value by the base. In fact, each place value can be calculated by multiplying the previous place value by the base. So, given the five place values above,

$$10^0 = 1$$
$$10^1 = 1 \times 10 = 10$$
$$10^2 = 10 \times 10 = 100$$
$$10^3 = 100 \times 10 = 1000$$
$$10^4 = 1000 \times 10 = 10{,}000$$

So, the first five place values of the base 10 numbering system are

$$10{,}000 \quad 1000 \quad 100 \quad 10 \quad 1$$

Reading a number such as 57,258 in terms of place values means there are five quantities of 10,000, seven quantities of 1000, two quantities of 100, five quantities of 10, and eight quantities of 1. That is,

$$5 \times 10{,}000 = 50{,}000$$
$$7 \times 1000 = 7000$$
$$2 \times 100 = 200$$
$$5 \times 10 = 50$$
$$8 \times 1 = 8$$

Adding these individual results together, the result is 50,000 + 7000 + 200 + 50 + 8 = 57,258.

All of us are so acquainted with working in base 10 that we seldom think of breaking a number

down into its place values. However, this technique is essential to decipher numbers in other bases.

PREV NEXT

# Working with Binary Numbers

Computers are, at the most fundamental level, simply a collection of electrical switches. Numbers and characters are represented by the positions of these switches. Because a switch has only two positions, on or off, it uses a binary, or base 2, numbering system. (The root *bi* means two.) A base 2 system has only two digits: 0 and 1.

Computers usually group these digits into eight place values, known as a *byte* or an *octet*. The eight place values are

$$2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

The place values are calculated as follows:

$$\begin{aligned}
2^0 &= 1 \\
2^1 &= 1 \times 2 = 2 \\
2^2 &= 2 \times 2 = 4 \\
2^3 &= 4 \times 2 = 8 \\
2^4 &= 8 \times 2 = 16 \\
2^5 &= 16 \times 2 = 32 \\
2^6 &= 32 \times 2 = 64 \\
2^7 &= 64 \times 2 = 128
\end{aligned}$$

So, the place values of a binary octet are

128 64 32 16 8 4 2 1

Thus, the binary octet 10010111 can be read as follows:

$$\begin{aligned}
&1 \times 128 = 128 \\
&0 \times 64 = 0 \\
&0 \times 32 = 0 \\
&1 \times 16 = 16 \\
&0 \times 8 = 0 \\
&1 \times 4 = 4 \\
&1 \times 2 = 2 \\
&1 \times 1 = 1 \\
&\text{or } 128 + 16 + 4 + 2 + 1 = 151
\end{aligned}$$

Working in binary is easy because for every place value there is either one quantity of that value or none of that value. For another example, 11101001 = 128 + 64 + 32 + 8 + 1 = 233.

Where converting binary to decimal is a matter of adding the place values, converting from decimal to binary is a matter of subtracting place values. To convert the decimal number 178 to binary, for instance, begin by subtracting the highest base 2 place value possible from the number:

1. 178 is greater than 128, so we know there is a 1 at that place value: 178 128 = 50.
2. 50 is less than 64, so there is a 0 at that place value.

3. 50 is greater than 32, so there is a 1 at that place value: 50 32 = 18.
4. 18 is greater than 16, so there is a 1 at that place value: 18 16 = 2.
5. 2 is less than 8, so there is a 0 at that place value.
6. 2 is less than 4, so there is a 0 at that place value.
7. 2 is equal to 2, so there is a 1 at that place value: 2 2 = 0.
8. 0 is less than 1, so there is a 0 at that place value.

Putting the results of all these steps together, 178 is 10110010 in binary.

Another example might be helpful. Given 110,

1. 110 is less than 128, so there is a 0 at that place value.
2. 110 is greater than 64, so there is a 1 at that place value: 110 64 = 46.
3. 46 is greater than 32, so there is a 1 at that place value: 46 32 = 14.
4. 14 is less than 16, so there is a 0 at that place value.
5. 14 is greater than 8, so there is a 1 at that place value: 14 8 = 6.
6. 6 is greater than 4, so there is a 1 at that place value: 6 4 = 2.
7. There is a 1 at the 2 place value: 2 2 = 0.
8. 0 is less than 1, so there is a 0 at that place value.

Therefore, 110 is 01101110 in binary.

PREV NEXT

# Working with Hexadecimal Numbers

Writing out binary octets isn't much fun. For people who must work with such numbers frequently, a briefer notation is welcome. One possible notation is to have a single character for every possible octet; however, there are $2^8$ = 256 different combinations of eight bits, so a single-character representation of all octets would require 256 digits, or a base 256 numbering system.

Life is much easier if an octet is viewed as two groups of four bits. For instance, 11010011 can be viewed as 1101 and 0011. There are $2^4$ = 16 possible combinations of four bits, so with a base 16, or hexadecimal, numbering system, an octet can be represented with two digits. (The root *hex* means six, and *deci* means ten.) Table A-1 shows the hexadecimal digits and their decimal and binary equivalents.

Table A-1. Hex, decimal, and binary equivalents.

| Hex | Decimal | Binary |
|---|---|---|
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| A | 10 | 1010 |
| B | 11 | 1011 |
| C | 12 | 1100 |
| D | 13 | 1101 |
| E | 14 | 1110 |
| F | 15 | 1111 |

Because the first 10 characters of the decimal and the hexadecimal numbering system are the same, it is customary to precede a hex number with a *0x*, or follow it with an *h*, to distinguish it from a decimal number. For example, the hex number 25 would be written as 0x25 or as 25h.

This book uses the 0x convention.

After working with binary for only a short while, it is easy to determine a decimal equivalent of a 4-bit binary number in your head. It is also easy to convert a decimal digit to a hex digit in your head. Therefore, converting a binary octet to hex is easily done in three steps:

1. Divide the octet into two 4-bit binary numbers.
2. Convert each 4-bit number to decimal.
3. Write each decimal number in its hex equivalent.

For example, to convert 11010011 to hex,

1. 11010011 becomes 1101 and 0011.
2. 1101 = 8 + 4 + 1 = 13, and 0011 = 2 + 1 = 3.
3. 13 = 0xD, and 3 = 0x3.

Therefore, 11010011 in hex is 0xD3.

Converting from hex to binary is a simple matter of working the three steps backward. For example, to convert 0x7B to binary,

1. 0 x 7 = 7, and 0xB = 11.

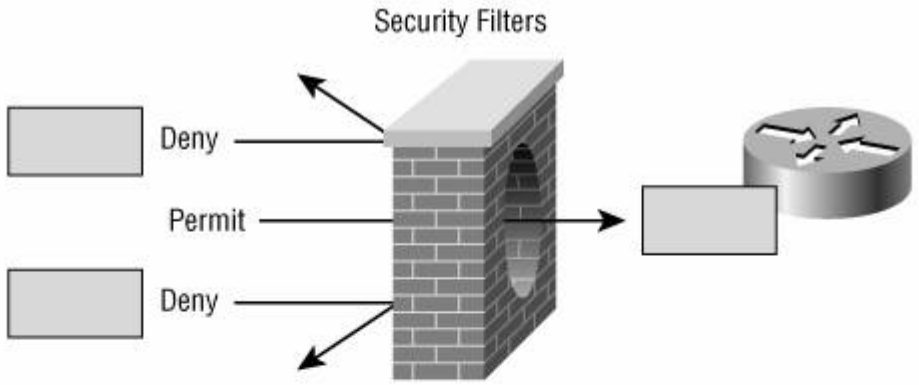
PREV NEXT

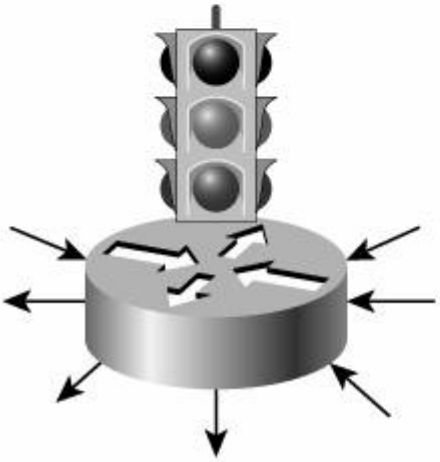
# Appendix B. Tutorial: Access Lists

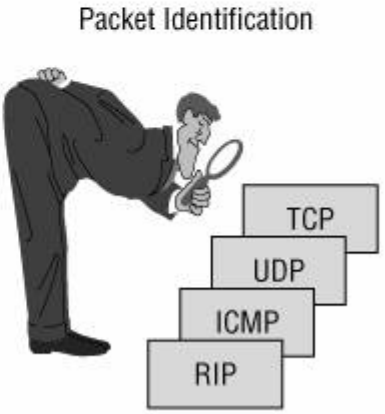
Access lists are probably misnamed these days. As the name implies, the original intention of an access list was to permit or deny access of packets into, out of, or through a router. Access lists have become powerful tools for controlling the behavior of packets and frames. Their use falls into three categories (see Figure B-1):

- *Security filters* protect the integrity of the router and the networks to which they are passing traffic. Typically, security filters permit the passage of a few, well-understood packets and deny the passage of everything else.
- *Traffic filters* prevent unnecessary packets from passing onto limited-bandwidth links. These filters look and behave much like security filters, but the logic is generally inverse: Traffic filters deny the passage of a few unwanted packets and permit everything else.
- *Packet identification* is required for many tools available on Cisco routers, such as dialer lists, route filters, route maps, and queuing lists. The tools must be able to identify certain packets to function properly. Access lists might be linked to these and other tools to provide this packet identification function.

Figure B-1. Access lists are used as security filters, as traffic filters, and for packet identification.


Security Filters
Deny
Permit
Deny


Traffic Filters


Packet Identification
TCP
UDP
ICMP
RIP

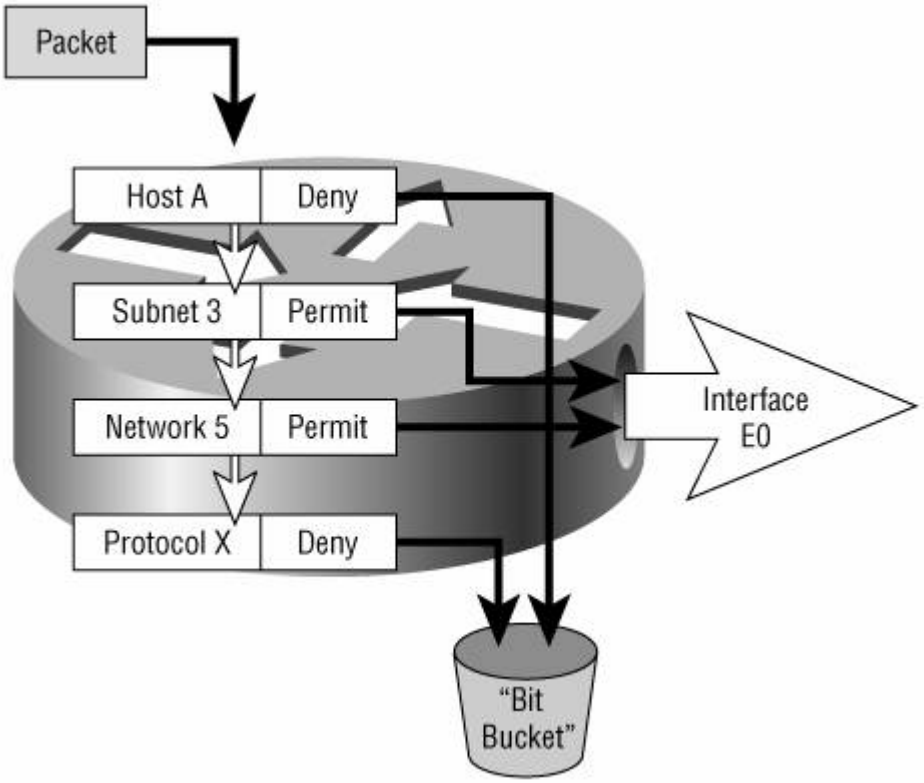
PREV NEXT

# Access List Basics

An access list is a sequential series of filters. Each filter comprises some sort of matching criteria and an action. The action is always either permit or deny. The matching criteria might be as simple as a source address; alternatively, they might be a more complex combination of source and destination addresses, protocol types, ports or sockets, and specifications of the state of certain flags, such as the TCP ACK bit.

A packet is "dropped into" the top of the stack of filters. (See Figure B-2.) At each filter, the matching criteria are applied. If a match occurs, the specified permit or deny action is executed. If a match does not occur, the packet "drops down" to the next filter in the stack, and the matching process is applied again.

Figure B-2. An access list is a sequential list of filters, each of which defines a matching criterion and an action.




In Figure B-2, a permit means that the packet will be allowed to exit on interface E0; a deny means that the packet will be dropped. For instance, a packet with a source address of HOST A will be dropped at the first filter. Suppose the packet's source address is Host D of Subnet 2 of Network 5. The first filter specifies a match criteria of Host A, so the packet will not match and will drop to the second layer. The second filter specifies Subnet 3again, no match. The packet drops to the third filter, which specifies Network 5. This matches; the action at layer three is permit, so the packet is allowed to exit interface E0.

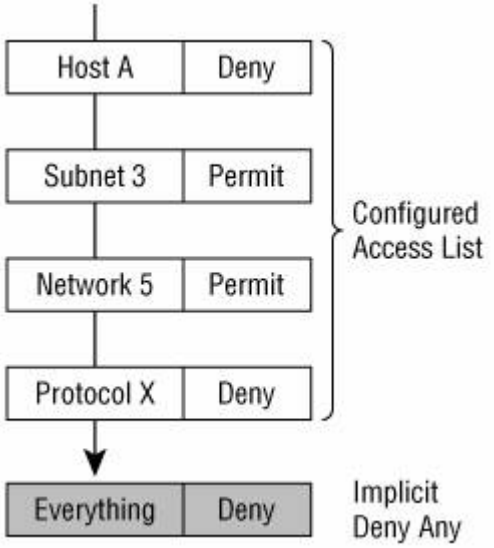
## Implicit Deny Any

What happens if a packet drops through all the filters and a match never occurs? The router must know what to do with a packet in this situation; that is, there must be a default action. The default action can be either to permit all packets that don't match or to deny them. Cisco chose to deny them: Any packet that is referred to an access list and does not find a match is automatically dropped.

This approach is the correct engineering choice, particularly if the access list is being used for security. It is better to drop some packets that shouldn't have been dropped than to permit packets you inadvertently neglected to filter.

This last filter is called an implicit deny any (Figure B-3). As the name implies, the line does not show up in any access list you build. It's simply a default action, and it exists at the end of any and all access lists.

Figure B-3. All access lists end with an implicit deny any, which discards all packets that do not match a line in the list.



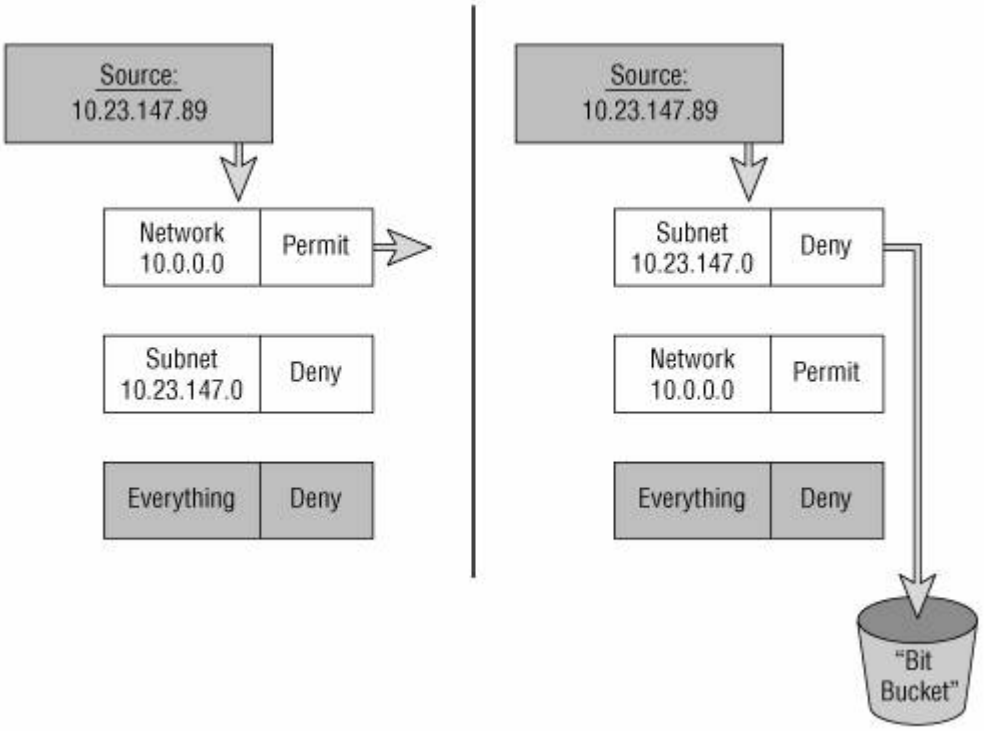

This default can be overridden by making the last line of the list an explicit permit any. The implication here is that packets dropping through all the other filters will match the permit any before they get to the default deny any; therefore, all packets not matching anything else will be permittednothing will ever reach the implicit deny.

## Sequentiality

Access lists are executed sequentially, from the top down. This concept is important: Perhaps, the most common cause of malfunctioning access lists is putting the individual filtering lines in the wrong sequence. The first match encountered in the sequenced access list is always taken. After the first match is made, the rest of the access list is ignored.

In Figure B-4, subnet 10.23.147.0/24 should be denied and the rest of network 10.0.0.0 should be permitted. The list on the left is out of sequence; network 10.0.0.0, including its subnet 10.23.147.0, will match the first line and will be permitted. Packets with the subnet to be denied will never reach the second line.

Figure B-4. If the individual filter layers of an access list are not configured in the correct sequence, the access list will not function correctly.




The list on the right is correct. Subnet 10.23.147.0 matches the first line and is denied, whereas all other subnets of 10.0.0.0 drop to the next line and are permitted.

## Access List Types

The actual configuration lines for the access list shown graphically at the right of Figure B-4 are displayed in Example B-1.

Example B-1. The access list configuration for sequence 2 of Figure B-4 shows one line for each filter layer.

```
access-list 9 deny 10.23.147.0 0.0.0.255
access-list 9 permit 10.0.0.0 0.255.255.255
```

Every filter layer of an access list is represented by one configuration line. The various components of an access list line are discussed shortly, but for now, notice the number 9 in both lines. This number is the access list number, and it serves two purposes:

- It links all the lines of this list together and makes the list distinct from any others that might exist in the router's configuration file. (It is common to have several access lists on a single router.)
- The router must have a way to distinguish the access list type. Cisco IOS Software has access lists for IP, IPX, AppleTalk, DEC, NetBIOS, bridging, and many other protocols.

Further, many of these protocols have multiple access list types. The access list number tells the router what type of list it is.

Access list types can be identified by either a number or a name. Table B-1 shows some of the numbered access list types and the range of access list numbers available for each. For example, as shown in the table, access-list 1010 is identifying IPX SAPs because the number is between 1000 and 1099.

### Table B-1. Cisco access list numbers

| Access List Type | Range |
|---|---|
| Standard IP | 199, 13001999 |
| Extended IP | 100199, 20002699 |
| Ethernet type code | 200299 |
| Ethernet address | 700799 |
| Transparent bridging (protocol type) | 200299 |
| Transparent bridging (vendor code) | 700799 |
| Extended transparent bridging | 11001199 |
| DECnet and extended DECnet | 300399 |
| XNS | 400499 |
| Extended XNS | 500599 |
| AppleTalk | 600699 |
| Source-route bridging (protocol type) | 200299 |
| Source-route bridging (vendor code) | 700799 |
| Standard IPX | 800899 |
| Extended IPX | 900999 |
| IPX SAP | 10001099 |
| NLSP route summary | 12001299 |
| Standard VINES | 199 |
| Extended VINES | 100199 |
| Simple VINES | 200299 |

Within a range, access list numbers do not need to follow any particular sequence. That is, the first standard IP list on a router does not need to be 1, the second 2, and so on. They can be any number between 1 and 99, or 1300 and 1999, just so each list is uniquely numbered on a single router.

Also, notice that some number ranges are the same for different protocols: Ethernet Type Code,

Source Route Bridging, and Simple VINES, for instance. In these cases, the router differentiates between access list types by the format of the access list lines themselves.

The following access list types can be identified by names instead of numbers:

- Apollo domain
- Standard IP
- Extended IP
- ISO CLNS
- Source-route bridging NetBIOS
- Standard IPX
- Extended IPX
- IPX Sap
- IPX NetBIOS
- NLSP route summary

An example of an access list named Boo, identifying IPX NetBIOS, is displayed in Example B-2.

### Example B-2. The access list named Boo denies various NetBIOS devices.

```
netbios access-list host Boo deny Atticus
netbios access-list host Boo deny Scout
netbios access-list host Boo deny Jem
netbios access-list host Boo permit *
```

Note that although standard and extended IP access lists normally are numbered, they can also be named access lists. This convention is supported in IOS 11.2 and later. In some environments, a router might be configured with a large number of IP lists. By using names instead of numbers, individual lists might be more easily identified.

Named IP access lists currently can be used only with packet and route filters. Refer to the Cisco configuration guides for more information.

## Editing Access Lists

Anyone who has edited an access list longer than a few lines from the console will tell you that this process can be an exercise in frustration. Before 12.2(14), there was no way, from the console, to add a line to the middle of the list. All new lines were added to the bottom. And if you had typed a mistake and tried to eliminate a particular line by typing, for instance,

```
no access-list 101 permit tcp 10.2.5.4 0.0.0.255 192.168.3.0 0.0.0.255 eq 25
```

this line, and all of access list 101, would have been deleted!

A far more convenient technique is to cut and paste the list to the notepad of your PC, or upload the configuration to a TFTP server, and do the editing from there. When finished, the new access

list can be loaded back into the router. A word of caution, however: All new lines are added to the bottom of an access list. Always add no access-list #, where # is the number of the list you're editing, to the beginning of the edited list. Example B-3 shows a sample.

### Example B-3. *no access-list* is added to the beginning of an access list that is created and edited on a PC or server, so the access list is created anew each time it is loaded into the router.

```
no access-list 5
access-list 5 permit 172.16.5.4 0.0.0.0
access-list 5 permit 172.16.12.0 0.0.0.255
access-list 5 deny 172.16.0.0 0.0.255.255
access-list 5 permit any
```

The line no access-list 5 deletes the old list 5 from the configuration file before adding the new one. If you omit this step, the new list is simply added onto the end of the old one.

The command show access-list displays currently configured lists, as Example B-4 demonstrates.

### Example B-4. *show access-list* displays configured access lists on the router.

```
Router#show access-list 5
Standard IP access list 5
    10 permit 172.16.5.4
    20 permit 172.16.12.0, wildcard bits 0.0.0.255
    30 deny 172.16.0.0, wildcard bits 0.0.255.255
    40 permit any
Router#
```

Notice the numbers before each access list entry. These are sequence numbers. Sequence numbers are automatically added to access-list entries, as of IOS 12.2(14)S. The sequence numbers allow you to insert an entry into the middle or top of the list. If you don't specify a sequence number, the first entry will be assigned number 10, and the sequence number of every subsequent entry will be incremented by 10. When a router is reloaded, the sequence numbers are reset, to 10, 20, 30, and so on. Sequence numbers also allow you to delete specific entries from an access list.

The entry in access list 5 that permits all hosts on subnet 172.16.12.0 can be replaced with an entry that permits all hosts on subnet 172.16.20.0 with the configuration in Example B-5.

### Example B-5. Access lists can be modified by replacing entries or adding entries using sequence numbers.

```
ip access-list standard 5
no 20
20 permit 172.16.20.0 0.0.0.255
```

PREV NEXT

# Standard IP Access Lists

There are two ways to enter access lists. One format of a standard access list line is

**access-list** *access-list-number* {**deny** | **permit**} *source*[*source-wildcard*]

The other way to configure the access list is to enter a global access-list command, which takes you into access-list configuration mode. In the access-list configuration mode, packets are permitted or denied, sequence numbers are specified and remarks are made:

**ip access-list standard** {*access-list-number* | *name*}

This puts you into the access-list configuration mode. Further configuration options for standard IP access lists are

[*sequence-number*] {{{**deny** | **permit**} *source* [*source-wildcard*]} | {**remark** *up-to-100-characters-of-a-remark*}}

This command specifies the access list number, which according to Table B-1 is between 1 and 99 and between 1300 and 1999; the action (permit or deny); a source IP address; and the wildcard (or inverse) mask. Example B-6 shows a standard IP access list.

### Example B-6. Standard access list 1 permits and denies various hosts and subnet addresses.

```
access-list 1 permit 172.22.30.6 0.0.0.0
access-list 1 permit 172.22.30.95 0.0.0.0
access-list 1 deny 172.22.30.0 0.0.0.255
access-list 1 permit 172.22.0.0 0.0.31.255
access-list 1 deny 172.22.0.0 0.0.255.255
access-list 1 permit 0.0.0.0 255.255.255.255
```

The first two lines of the example permit passage of packets whose source addresses belong to tw specific hosts, 172.22.30.6 and 172.22.30.95. This seems quite obvious from looking at the lines, although the inverse mask of 0.0.0.0 might not make sense yet. The third line denies all other hosts on subnet 172.22.30.0. Again, it's fairly intuitive. The purpose of the fourth line is not so obvious. It permits all hosts with addresses in the range of 172.22.0.1 to 172.22.31.255. The inverse mask is what allows the specification of this range of addresses with a single line. The fifth line denies all other subnets of the Class B network 172.22.0.0, and the last line permits all other addresses.

The alternative way to configure the same list is displayed in Example B-7.

### Example B-7. The same standard IP access list as shown in example B-6 is written here using the access list configuration mode on the router.

```
ip access-list standard 1
 10 permit 172.22.30.6 0.0.0.0
```

```
 15 permit 172.22.30.95 0.0.0.0
 20 deny 172.22.30.0 0.0.0.255
 permit 172.22.0.0 0.0.31.255
 deny 172.22.0.0 0.0.255.255
 permit 0.0.0.0 255.255.255.255
```

The sequence numbers of the first three entries are specified. The fourth, fifth, and sixth entries are automatically assigned a sequence number 10 greater than the previous entry, or 30, 40, and 50. A new statement can be added between two entries simply by specifying a sequence number that falls between the sequence number of the entry above and below the desired location. Example B-8 shows a sample of this.

### Example B-8. A new entry is added to the middle of a standard IP access list using sequence numbers.

```
ip access-list standard 1
 17 permit 172.22.30.100 0.0.0.0
```

Example B-8 adds the new entry after the entry that permits 172.22.30.95 and before the entry that denies the rest of the subnet, deny 172.22.30.0 0.0.0.255.

This entry can simply be deleted as well. Example B-9 deletes the entry with sequence number 17

### Example B-9. An entry is deleted from the middle of a standard IP access list using sequence numbers.

```
ip access-list standard 1
 no 17
```

Comments can be added to access lists before or after any entry to make understanding the list easier in the future. The access list configurations in Example B-10 and Example B-11 contain remarks.

### Example B-10. Remarks are added to a standard IP access list.

```
access-list 1 remark permit the 2 management hosts
access-list 1 permit 172.22.30.6 0.0.0.0
access-list 1 permit 172.22.30.95 0.0.0.0
access-list 1 remark deny everyone else on the subnet
access-list 1 deny 172.22.30.0 0.0.0.255
access-list 1 permit 172.22.0.0 0.0.31.255
access-list 1 deny 172.22.0.0 0.0.255.255
access-list 1 permit 0.0.0.0 255.255.255.255
```

### Example B-11. Remarks are added to a standard IP access list using the router's access-list configuration mode.

```
ip access-list standard 1
 remark permit the 2 management hosts
 10 permit 172.22.30.6 0.0.0.0
 15 permit 172.22.30.95 0.0.0.0
```

```
remark deny everyone else on the subnet
20 deny 172.22.30.0 0.0.0.255
permit 172.22.0.0 0.0.31.255
deny 172.22.0.0 0.0.255.255
permit 0.0.0.0 255.255.255.255
```

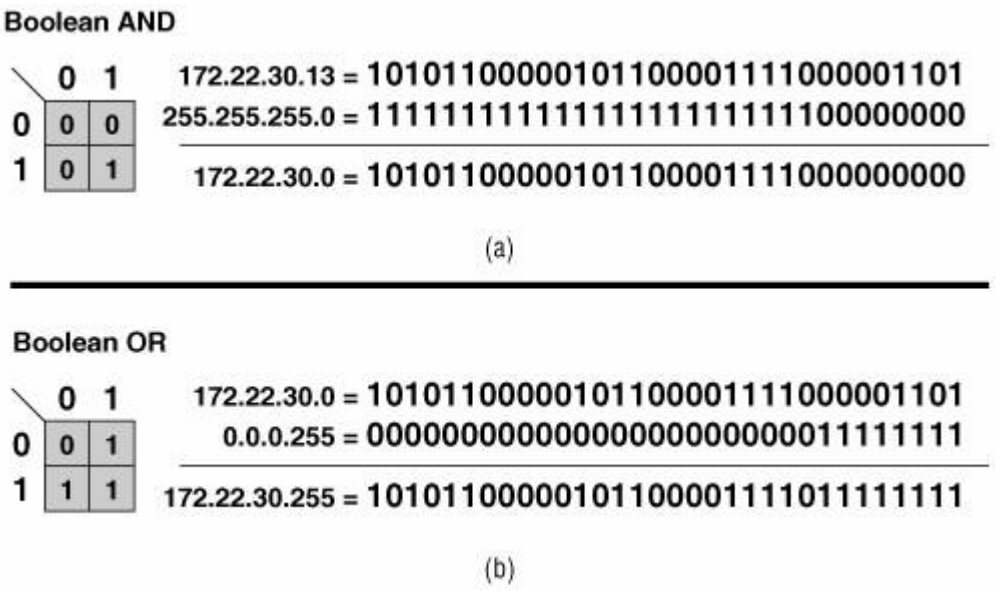
Example B-10 and Example B-11 are two ways to configure the same lists. Remarks don't have an functional affect, but they can make a complicated access list a little more friendly to future readers.

To fully understand the functionality of this access list, you need to understand inverse masks.

Recall how IP address masks function: To derive a network or subnet address from a host address a one is set in the mask corresponding to each bit of the network address, and a zero is set for each bit of the host address. A Boolean AND is performed on each bit, and the result is the networ or subnet number. Figure B-5(a) includes a truth table for the AND function; in English, the function states the following:

> Compare two bits. The result is one if and only if both bits are one.

Figure B-5. Truth tables and examples of a Boolean AND (a) and a Boolean OR (b).




A Boolean OR is the inverse of this function, as its truth table in Figure B-5(b) shows:

> Compare two bits. The result is zero if and only if both bits are zero.

An *inverse mask* (Cisco prefers the term *wildcard mask*) sets a zero for each bit of the address tha should be exactly matched and a one for each bit where anything will matchthe one bits are frequently referred to as "don't care" bits. The inverse mask is then ORed with the address.

Notice the result of the OR example in Figure B-5(b), 172.22.30.255. In IP terms, this result means "all host addresses on subnet 172.22.30.0." Any specific address from 172.22.30.0 will match this address/inverse mask combination.

Figure B-6 shows two shortcuts that might be used when writing standard IP access lists. Figure B 6(a) shows an inverse mask of all zeros to indicate that all 32 bits of the address in question must

match 172.22.30.6 exactly. The default mask for a standard IP access list is 0.0.0.0. So, the alternative statement shown, with no mask specified, is the same as the first statement. Note that this default does not apply to extended IP access lists, which are covered in the following section.

Figure B-6. Two shortcuts can be used when writing standard IP access lists.

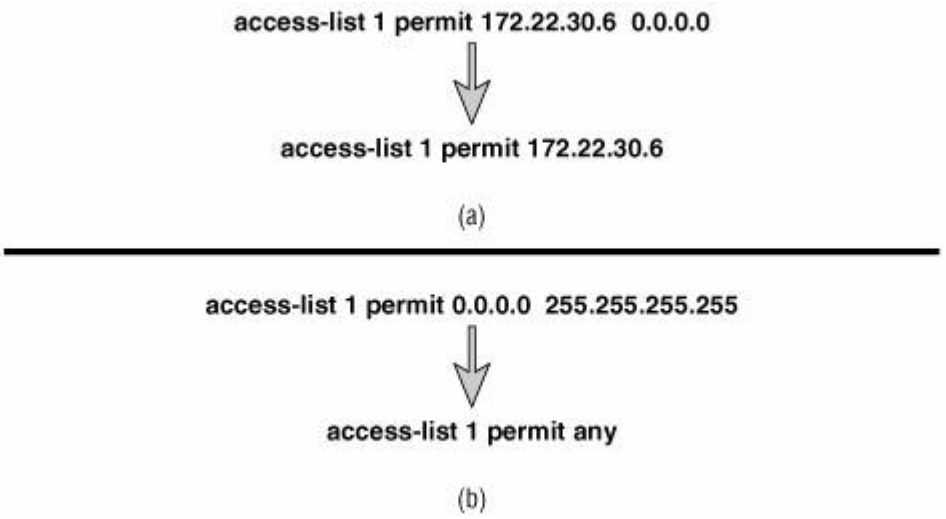



Figure B-6(b) shows the permit anything address/inverse mask combination. The address of 0.0.0.0 is actually just a placeholder; the mask, 255.255.255.255, actually does all the work. By placing a 1 in all 32-bit positions, this mask will match anything. The alternative statement shown uses the keyword any, which has the same meaning as the first statement.

PREV NEXT

# Extended IP Access Lists

Extended IP access lists provide far more flexibility in the specification of what is to be filtered. The basic syntax of the extended IP access list line follows:

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
    {deny | permit} protocol source source-wildcard
    destination destination-wildcard [precedence precedence] [tos tos]
    [log | log-input] [time-range time-range-name] [fragments]
```

You can configure extended access-lists using the global access-list configuration mode in the same way it is done with standard lists.

Sequence numbers can also be used with extended access lists. They are entered the same way as standard lists. Reflexive access lists can be configured only using the global access list configuration mode and can be configured only with named IP access lists. Reflexive access lists are discussed in a later section of this appendix.

Some of the features here are familiar, and some are new:

- *access-list-number*, for extended IP access lists, is between 100 and 199, or 2000 and 2699.
- dynamic identifies this list as a dynamic access list. Dynamic access lists are used by the "Lock-and-Key" security feature. A user uses Telnets to access the router, gets authenticated by an authentication server such as TACACS+ or RADIUS, and then is permitted or denied access based on the source and destination information in the dynamic entry.
- timeout defines the maximum amount of time, in minutes, a temporary entry can remain in a dynamic list. The default is not to time out the entry at all. It remains forever.
- *protocol* is a new variable that looks for a match in the protocol field of the IP packet header. The keyword choices are eigrp, gre, icmp, igmp, igrp, ip, ipinip, nos, ospf, tcp, or udp. An integer in the range 0 to 255 representing an IP protocol number can also be used. ip is a generic keyword, which matches any and all IP protocols, in the same way inverse mask 255.255.255.255 matches all addresses.
- Notice that both the *source* and *destination* packet addresses are examined for matches; each has its own inverse mask.
- precedence and tos are optional variables that look for a match in the Precedence and Type of Service fields of the IP packet header. Precedence can be an integer from 0 to 7, and TOS can be an integer from 0 to 15, or either field can be described by one of several keywords. Refer to the Cisco documentation for a list of available keywords.
- log is an optional specification that turns on informational logging. The router attempts to include the list number or name that logged the match, source and destination address, upper layer port number, and number of packets logged.
- log-input adds the input interface and source MAC address or virtual circuit number to the log output.
- time-range creates temporary access lists. Time-range defines the time interval that the access-list entry is valid. The time-range parameter in the extended access list references

a global time-range command. The global time-range defines the actual time parameters.

- fragments keyword defines how fragmented packets are handled by the access-list entry. Fragments are handled in different ways depending upon if Layer 3 or Layer 3 and Layer 4 information is specified in the access-list entry, and depending upon if the entry is to permit or deny the packet. The default behavior (no fragments keyword specified) for entries that contain Layer 3 (IP addresses, IP port numbers) information is to apply the entry to all nonfragmented packets, initial fragments, and noninitial fragments of packets. For entries that contain Layer 3 and 4 (TCP or UDP port numbers in addition to IP addresses) information, the entry is applied to nonfragments and initial fragments. The entry is also applied to noninitial fragments in the following way: If the noninitial fragment's Layer 3 information matches the Layer 3 information of the entry (the IP addresses, IP port number), and it is a permit statement, the fragment is permitted. If the entry is a deny statement, the next access-list entry is processed. If fragments is specified, the entry is applied only to noninitial fragments. The fragments keyword cannot be configured for entries that contain Layer 4 information, such as TCP or UDP port numbers.

A sample extended IP access list is displayed in Example B-12.

## Example B-12. An extended IP access list permits and denies packets in various ways.

```
access-list 101 permit ip 172.22.30.6 0.0.0.0 10.0.0.0 0.255.255.255 time-range
morning
access-list 101 permit ip 172.22.30.95 0.0.0.0 10.11.12.0 0.0.0.255
access-list 101 deny ip 172.22.30.0 0.0.0.255 192.168.18.27 0.0.0.0
access-list 101 permit ip 172.22.0.0 0.0.31.255 192.168.18.0 0.0.0.255
access-list 101 deny ip 172.22.0.0 0.0.255.255 192.168.18.64 0.0.0.63
access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
time-range morning
periodic weekdays 08:00 to 11:59
```

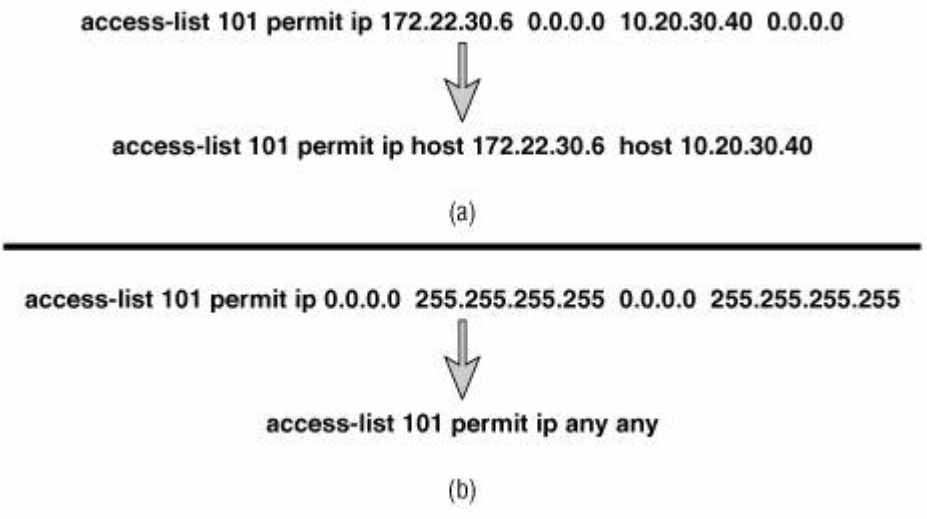
The following is an explanation of the lines in Example B-12:

- Line 1 IP packets with a source address of 172.22.30.6 and with a destination address that belongs to network 10.0.0.0 are permitted during a specific time range. The time range is defined in lines 7 and 8. During other times, the access-list entry is inactive. An inactive entry means that the entry is ignored as if it weren't in the list at all.
- Line 2 IP packets with a source address of 172.22.30.95 and with a destination address that belongs to subnet 10.11.12.0/24 are permitted.
- Line 3 IP packets with a source address that belongs to subnet 172.22.30.0/24 and with a destination address of 192.168.18.27 are dropped.
- Line 4 IP packets with source addresses between 172.22.0.0 and 172.22.31.255 and with a destination address that belongs to network 192.168.18.0 are permitted.
- Line 5 IP packets with a source address that belongs to network 172.22.0.0 and with a destination address whose first 26 bits are 192.168.18.64 are dropped.
- Line 6 IP packets from any source to any destination are permitted.
- Lines 7 and 8 The time range called "morning," which is referenced in line 1, is defined to be weekday mornings, from 08:00 to 11:59.

Figure B-7 shows two shortcuts that can be used when writing extended IP access lists. Recall that standard IP access lists have a default mask of 0.0.0.0. This default does not apply to extended access lists; there would be no way for the router to interpret it correctly. An

alternative exists for extended lists, however. In Figure B-7(a), packets are permitted if their source is host 172.22.30.6 and their destination is host 10.20.30.40. Any time the mask in an extended IP access list is 0.0.0.0, it can be replaced by adding the keyword host before the address.

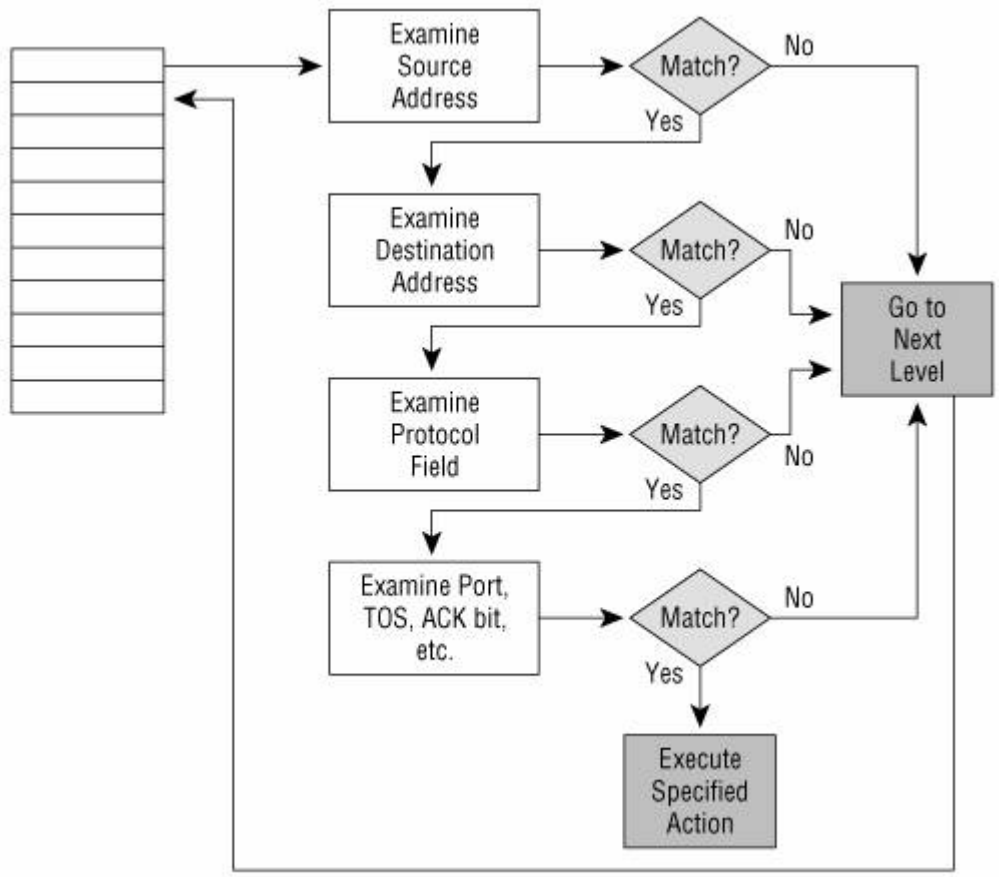
Figure B-7. Two shortcuts can be used when writing extended IP access lists.




The example in Figure B-7(b) permits any IP packets from any source to any destination. Just as with standard access lists, the any keyword can be used in place of the 0.0.0.0 255.255.255.255 address/inverse mask combination for the source, the destination, or both.

Extended access lists can be more powerful than standard access lists because the former examine more than the packet's source address, but everything has a price. The price you pay with extended lists is increased processing (Figure B-8). Because each line of the access list is examining multiple fields within the packet, multiple CPU interrupts can occur. If the access list is large or the router is busy, this requirement can affect performance adversely.

Figure B-8. The decision flow of an extended IP access list.




Keeping access lists as small as possible reduces the processing burden on the router. Also notice that when a match occurs, the specified action is invoked and processing stops. Therefore, if you can write your lists so that most matches occur in the first few lines, performance will be improved. This approach isn't always feasible, but it is something to keep in mind when designing access lists.

Some router platforms support a function called "Turbo ACLs," which are compiled access lists. The configured access lists are compiled into a lookup table. The sequence of the entries is maintained, but the lookup time and CPU required for lookup is greatly reduced. Certain entries, such as time ranges, cannot be included in a compiled list. To configure the router to use turbo access lists, enter the command access-list compiled.

As an exercise, try making the access list given in Example B-12 more elegant. That is, rewrite the list with as few lines as possible without losing any of its functionality. (Hint: A list with the same functionality can be written with only four lines, not including the two time commands at the end.) An answer is given in the next paragraph. Try to rewrite the list before reading further.

Line 1 can be removed. Line 1 permits host 172.22.30.6 access to the 10.0.0.0/8 address, during the weekday mornings. Without this line, access from this host to the 10.0.0.0 address is still permitted by line 6, which permits anything that is not previously denied.

Line 2 can also be removed. The host 172.22.30.95 is also permitted access to 10.11.12.0/24 in line 6.

You might be tempted to think that line 4 can also be removed, but notice that line 5 denies a larger range of addresses that includes those permitted in line 4. Therefore, line 4 is necessary to permit a small subset of addresses before the rest of the addresses specified in line 5 are dropped.

## TCP Access Lists

The syntax for an extended access list line that examines a TCP segment follows:

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
    {deny | permit} tcp source source-wildcard [operator [port]]
    destination destination-wildcard[operator [port]] [established]
    [precedence precedence] [tos tos] [log | log-input]
    [time-range time-range-name] [fragments]
```

Notice that the *protocol* variable is tcp. Probably the most significant feature here is that the access list can examine the source and destination port numbers in the TCP segment header. As a result, you have the option of filtering packets not only to and from a particular address, but also to and from a particular socket (an IP address/application port combination).

The features of the TCP access list that have not yet been explained are *operator* and *port*:

- *operator* specifies a logical operand. The options are eq (equal to), neq (not equal to), gt (greater than), lt (less than), and range for specifying an inclusive range of ports. If the range operator is used, two port numbers are specified.
- *port* specifies the application layer port to be matched. A few common port numbers are for Telnet (23), FTP (20 and 21), SMTP (25), and SNMP (169). A complete listing of TCP port numbers can be found in RFC 1700.

What happens if you implemented an access list to prevent TCP sessions from being established into your network, but you want to ensure that the access list passes the responses if your network establishes a TCP session? The established keyword allows this event by checking the ACK and RST flags in the TCP segment header. If one of these flags is set, a match occurs. If neither bit is set, the source is trying to establish a TCP connection to the destination, and a match will not occur. The packet will be denied on a subsequent line of the access list.

A sample TCP access list is displayed in Example B-13.

### Example B-13. This TCP access list permits established sessions and permits certain addresses access for SMTP and Telnet.

```
access-list 110 permit tcp any 172.22.0.0 0.0.255.255 established
access-list 110 permit tcp any host 172.22.15.83 eq 25
access-list 110 permit tcp 10.0.0.0 0.255.255.255 172.22.114.0 0.0.0.255 eq 23
```

The following is an explanation of the lines in Example B-13:

- Line 1 Permit TCP packets from any source to network 172.22.0.0 if the connection was established from that network.
- Line 2 Permit TCP packets from any source if the destination is port 25 (SMTP) of host 172.22.15.83.
- Line 3 Allow any TCP packet with a source address from network 10.0.0.0 to telnet (port 23) to any address on subnet 172.22.114.0/24.

All other packets will be dropped by the implicit deny any.

## UDP Access Lists

The syntax for an extended access list line that examines a UDP segment follows:

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
   {deny | permit} udp source source-wildcard [operator [port]]
   destination destination-wildcard [operator [port]]
   [precedence precedence] [tos tos] [log | log-input]
   [time-range time-range-name] [fragments]
```

This format is similar to the TCP format, except that the *protocol* variable now is udp. The other difference is that there is no established keyword. The reason is that UDP is a connectionless transport service, and no connections are established between hosts.

In Example B-14, three lines have been added to the previous TCP example.

### Example B-14. This access list permits TCP and UDP packets.

```
access-list 110 permit tcp any 172.22.0.0 0.0.255.255 established
access-list 110 permit tcp any host 172.22.15.83 eq 25
access-list 110 permit tcp 10.0.0.0 0.255.255.255 172.22.114.0 0.0.0.255 eq 23
access-list 110 permit udp 10.64.32.0 0.0.0.255 host 172.22.15.87 eq 69
access-list 110 permit udp any host 172.22.15.85 eq 53
access-list 110 permit udp any any eq 161
```

The following is an explanation of the lines in Example B-14:

- Line 4 Permit UDP packets from subnet 10.64.32.0/24 to the TFTP port (69) on host 172.22.15.87.
- Line 5 Permit UDP packets from any source to the Domain Name Server (port 53) on host 172.22.15.85.
- Line 6 Permit all SNMP packets (port 161) from any source to any destination.

The implicit deny any still drops all packets not finding a match in the list.

## ICMP Access Lists

The syntax for an extended access list line that examines an ICMP packet follows:

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
    {deny | permit} icmp source source-wildcard destination destination-wildcard
    [icmp-type[icmp-code] | icmp-message] [precedence precedence] [tos tos]
    [log | log-input] [time-range time-range-name] [fragments]
```

icmp is now in the *protocol* field. Notice that there are no source or destination ports here; ICMP is a network layer protocol. This line can be used to filter all ICMP messages, or you can use the following options to filter specific ICMP messages:

- *icmp-type* is a number between 0 and 255. All ICMP type numbers can be found in RFC 1700.

- The granularity of filtering can be increased by specifying *icmp-code*. An ICMP code specifies a subset of ICMP packet types; the codes are a number between 0 and 255 and are also found in RFC 1700.
- Instead of an ICMP type and ICMP code, an ICMP message name can be entered.

A sample of an ICMP access list is displayed in Example B-15.

### Example B-15. This ICMP access list denies specific packets and permits all others.

```
access-list 111 deny icmp 172.22.0.0 0.0.255.255 any 0
access-list 111 deny icmp 172.22.0.0 0.0.255.255 any 3 9
access-list 111 deny icmp 172.22.0.0 0.0.255.255 any 3 10
access-list 111 deny icmp 172.22.0.0 0.0.255.255 any traceroute
access-list 111 permit ip any any
```

The following is an explanation of the lines in Example B-15:

- Line 1 Deny ICMP ping responses (Echo Reply, ICMP type 0) from network 172.22.0.0 to any destination.
- Line 2 Deny ICMP destination unreachable packets (type 3) with a code number of 9 (Network Administratively Prohibited) from network 172.22.0.0 to any destination.
- Line 3 Deny ICMP destination unreachable packets (type 3) with a code number of 10 (Host Administratively Prohibited) from network 172.22.0.0 to any destination.
- Line 4 Deny ICMP traceroute from network 172.22.0.0 to any destination.
- Line 5 Permit all other IP packets.

PREV NEXT

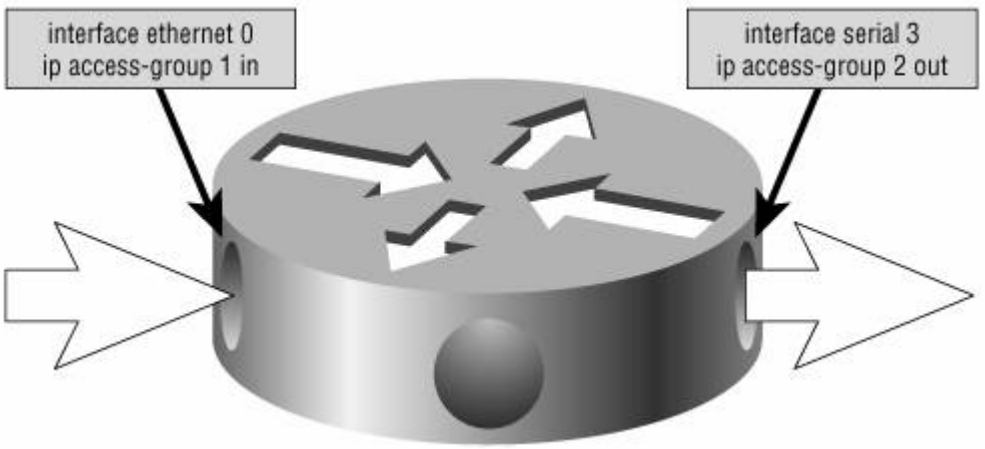
# Calling the Access List

An access list does nothing unless packets are sent to it by a calling command, which defines how the access list is to be used. One such command is

```
ip access-group access-list-number {in | out}
```

This command is configured on an interface to create security or traffic filters and can be applied to incoming or outgoing traffic. If neither the in nor the out keyword is specified, the filter defaults to outgoing. The access list number, of course, is the access list to which this command will send packets. Figure B-9 shows two configurations of this command.

Figure B-9. The *ip access-group* command uses the specified access list to create a filter on an interface for either incoming or outgoing packets.




Access list 1 in Figure B-9 filters incoming IP packets on interface E0. It has no effect on outgoing IP traffic and no effect on packets originated by other protocols, such as IPX. Access list 2 filters IP packets going out interface S3. It has no effect on incoming IP packets and no effect on packets originated by other protocols. Denying as much as possible on the inbound access list instead of the outbound list alleviates the need for the router to perform work on traffic that will be dropped.

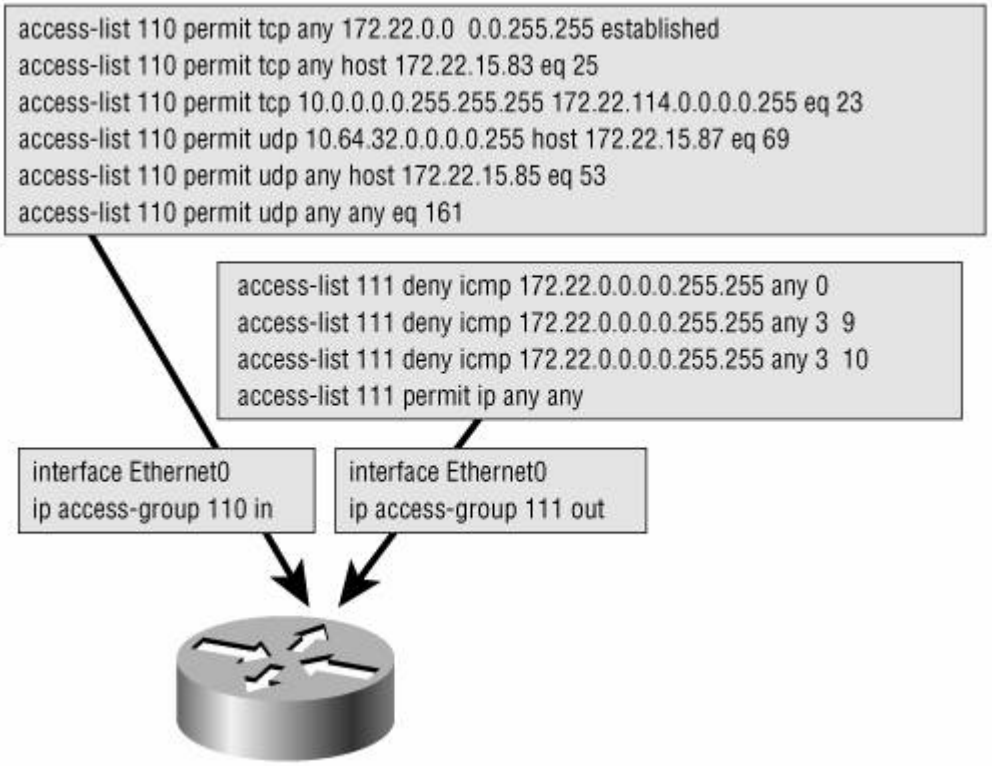
Multiple interfaces can make calls to the same access list, but any one interface can have only one incoming and one outgoing access list for each protocol.

In Figure B-10, the TCP, UDP, and ICMP access lists given earlier as examples are used as filters. Access list 110, from the previous two examples, has been applied to the Ethernet 0 interface to check incoming traffic. Access list 111 is applied to the same interface to check outgoing traffic. Analyze the two access lists carefully, including their interrelationship, and consider the following:

- A ping response from 172.23.12.5 to 10.64.32.7 wants to exit interface Ethernet 0. Will it be allowed to pass?

- Someone on 172.22.67.4 wants to ping a device at 10.64.32.20, exiting Ethernet 0. Will the ping be successful?

Figure B-10. Access list 110 is used here to filter incoming packets on the Ethernet interface. Access list 111 is used here to filter outgoing packets on the same interface.



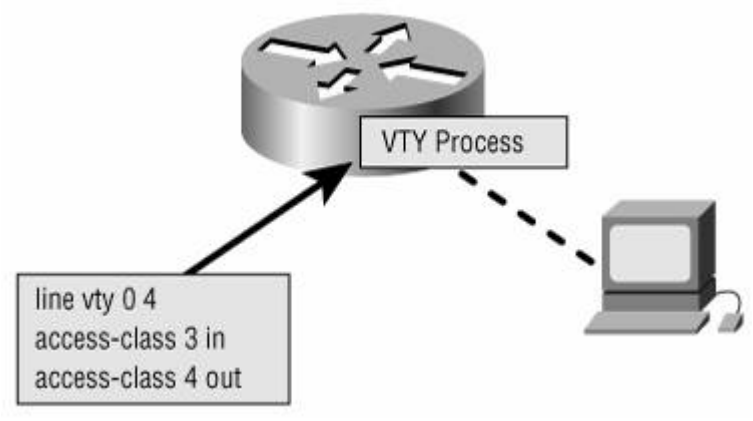

The ping response from 172.23.12.5 will be allowed to exit interface Ethernet 0. Ping responses are denied for packets from 172.22.0.0/16, not 172.23.0.0/16. The ping from 172.22.67.4 to 10.64.32.20, exiting Ethernet 0, will not be successful. The ping request will successfully exit the interface, but the response will be denied by the inbound access list.

Another command that makes calls to an access list is the access-class command. This command is used to regulate Telnet sessions to and from the router's virtual terminal lines, not for packet filtering. The format of the command is

**access-class** *access-list-number* {**in** | **out**}

Figure B-11 shows an example of the access-class command. Access list 3 regulates the addresses from which the router's VTY lines will accept Telnet sessions. Access list 4 regulates the addresses to which the router's virtual terminal lines may connect.

Figure B-11. The *access-class* command uses an access list to regulate Telnet traffic to and from the router's virtual terminal lines.




The access-class command has no effect on Telnet traffic transiting the router. It influences only Telnet sessions to and from the router itself.

PREV NEXT

# Reflexive Access Lists

Reflexive access lists are automatically populated, temporary, session-based filters. If a router per to be initiated from within a network to an external host, a reflexive list permits return session tra lists are used with extended named IPv4 access lists. Session filters using reflexive lists can be co established keyword used with TCP filters. Using the established keyword, a TCP session is initi within a network. If the return traffic has the ACK or RST flag set, the packet is part of a previousl session, and the packet is permitted. This entry with the established keyword is a permanent ent access list.

Reflexive access lists use different parameters to determine if the packet is part of a previously es session. For TCP and UDP packets, reflexive access lists use source and destination IP addresses a destination TCP or UDP port numbers.

When a session is initiated from within a network, a reflexive access list is populated with the sess information gleaned from the initial packet. The source and destination IP addresses and the sourc destination port numbers are swapped and added, along with the upper layer protocol type (such UDP) as a permit statement to the temporary reflexive list. This entry remains active until there is traffic for the session and the timeout value expires, until two FIN-flagged packets are received, o flag is set on a TCP packet.

Example B-16 shows an example of a reflexive access list configuration.

### Example B-16. This reflexive access list is named infilter.

```
interface Serial0/0.1 point-to-point
 ip address 172.25.150.65 255.255.255.192
 ip access-group infilter in
 ip access-group outfilter out
!
ip access-list extended infilter
 permit eigrp any any
 permit udp any any eq rip
 evaluate sessiontraffic
ip access-list extended outfilter
 permit tcp any any reflect sessiontraffic
 permit icmp any any echo time-range morning reflect sessiontraffic
!
time-range morning
 periodic weekdays 9:00 to 12:30
```

In this example, the filters are applied to an interface that connects to an external network. The o permits all TCP packets and ICMP echo requests on weekday mornings between 9:00 and 12:30 o from the internal network. The outfilter list is applied outbound on serial0/0.1. The reflect keywor the permit statements. This creates the reflexive access list called sessiontraffic. The reflexive acc populated when packets match the permit entries that use the reflect keyword.

Packets coming inbound to interface serial0/0.1 are filtered by the infilter access list. These would packets sourced from an external network. In this case, infilter permits EIGRP and RIP packets. Af incoming packet is matched against the EIGRP and RIP entries, the reflexive access list sessiontra evaluated sequentially. The reflexive access list does not have an implicit deny-all at the end, but access list in which the reflexive list is nested does.

Example B-17 shows the access lists before TCP and ICMP traffic has exited serial0/0.1. Example E

access lists after TCP and ICMP traffic has exited serial0/0.1.

### Example B-17. *show ip access-list* displays all the configured permanen temporary IP access lists configured on a router.

```
Router#show access-lists
Extended IP access list infilters
  10 permit eigrp any any
  20 permit udp any any eq rip (1074 matches)
  30 evaluate sessiontraffic
Extended IP access list outfilter
  10 permit tcp any any reflect sessiontraffic (45 matches)
  20 permit icmp any any echo time-range morning (active) reflect sessiontraffic
Reflexive IP access list sessiontraffic
```

Ping and Telnet have been initiated from the internal network to the external network. Example B- access lists after this traffic has been initiated.

### Example B-18. The *show ip access-list* displays dynamically created ent reflexive access list.

```
Router#show ip access-list
Extended IP access list infilters
    10 permit eigrp any any
    20 permit udp any any eq rip (1101 matches)
    30 permit udp any any eq 521
    40 evaluate sessiontraffic
Extended IP access list outfilter
    10 permit tcp any any reflect sessiontraffic (188 matches)
    20 permit icmp any any echo time-range morning (active) reflect sessiontraffic
Reflexive IP access list sessiontraffic
     permit tcp host 192.168.16.225 eq telnet host 192.168.50.130
     eq 11002 (55 matches) (time left 293)
    permit icmp host 192.168.16.225 host 192.168.50.130 (19 matches) (time left 27(
```

The output in Example B-17 displays the access list's infilter and outfilter and their configured para Notice that the ICMP entry in the outfilter says it is active. This means the time and day of the wee router falls within the configured time range. The output also displays the nonpopulated reflexive a sessiontraffic.

After ICMP pings and a Telnet session have been initiated and packets have exited serial0/0.1, the are displayed again in Example B-18. This time there are entries in the reflexive access list. These matched against all packets arriving into serial0/0.1 from the external network, until the timer exp session has been closed. Telnet session and ICMP echos are not successful when initiated from the network.

Reflexive access lists do not work for protocols that change port numbers during a session, such a

# Keyword Alternatives

Most networking professionals know some of the more commonly used TCP port numbers, and maybe a few UDP port numbers. Fewer can say what the ICMP type is for a ping or a destination unreachable, much less what the ICMP codes are for destination unreachable types. Beginning with IOS 10.3, access lists can be configured with keywords in place of many port, type, or code numbers. Using keywords, the access lists 110 and 111 from Figure B-10 are displayed in Example B-19.

### Example B-19. Keywords can replace port numbers in access lists.

```
access-list 110 permit tcp any 172.22.0.0 0.0.255.255 established
access-list 110 permit tcp any host 172.22.15.83 eq smtp
access-list 110 permit tcp 10.0.0.0 0.255.255.255 172.22.114.0 0.0.0.255 eq telnet
access-list 110 permit udp 10.64.32.0 0.0.0.255 host 172.22.15.87 eq tftp
access-list 110 permit udp any host 172.22.15.85 eq domain
access-list 110 permit udp any any eq snmp
!
access-list 111 deny icmp 172.22.0.0 0.0.255.255 any echo-reply
access-list 111 deny icmp 172.22.0.0 0.0.255.255 any net-unreachable
administratively-prohibited
access-list 111 deny icmp 172.22.0.0 0.0.255.255 any host-unreachable
administratively-prohibited
access-list 111 permit ip any any
```

A word of caution: If you upgrade a router from a pre-10.3 image, the new IOS, upon bootup, will rewrite the access lists in the configuration file to the new syntax, including keywords. If you subsequently need to reload the original pre-10.3 image, the revised access lists will not be understood. Always upload a copy of the original configuration file to a TFTP server before upgrading.

# Named Access Lists

The limit of 798 standard access lists or 799 extended IP access lists per router would seem to be more than enough; however, there are cases, such as with dynamic access lists,[1] in which these maximums might not be sufficient. Named access lists, available beginning with IOS 11.2, extend these limits. The other advantage is that descriptive names can make large numbers of lists more manageable.

[1] Dynamic access lists are not covered in this tutorial. Refer to the Cisco IOS Security Configuration GuideConfiguring Lock-and-Key Security (Dynamic Access Lists) for more information.

To use names, use the following syntax in the first line of the access list:

```
ip access-list {standard | extended} name
```

Because there are no numbers to differentiate list types, this line specifies the list as IP and either standard or extended.

Below the beginning line, go the permit and deny statements. The syntax for the standard list is

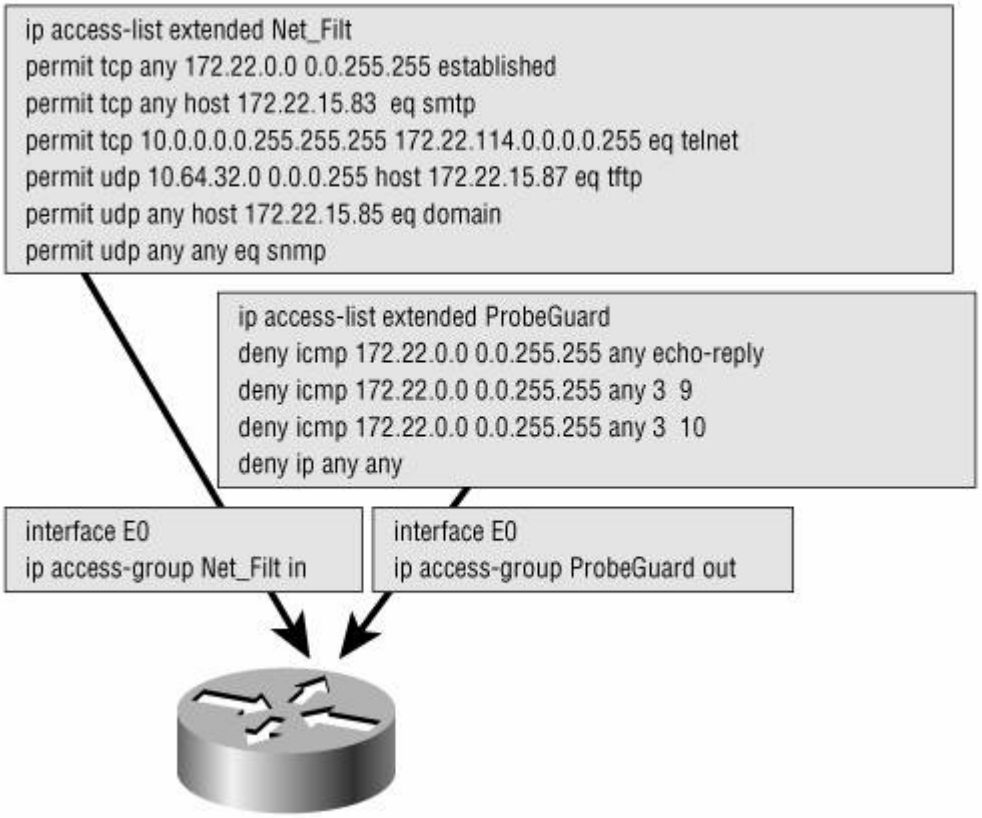
```
{deny | permit} source [source-wildcard]
```

The syntax for the basic extended list is

```
{deny | permit} protocol source source-wildcard destination destination-wildcard
    [precedence precedence][tos tos][log]
```

In both cases, the access-list *access-list-number* portion of the command has disappeared, but everything else remains the same. Standard and extended access lists on the same router cannot share the same name. The command for establishing a named access list on an interface refers to the name instead of a number but in all other ways remains the same. Figure B-12 shows the access lists of Figure B-10 converted to the named format.

Figure B-12. The access lists shown in Figure B-10 are now configured as named access lists.


ip access-list extended Net_Filt
permit tcp any 172.22.0.0 0.0.255.255 established
permit tcp any host 172.22.15.83 eq smtp
permit tcp 10.0.0.0.0.255.255.255 172.22.114.0.0.0.0.255 eq telnet
permit udp 10.64.32.0 0.0.0.255 host 172.22.15.87 eq tftp
permit udp any host 172.22.15.85 eq domain
permit udp any any eq snmp
ip access-list extended ProbeGuard
deny icmp 172.22.0.0 0.0.255.255 any echo-reply
deny icmp 172.22.0.0 0.0.255.255 any 3 9
deny icmp 172.22.0.0 0.0.255.255 any 3 10
deny ip any any
interface E0
ip access-group Net_Filt in
interface E0
ip access-group ProbeGuard out

PREV NEXT

# Prefix Lists

Prefix lists are used to specify an address or range of addresses to be permitted or denied in route updates. The BGP routing protocol uses prefix lists for IPv4. All IPv6 routing protocols can use prefix lists when exchanging IPv6 addresses between protocols or when filtering updates.

Prefix lists are named lists. An entry in the list permits or denies an address or range of addresses, as in Example B-20.

### Example B-20. Prefix lists permit and deny IPv6 addresses.

```
ipv6 prefix-list v6_addr_filt permit 2001:db8:0:1::/64
ipv6 prefix-list v6_addr_filt permit 2001:db8:0:10::/60 le 64
ipv6 prefix-list v6_addr_filt permit ::/0 ge 62 le 64
```

The first entry permits prefix 2001:db8:0:1:: with the prefix length exactly 64 bits. The second and third entries permit a range of addresses. The keyword le indicates that the range of prefix lengths to be matched is from the length specified, after the prefix to the length specified after the le keyword. The second entry in the prefix list v6_addr_filt permits prefixes which match 2001:db8:0:10:: and have a length in the range 60 to 64. The ge keyword specifies the minimum length of the prefix in a range of addresses. If it is used with no le keyword, it is assumed that the maximum length for the range of prefixes matched is 128 bits, the maximum number of bits in an IPv6 prefix. When used with the le keyword, the maximum matched length of the range is specified after le. The third entry in prefix list v6_addr_filt permits any prefix with a length between 62 and 64 bits.

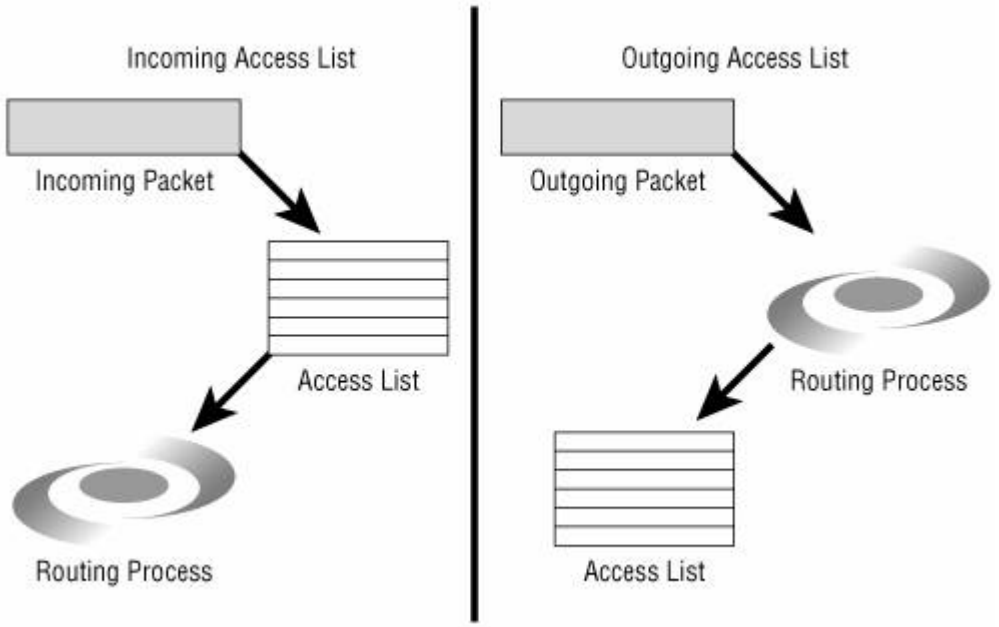
# Filter Placement Considerations

For the best performance, you must consider not only the efficient design of the access list itself, but also the placement of the filter on the router and in the network.

As a rule of thumb, security filters usually are incoming filters. Filtering unwanted or untrusted packets before they reach the routing process, prevents *spoofing attacks*wherein a packet fools the routing process into thinking it has come from somewhere it hasn't. Traffic filters, on the other hand, usually are outgoing filters. This approach makes sense when you consider that the point of a traffic filter is to prevent unnecessary packets from occupying a particular data link.

Aside from these two rules of thumb, another factor to consider is the number of CPU cycles the combined access list and routing processes will use. An incoming filter is invoked before the routing process, whereas an outgoing filter is invoked after the routing process (Figure B-13). If most packets passing through the routing process are to be denied by the access list, an incoming filter might save some processing cycles.

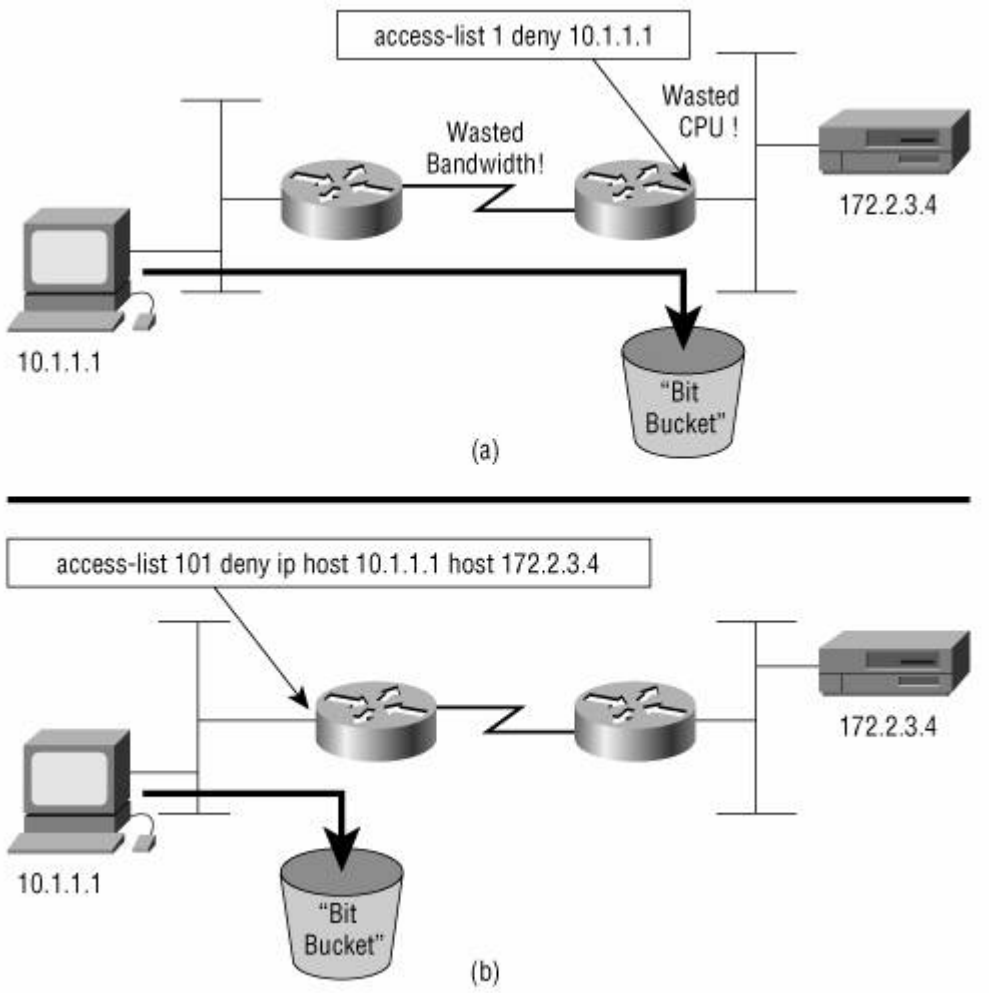
Figure B-13. Incoming packet filters are invoked before the routing process, whereas outgoing packet filters are invoked after the routing process.




Standard IP access lists can filter only on source addresses. Consequently, a filter using a standard list must necessarily be placed as close to the destination as possible so that the source still has access to other, nonfiltered destinations (Figure B-14(a)). As a result, bandwidth and CPU cycles might be wasted delivering packets that will ultimately be dropped.

Extended IP access lists, because of their capability to identify specific packet characteristics, should be placed as close to the source as possible to prevent wasting bandwidth and CPU transporting "doomed" packets (Figure B-14(b)). On the other hand, the complexity of extended lists means more of a processing burden. These tradeoffs must be considered when deciding where on the network to place a filter.

Figure B-14. Filters that use standard access lists generally must be placed close to the destination (a), whereas extended access lists can be placed close to the source (b).




You must also understand how your access list will affect switching on the router. For instance, an interface using an extended IP access list cannot be autonomously switched; dynamic access lists cannot be silicon-switched and might affect silicon-switching performance. Named access lists are not supported at all before IOS 11.2.

The effect of an access list on switching might be critical on backbone or core routers. Be sure to fully research and understand the effects an access list might have by reading the Cisco Configuration Guide for the IOS being used on your router. In some cases, a *packet filtering router*a smaller router dedicated to nothing but packet filteringcan be used to offload the filtering burden from a mission-critical router.

PREV NEXT

# Access List Monitoring and Accounting

It is useful to examine an access list, or even all access lists, without having to display the entire router configuration file. The command show ip access-list displays an abbreviated syntax of all IP access lists on the router. If a specific access list is to be observed, the list can be specified by name or number (Example B-21). If you leave off the ip keyword (show access-list), all access lists will display.

### Example B-21. The *show ip access-list* command displays an abbreviated syntax of the access lists.

```
Woody#show ip access-list 110
Extended IP access list 110
    10 permit tcp any 172.22.0.0 0.0.255.255 established
    20 permit tcp any host 172.22.15.83 eq smtp
    30 permit tcp 10.0.0.0 0.255.255.255 172.22.114.0 0.0.0.255 eq telnet
    40 permit udp 10.64.32.0 0.0.0.255 host 172.22.15.87 eq tftp
    50 permit udp any host 172.22.15.85 eq domain
    60 permit udp any any eq snmp
Woody#
```

It is also useful, as part of a security plan or a capacity planning strategy, to track packets that have been denied by an access list.

The command ip accounting access-violations might be configured on individual interfaces to create a database of all packets that have been denied by any access lists on that interface. To examine the database, use the command show ip accounting access-violations. The source and destination addresses, the number of packets and number of bytes matching these addresses and the access list number that denied the packet will be shown (Example B-22). The command clear ip accounting clears the accounting database.

### Example B-22. The access list accounting database can be observed with the command *show ip accounting access-violations*.

```
Woody#show ip accounting access-violations
   Source          Destination             Packets          Bytes     ACL
  10.1.4.1       255.255.255.255                13            936     110
  10.1.4.1       172.22.1.1                     12           1088     110

Accounting data age is 10
Woody#
```

Accounting disables autonomous and silicon switching on an interface. Do not use accounting on an interface where these switching modes are required.

As a final "trick," be aware that its accounting does not track packets discarded by the implicit deny any at the end of the list. To track these packets, simply configure a deny any at the end of the list as in Example B-23.

Example B-23. A *deny any* entry is added to the end of the access list to track packets discarded because they did not match any other entry in the list.

```
access-list 110 permit tcp any 172.22.0.0 0.0.255.255 established
access-list 110 permit tcp any host 172.22.15.83 eq smtp
access-list 110 permit tcp 10.0.0.0 0.255.255.255 172.22.114.0 0.0.0.255 eq telnet
access-list 110 permit udp 10.64.32.0 0.0.0.255 host 172.22.15.87 eq tftp
access-list 110 permit udp any host 172.22.15.85 eq domain
access-list 110 permit udp any any eq snmp
access-list 110 deny ip any any 1
```

PREV NEXT

# Appendix C. CCIE Preparation Tips

Becoming a Cisco Certified Internetworking Expert (CCIE) is a far cry from the "read a book, take an exam" process of some other industry certifications. After a relatively easy written exam, you will be required to prove your expertise in a hands-on, notoriously difficult lab exam. Even though you must be intimately familiar with the Cisco configuration commands, the most difficult challenges of the lab are not Cisco IOS Software-specific; instead, they test the depths of your understanding of switches, routers, and routing protocols. It is for this knowledge that CCIEs are recognized and sought out as proven internetworking experts.

The structured creation of a network involves four phases, and those same four phases are also useful for creating a structured preparation program for the CCIE lab:

- Plan Take a cold, hard look at your present experience level and your shortcomings. Evaluate the daily time you have available for study. Evaluate the resources at your disposal, including lab equipment; funds and time available for training; books; and acquaintances that can serve as coaches, tutors, and subject matter experts. Evaluate your personal strengths and weaknesses: Are you a good test taker? Do you work well under pressure? How do you react to setbacks and disappointments? Do you have good study habits? Do you learn best from reading or from verbal instruction? Using the raw data from your evaluations, write a list of your assets and liabilities. Develop a plan to fully capitalize on your assets and eliminate as many liabilities as possible.

- Design Design a personalized preparation program that meets your needs while fitting your schedule and resources. Talk to as many CCIEs as possible; ask them about their own preparation programs. Find out what worked and what didn't work for them. Your program should take you from your present experience level right up to the CCIE lab, with definite deadlines and milestones. Build the project from a series of miniprojects, each with a well-defined goal. Be realistic when you design your schedule, taking into consideration the predictability (or lack of) of both your job and your personal life. The level of support you can expect from your employer and your family is an important factor in deciding whether your preparation schedule should be intensive or more relaxed. Exceeding the tolerance of those close to you will hurt your schedule far more than help it.

- Implement Many projects fail because the implementation begins before the design is complete. Your preparation program should be a written document clearly defining all steps of the project from kickoff to completion. After you begin your preparation program, stick to it. Don't give up, don't be discouraged, and don't be lazy. Check off your goals and milestones as you meet them.

- Optimize Your preparation program should be a living document. As you progress, some subjects will be more difficult than expected, and some subjects will be easier than expected. Always move forward but be flexible enough to add any extra tasks necessary for you to master each topic.

Only you can design a preparation program that best suits you. The advice in the following sections is not meant to be followed unswervingly, but is meant to give you some ideas for creating your study program. These tips come from my personal experience as both a CCIE and as a Cisco Systems Instructor, and from the experiences of associates who have successfully passed the CCIE lab.

# Laying the Foundations

If you are a beginner, or your networking experience is limited, your first step is to get a solid grip on the basics of both networking and Cisco routers. This effort will involve both classroom training and self-study. I recommend Cisco.com; look at the wealth of information about career certifications in the "Learning and Events" section.

Through its Authorized Learning Partners, Cisco Systems offers many hands-on training classes. You should attend as many of these classes as your time and resources allow, but of particular importance are the following:

- Interconnecting Cisco Network Devices (ICND)
- Building Scalable Cisco Internetworks (BSCI)
- Building Cisco Remote Access Networks (BCRAN)
- Building Cisco Multilayer Switched Networks (BCMSN)
- Cisco Internetwork Troubleshooting (CIT)

Take full advantage of every class you attend. Ask questions of the instructor and discuss the class topics with your fellow students. Most importantly, take advantage of the access you have to the lab equipment. Don't just work the labs; be sure you fully understand the whys and hows of the lab exercises. When you finish a lab, don't just stop. Play with the equipment. See what configuration and troubleshooting options are available and try them. If you have time, try building the lab configuration several times to gain proficiency.

The classroom work will help you identify gaps in your networking knowledge. Read as much as you can to fill in the gaps in your knowledge of basic networking protocols and technology. Many good tutorials are available on the Internet from both commercial vendors and private individuals. Whenever you begin studying a particular subject, be sure to perform a Web search for the topic.

# Following the Certification Path

When the first edition of this book was written, the CCIE was the only certification Cisco Systems offered. Now, there are a number of levels of certification and even a number of specialties within the CCIE program. Taking advantage of these levels is encouraged (although none of them are required prerequisites), as it gives you milestones along your road to the CCIE and also awards you with meaningful certifications along the way.

The first goal to work toward is the Cisco Certified Networking Associate (CCNA). This certification proves that you have mastered the basics of networking and network protocols, can install and operate small LAN, WAN, and access networks, and understand essential Cisco hardware and IOS Software.

Your next goal is the Cisco Certified Network Professional (CCNP). This advanced certification indicates a mastery of the technologies necessary for running and troubleshooting larger enterprise networks. With the CCNP in hand, you are ready to move to the expert level and prove it with your CCIE.

# Hands-On Experience

Almost all CCIEs will tell you that hands-on experience is an invaluable part of preparing for the lab exam. Never pass up an opportunity to configure or troubleshoot a router. If you do not work with routers and switches on your present job, get friendly with the network engineers and technicians in your organization. Explain your goals to them and offer to assist them whenever possible.

If you have access to lab facilities, take full advantage of them. There is no replacement for the experience you can gain from working in a lab, where you can configure whatever you want to configure and introduce whatever problems you want to introduce, without risk of disrupting a production network.

Several online companies provide remote access to practice labs for a fee. The usage fees can vary widely, usually dependent on the rack of equipment available to you.

Another option is to build your own lab. Although this option is expensive, the salary you can command as a CCIE might make the investment worthwhile. Many sources sell used Cisco equipment at fairly reasonable prices. Subscribe to the Cisco newsgroup on the Internet, at http://comp.dcom.sys.cisco, or a study group such as groupstudy.com; people frequently post used routers for sale, and you can find some good deals. Most of my own lab routers, used for developing the examples in this book, were purchased on eBay. Although even two routers are useful, you should try to obtain at least four, one of which should have four or more serial interfaces so that you can configure it as a Frame Relay switch. Remember that you don't need top-of-the-line equipment; obsolete routers are especially good buys because no one wants them in a production network.

# Intensifying the Study

Beyond the basics, and in parallel with your acquisition of practical hands-on experience, you must begin building a deep understanding of the networking protocols. At a minimum, you should read the RFCs recommended in this book. Ideally, read as many relevant RFCs as you can. The best site is http://www.ietf.org.

Of course, not all networking protocols are described in RFCs. Look for advanced books, white papers, and tutorials on the non-IP protocols, that might be included on the exam. You should also study Ethernet and WAN protocols, such as T-1, ISDN, Frame Relay, and ATM. You can find a wealth of publicly available information at Cisco.com. And of course, Cisco Press offers a number of books besides this one and its companion volume that are specifically focused on CCNA, CCNP, and CCIE. Go to Ciscopress.com to find the books appropriate for your needs.

A useful study tool, incorporating both theory and practical knowledge, is the Cisco newsgroup at http://comp.dcom.sys.cisco. Copy particularly challenging questions and problems posted to the newsgroup and find your own answer. Then, watch for the answers posted from CCIEs and Cisco engineers and see whether you are right. If not, determine why. Post questions to the Cisco newsgroup, too. Most regular participants are friendly and willing to share their expertise.

Several online study groups focus specifically on preparing for the CCNP and CCIE exams. One of the oldest and most well-known of these is groupstudy.com. Joining such a group is an excellent way of locating resources, sharing experiences, joining discussions, and getting answers to thorny questions. I highly recommend joining an online study group.

Finally, if you have associates with the same goals, form your own study group. I know of CCIE study groups within several companies that have been very effective and have produced many CCIEs.

# The Final Six Months

With a now-solid background combining practical and theoretical knowledge, your final six months of preparation should involve reading the Cisco IOS Configuration Guides from cover to cover. As you read each chapter of this book, review the associated chapter in the Cisco IOS Command Reference to ensure that you are familiar with the full configuration capabilities of the IOS for that protocol. Then, use your lab to configure the protocol covered in the chapter in as many ways as you can. Play "what if" games, trying to make the protocol work in unusual situations. You will find that the best learning experiences result not when a configuration works as expected, but when it doesn't.

Keep a notebook of your configurations and your thoughts on how they worked or didn't work. Be sure to explore and record all troubleshooting tools, such as debug and show commands, relevant to the protocol.

Your goal at the end of each chapter of the Configuration Guide is to be able to configure at least the essential elements of the protocol from memory. When you are taking the CCIE lab, it is important to be able to configure the "easy stuff" with little thought so that you can be thinking ahead to the difficult configuration problems. You should also be familiar with how the protocol behaves, how the protocol interacts with other protocols, what configuration options exist, and how to troubleshoot the protocol. When you meet these goals for one chapter, move to the next.

Early in this final six months, you should take the written portion of the CCIE exam. Don't be fooled by this examit is intended primarily to weed out people who are completely unprepared so that they are only out the price of the written test instead of the much higher price of the lab exam. If you followed a good study program, you will not find the written exam to be particularly difficult.

Several companies offer CCIE preparation courses and "CCIE boot camps," which give you experience doing lab work under conditions similar to the CCIE lab exam. Do not mistake these classes as a substitute for diligent study and practice. If you choose to attend one of these classes, you should first be fully prepared to take the CCIE exam. The greatest benefit you will gain from the commercial preparation labs is a feel for being "under the gun," working difficult problems within tight time constraints.

# Exam Day

The CCIE lab exam tests not only your practical and theoretical knowledge, but also your ability to use that knowledge under pressure. You are in for an intense day, so do not add unnecessarily to that pressure:

- Be aware that most people fail the CCIE exam on their first try. Have a contingency plan for taking the exam a second time. This strategy can help you stay calm during your first try and might make your contingency plan unnecessary.
- Arrange your travel plans so that you arrive in plenty of time the day before the exam. Arrange to leave the day after the exam, not the same evening. You don't want to think about your travel schedule during the exam.
- Locate the test facilities the evening before the exam. You don't want to show up for the exam flustered because you got lost.
- Do no more than a light review on the evening before the exam. If you try to "cram," you will make yourself nervous and sleepless.
- Eat a good dinner, with no alcohol, and get a good night's sleep.
- On the day of the exam, eat a good breakfast. It is an established fact that eating correctly will help you perform better.
- Dress comfortably. There are no points for appearance.

Before the exam, you will be required to submit an online nondisclosure form stating that you will not divulge the details of the lab. The exam is eight hours long with a break for lunch. Although other CCIE candidates will be working on their own labs in the room, you will be working alone. In all likelihood, the other candidates' assignments will be different from yours. You will be assigned a network topology and a set of questions and configuration requirements that must be fulfilled within the eight-hour lab.

If you do not understand a particular requirement during any part of the exam, do not hesitate to ask your lab proctor. The proctor is there not only to test you but also to help you. Most important, relax and stay focused.

The lab is graded after hours, and the results are usually sent the next business day after the exam. You will receive an e-mail telling you that your results are available, and you can access your online candidate profile to see the result. If you have passed, you will see your CCIE number; if not, you will see a score report.

When you have won your CCIE, you will have accomplished something to be proud of. If this book or the advice in this appendix has helped you reach your goal, please e-mail me so that I can share your pride.

# Appendix D. Answers to Review Questions

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Chapter 10

Chapter 11

Chapter 12

Chapter 14

# Chapter 1

1 The five layers of the TCP/IP protocol suite are the following:

- Physical layer
- Data-link layer
- Internet (or IP) layer
- Host-to-host layer
- Application layer

The *physical layer* contains the protocols of the physical medium.

The *data link layer* contains the protocols that control the physical layer: how the medium is accessed and shared, how devices on the medium are identified, and how data is framed before being transmitted on the medium.

The *internet layer* contains the protocols that define the logical grouping of data links into a network and the communication across that network.

The *host-to-host layer* contains the protocols that define and control the logical, end-to-end paths across the network.

The *application layer* corresponds to the OSI session, presentation, and application layers.

2 The most common IP version now in use is version 4.

3 Routers perform fragmentation when a packet is longer than the maximum packet length (Maximum Transmission Unit, or MTU) supported by a data link onto which the packet must be transmitted. The data within the packet will be broken into fragments, and each fragment will be encapsulated in its own packet. The receiver uses the Identifier and Fragment Offset fields and the MF bit of the Flags field to reassemble the fragments.

4 The Time to Live (TTL) field prevents "lost" packets from being passed endlessly through the IP network. The field contains an 8-bit integer that is set by the originator of the packet. Each router through which the packet passes will decrement the integer by one. If a router decrements the TTL to zero, it will discard the packet and send an ICMP "time exceeded" error message to the packet's source address.

5 The first octet rule determines the class of an IP address as follows:

- Class A: The first bit of the first octet is always 0.
- Class B: The first two bits of the first octet are always 10.
- Class C: The first three bits of the first octet are always 110.
- Class D: The first four bits of the first octet are always 1110.

- Class E: The first four bits of the first octet are always 1111.

6 The A, B, and C IP addresses are recognized in dotted decimal and binary as follows:

| Class | Binary Range of First Octet | Decimal Range of First Octet |
|---|---|---|
| A | 0000000101111110 | 1126 |
| B | 1000000010111111 | 128191 |
| C | 1100000011011111 | 192223 |

7 An IP address mask identifies the network part of an IP address. Each one in the 32-bit mask marks the corresponding bit in the IP address as a network bit. A zero in the mask marks the corresponding bit in the IP address as a host bit. A Boolean AND is performed in all 32 bits of the address and the mask; in the result, all network bits of the mask will be repeated, and all host bits will be changed to zero.

8 A subnet is a subgrouping of a class A, B, or C IP address. Without subnetting, the network part of a major class A, B, or C IP address can only identify a single data link. Subnetting uses some of the host bits of a major IP address as network bits, allowing the single major address to be "subdivided" into multiple network addresses.

9 A classful routing protocol has no way to differentiate between the all-zeros subnet and the major IP address, and between the all-ones subnet and the all-hosts, all-subnets broadcast address of the major IP address.

10 ARP, or Address Resolution Protocol, is a function that maps the IP addresses of interfaces on a data link to their corresponding MAC identifiers.

11 Proxy ARP is a function of an IP router. If the router hears an ARP request, and

- The destination network or subnet is in the router's routing table, and
- The table indicates that the destination is reachable via a different router interface than the one on which the ARP request was received,
- The router will respond to the ARP request with its own MAC address.

12 A redirect is an IP router function. If a device has sent a packet to the router and the router must forward the packet to a next-hop router on the same data link, the router will send a redirect to the originating device. The redirect will inform the device that it can reach the next-hop router directly.

13 TCP, or Transmission Control Protocol, provides a connection-oriented service over the connectionless internet layer. UDP, or User Datagram Service, provides a connectionless service.

14 Correct sequencing is accomplished with sequence numbers. Reliability is accomplished by using checksums, acknowledgments, timers, and retransmissions. Flow control is accomplished by windowing.

15 A MAC identifier is a fixed-length binary integer. If IP used MAC identifiers as the host part of the IP address, subnetting would not be possible because there would be no flexibility in using some of the host bits as network bits.

16 The only purpose of the UDP header is to add fields for the source and destination port numbers.

PREV NEXT

# Chapter 2

1. IPv6 addresses are 128 bits in length.

2. IPv6 addresses are represented as eight 16-bit hexadecimal segments separated by colons.

3. The two rules for compacting IPv6 addresses are

   a. The leading zeroes in any 16-bit segment do not have to be written.

   b. Any single, continuous string of one or more 16-bit segments consisting of all zeroes can be represented with a double colon.

4. Using more than one double colon ambiguates the address; the exact length of each string of zeroes cannot be determined.

5. Both addresses are all zeros. ::/0 is the default address, whereas ::/128 is the unspecified address.

6. The part of a unicast IPv6 address that specifies the host is the Interface ID, and it is usually 64 bits in length.

7. The Subnet ID of the unicast IPv6 address is 16 bits long.

8. An IPv6 address beginning with FF80::/10 is a link-local address.

9. This is a global unicast address, identified by the first three bits of 001.

10. An anycast address is an address that represents a service rather than a device, and can therefore appear on more than one device.

11. A multicast address is an address that represents a group of devices rather than a single device.

12. The IPv6 header is 40 bytes in length.

13. The Flow Label field, by labeling individual flows (packets with the same source and destination address and the same source and destination ports) in the header, is intended to allow highly granular load balancing without having to pay a performance penalty from having to look into the packet payload.

14. The IPv6 Next Header field corresponds to the IPv4 Protocol Number field. It is named differently because the value of the field might specify a following protocol header or it might specify an IPv6 extension header.

15. The Hop Limit field corresponds to the IPv4 Time to Live (TTL) field. The name is changed because routers have never decremented the field according to transit time; rather, every transit router decrements the field by 1, marking a hop instead of a transit time.

16 The IPv6 Next Header field is like the IPv4 Protocol Number field in that it is an 8-bit field that can, if the next header is an upper-layer protocol header, specify the protocol number. But it is different from the Protocol Number field in that it can also specify, if the next header is an IPv6 extension header, that header's type number.

17 Extension headers make the IPv6 header more efficient by being specialized to specific functions and only being included when the specific function is used.

18 The Next Header value of ICMPv6 (corresponding to a Protocol Number) is 58.

19 Aside from the use of the Fragment extension header, the significant difference of IPv6 fragmentation from IPv4 fragmentation is that IPv6 routers do not fragment packets. It is up to the originating host to either fragment packets or ensure that no packet it originates is too large.

20 The five ICMPv6 messages used by NDP are Router Solicitation (RS), Router Advertisement (RA), Neighbor Solicitation (NS), Neighbor Advertisement (NA), and Redirect.

21 The M flag, when set, tells hosts to use DHCPv6 to configure its address. The O flag tells hosts to used DHCPv6 to find other link parameters.

22 The Reachable Timer field specifies the time, in milliseconds, that a node should assume a neighbor is reachable after the node has confirmed reachability.

23 The Retransmit Timer field specifies the period, in milliseconds, that a node should wait between successive transmissions of an NS.

24 A Router Lifetime value of 0 in the RA indicates that the originating router should not be added to a host's default router list.

25 The S flag, when set, indicates that the NA was sent in response to an NS. Two-way reachability is confirmed, and a neighbor address changed to Reachable state in the neighbor cache, only if the NA is in response to a solicitation; so the reception of an NA with the S bit cleared, indicating that it is unsolicited, does not change the state of a neighbor cache entry.

26 Stateful address autoconfiguration relies on DHCPv6 to allocate an address to the host. Stateless address autoconfiguration uses RAs to determine a prefix of larger scope than link-local, plus MAC-to-EUI64 conversion, to determine a host's address.

27 MAC-to-EUI64 conversion inserts a value of 0xFFFE in the middle of a MAC address, then flips the U/L bit to 1, to create a 64-bit Interface ID from a 48-bit MAC address.

28 Duplicate address detection must never be performed on an anycast address.

29 The prefix FF02:0:0:0:0:1:FF00::/104 is used for solicited node multicast addresses. It is prepended to the last 24 bits of an address that is being solicited.

30 IPv6 uses the NDP function Neighbor Address Resolution instead of ARP, and a neighbor cache instead of an ARP cache.

31 A privacy address is one in which the Interface ID is randomly generated, and changed both at some regular period and whenever a host acquires a new prefix. It is used in conjunction with an autoconfigured public address to ensure anonymity of the host. The public address is used for reachability, but the private address is used as the source address of any packets the host originates.

32 An Incomplete state indicates that Neighbor Address Resolution for the entry is in progress.

33 A Probe state indicates that an NS has been sent to verify two-way reachability of a Stale entry, but a responding NA has not yet been received.

34 Neighbor Unreachability Detection verifies two-way reachability of a neighbor either by "hints" from an upper-layer protocol that has received an acknowledgment of a sent message, or by actively probing the neighbor with an NS.

PREV NEXT

# Chapter 3

1. At a minimum, each entry of the routing table must include a destination address and the address of a next-hop router or an indication that the destination address is directly connected.

2. *Variably subnetted* means that the router knows of more than one subnet mask for subnets of the same major IP address.

3. Discontiguous subnets are two or more subnets of a major IP network address that are separated by a different major IP address.

4. `show ip route` is used to examine the routing table of a Cisco router.

5. The command `show ipv6 route` displays the IPv6 route table. Prefixes, prefix lengths, and the next-hop address or outgoing interface are displayed, as are the administrative distance and route metric.

6. The first bracketed number is the administrative distance of the routing protocol by which the route was learned. The second number is the metric of the route.

7. When a static route is configured to reference an exit interface instead of a next-hop address, the destination address will be entered into the routing table as directly connected.

8. A summary route is a single route entry that points to multiple subnets or major IP addresses. In the context of static routes, summary routes can reduce the number of static routes that must be configured.

9. An administrative distance is a rating of preference for a routing protocol or a static route. Every routing protocol and every static route has an administrative distance associated with it. When a router learns of a destination via more than one routing protocol or static route, it will use the route with the lowest administrative distance.

10. A floating static route is an alternative route to a destination. The administrative distance is set high enough that the floating static route is used only if a more-preferred route becomes unavailable.

11. Equal-cost load sharing distributes traffic equally among multiple paths with equal metrics. Unequal-cost load sharing distributes packets among multiple paths with different metrics. The traffic will be distributed inversely proportional to the cost of the routes.

12. If an inbound interface is configured for CEF, the packet will be switched with CEF, and CEF load balancing rules will apply: per destination or per packet for IPv4, depending upon the configuration, and per destination for IPv6. If CEF is not configured on the inbound interface, the outbound interface determines the switching mode, and thus the method of load sharing. If an outbound interface is fast switched, per destination load sharing is performed. If an interface is process switched, per packet load sharing is performed.

13 A recursive routing table lookup occurs when a router cannot acquire all the information it needs to forward a packet with a single routing table lookup. For example, the router may perform one lookup to find the route to a destination and then perform another lookup to find a route to the next-hop router of the first route.

PREV NEXT

# Chapter 4

1 A routing protocol is a "language" that routers speak to each other to share information about network destinations.

2 At a minimum, a routing protocol should define procedures for

- Passing reachability information about networks to other routers.
- Receiving reachability information from other routers.
- Determining optimal routes based on the reachability information it has and for recording this information in a route table.
- Reacting to, compensating for, and advertising topology changes in a network.

3 A route metric, also called a route cost or a route distance, is used to determine the best path to a destination. *Best* is defined by the type of metric used.

4 Convergence time is the time a group of routers takes to complete the exchange of routing information.

5 Load balancing is the process of sending packets over multiple paths to the same destination. Four types of load balancing are

- Equal cost, per packet.
- Equal cost, per destination.
- Unequal cost, per packet.
- Unequal cost, per destination.

6 A distance vector protocol is a routing protocol in which each router calculates routes based on the routes of its neighbors and then passes its routes to other neighbors.

7 Several problems associated with distance vector protocols are

- A susceptibility to incorrect routing information because of its dependence on neighbors for correct information.
- Slow convergence.
- Route loops.
- Counting to infinity.

8 Neighbors are routers connected to the same data link.

9 Route invalidation timers delete routes from a route table if they exceed a certain age.

10 Simple split horizon does not send route information back to the source of the route information. Split horizon with poisoned reverse sends the information back to the source but sets the metric to unreachable.

11 Counting to infinity occurs when routes update a route over a loop; each router increases the metric of the route until the metric reaches infinity. The effects of counting to infinity are controlled by defining *infinity* as a fairly low metric so that infinity is reached fairly quickly and the route is declared unreachable.

12 Holddown timers help prevent routing loops. If a route is declared unreachable or if the metric increases beyond a certain threshold, a router will not accept any other information about that route until the holddown timer expires. This approach prevents the router from accepting possibly bad routing information while the network is reconverging.

13 A distance vector router sends its entire route table, but it only sends the table to directly connected neighbors. A link state router sends only information about its directly connected links, but it floods the information throughout the networking area. Distance vector protocols usually use a variant of the Bellman-Ford algorithm to calculate routes, and link state protocols usually use a variant of the Dijkstra algorithm to calculate routes.

14 A topological database holds the link state information originated by all routers in the link state routing domain.

15 Each router floods a link state information advertisement describing its links, the states of its links, and any neighboring routers connected to those links, throughout the networking area. All routers store all received copies of the link state advertisement in a link state database. Each router calculates a shortest path tree from the information in the topological database and enters routes in its routing tables based on the shortest path tree.

16 Sequence numbers help a router differentiate between multiple copies of the same link state advertisement and also prevent flooded link state advertisements from circulating endlessly throughout the network.

17 Aging prevents old, possibly obsolete, link state information from residing in a topological database or from being accepted by a router.

18 A router builds a shortest path tree by first adding itself as the root. Using the information in the topological database, the router creates a list of all of its directly connected neighbors. The lowest-cost link to a neighbor becomes a branch of the tree, and that router's neighbors are added to the list. The list is checked for duplicate paths, and if they exist, the higher-cost paths are removed from the list. The lowest-cost router on the list is added to the tree, that router's neighbors are added to the list, and the list is again checked for duplicate paths. This process continues until no routers remain on the list.

19 Within a routing domain, areas are subdomains. They make link state routing more efficient by limiting the size of the link state database of each router in the area.

20 Depending on the usage, an autonomous system can be defined as a network under a common administrative domain or a single routing domain.

21 An Interior Gateway Protocol is a routing protocol that routes within an autonomous system. An Exterior Gateway Protocol is a routing protocol that routes between autonomous systems.

# Chapter 5

1. RIP uses UDP port 520.
2. RIP uses a hop count metric. An unreachable network is indicated by setting the hop count to 16, which RIP interprets as an infinite distance.
3. RIP sends periodic updates every 30 seconds minus a small random variable that prevents the updates of neighboring routers from becoming synchronized.
4. A route entry is marked as unreachable if six updates are missed.
5. The garbage collection timer, or flush timer, is set when a route is declared unreachable. When the timer expires, the route is flushed from the route table. This process allows an unreachable route to remain in the routing table long enough for neighbors to be notified of its status.
6. The random timer, whose range is one to five seconds, prevents a "storm" of triggered updates during a topology change.
7. A Request message asks a router for an update. A Response message is an update.
8. A Request message might either ask for a full update, or in some special cases, it might ask for specific routes.
9. A Response is sent when the update timer expires, or upon reception of a Request message.
10. RIP updates do not include the subnet mask of the destination address, so a RIP router depends on the subnet masks of its own interfaces to determine how an attached major network address is subnetted. If a router does not have an attachment to a particular major network address, it has no way to know how that major network is subnetted. Therefore, no subnets of a major network address can be advertised into another major network.

# Chapter 6

1. The Route Tag field, the Subnet Mask field, and the Next Hop field are RIPv2 extensions that do not exist in RIPv1 messages. The basic format of the RIP message remains unchanged between the two versions; Version 2 merely uses fields that are unused in Version 1.

2. In addition to the functions that use the new fields, RIPv2 supports authentication and multicast updates.

3. RIPv2 uses the multicast address 224.0.0.9. Multicasting of routing messages is better than broadcasting because hosts and non-RIPv2 routers will ignore the multicast messages.

4. When another routing protocol uses the RIPv2 domain as a transit domain, the protocol external to RIPv2 can use the Route Tag field to communicate information to its peers on the other side of the RIPv2 domain.

5. The Next Hop field is used to inform other routers of a next-hop address on the same multiaccess network that is metrically closer to the destination than the originating router.

6. RIPv2 uses the same UDP port number as RIPv1, port number 520.

7. RIPng uses the UDP port number 521.

8. A classless routing protocol does not consider the major network address in its route lookups, but just looks for the longest match.

9. To support VLSM, a routing protocol must be able to include the subnet mask of each destination address in its updates.

10. The Cisco implementation of RIPv2 supports clear-text authentication and MD5 authentication. Only clear-text authentication is defined in RFC 2453.

# Chapter 7

1. EIGRP is a distance vector protocol.

2. By default, EIGRP uses no more than 50 percent of the link's bandwidth, based on the bandwidth configured on the router's interface. This percentage can be changed with the command `ip bandwidth-percent eigrp`.

3. EIGRP and IGRP use the same formula to calculate their composite metrics, but EIGRP scales the metric by a factor of 256.

   The four basic components of EIGRP are

   - The Protocol Dependent Modules
   - The Reliable Transport Protocol
   - The Neighbor Discovery and Recovery Module
   - The Diffusing Update Algorithm

4. Reliable delivery means EIGRP packets are guaranteed to be delivered, and they are delivered in order. RTP uses a reliable multicast, in which received packets are acknowledged, to guarantee delivery; sequence numbers are used to ensure that they are delivered in order.

5. Sequence numbers ensure that a router is receiving the most recent route entry.

6. EIGRP uses the multicast address 224.0.0.10.

7. The packet types used by EIGRP are

   - Hellos
   - Acknowledgments
   - Updates
   - Queries
   - Replies
   - SIA-Queries
   - SIA-Replies

8. The default EIGRP Hello interval is 5 seconds, except on some slow-speed (T1 and below) interfaces, where the default is 60 seconds.

9. The EIGRP default hold time is three times the Hello interval.

10. The neighbor table stores information about EIGRP-speaking neighbors; the topology table lists all known routes that have feasible successors.

11 The feasible distance to a destination is a router's lowest calculated distance to the destination.

12 The feasibility condition is the rule by which feasible successors are chosen for a destination. The feasibility condition is satisfied if a neighbor's advertised distance to a destination is lower than the receiving router's feasible distance to the destination. In other words, a router's neighbor meets the feasibility condition if the neighbor is metrically closer to the destination than the router. Another way to describe this is that the neighbor is "downstream" relative to the destination.

13 A feasible successor to a destination is a neighbor that satisfies the feasibility condition for that destination.

14 A successor to a destination is a feasible successor that is currently being used as the next hop to the destination.

15 A route is active on a particular router if the router has queried its neighbors for a feasible successor and has not yet received a reply from every queried neighbor. The route is passive when there are no outstanding queries.

16 A route becomes active when no feasible successor exists in its topology table.

17 An active route becomes passive when a reply has been received from every queried neighbor.

18 If a router does not receive a reply from a queried neighbor within the active time (three minutes, by default), the route is declared stuck-in-active. A response with an infinite metric is entered on the neighbor's behalf to satisfy DUAL, and the neighbor is deleted from the neighbor table.

19 Subnetting is the practice of creating a group of subnet addresses from a single IP network address. Address aggregation is the practice of summarizing a group of network or subnet addresses with a single IP network address.

PREV NEXT

# Chapter 8

1. From the perspective of an OSPF router, a neighbor is another OSPF router that is attached to one of the first router's directly connected links.

2. An OSPF adjacency is a conceptual link to a neighbor over which LSAs can be sent.

3. The five OSPF packet types, and their purposes, are

    - Hellos, which are used to discover neighbors, and to establish and maintain adjacencies
    - Updates, which are used to send LSAs between neighbors
    - Database Description packets, which a router uses to describe its link state database to a neighbor during database synchronization
    - Link State Requests, which a router uses to request one or more LSAs from a neighbor's link-state database
    - Link State Acknowledgments, used to ensure reliable delivery of LSAs

4. A router originates a link-state advertisement to describe one or more destinations. An OSPF Update packet transports LSAs from one neighbor to another. Although LSAs are flooded throughout an area or OSPF domain, Update packets never leave a data link.

5. The most common LSA types and their purposes are

    - Type 1 (Router LSAs) are originated by every router and describe the originating router, the router's directly connected links and their states, and the router's neighbors.
    - Type 2 (Network LSAs) are originated by Designated Routers on multiaccess links and describe the link and all attached neighbors.
    - Type 3 (Network Summary LSAs) are originated by Area Border Routers and describe inter-area destinations.
    - Type 4 LSAs (ASBR Summary LSAs) are originated by Area Border Routers to describe Autonomous System Boundary Routers outside the area.
    - Type 5 (AS External LSAs) are originated by Autonomous System Boundary Routers to describe destinations external to the OSPF domain.
    - Type 7 (NSSA External LSAs) are originated by Autonomous System Boundary Routers within not-so-stubby areas.

6. The link state database is where a router stores all the OSPF LSAs it knows of, including its own. Database synchronization is the process of ensuring that all routers within an area have identical link-state databases.

7. The default OSPF HelloInterval is 10 seconds.

8 The default RouterDeadInterval is four times the HelloInterval.

9 A Router ID is an address by which an OSPF router identifies itself. It is either the numerically highest IP address of all the router's loopback interfaces, or if no loopback interfaces are configured, it is the numerically highest IP address of all the router's LAN interfaces. It can also be manually configured.

10 An area is an OSPF sub-domain, within which all routers have an identical link-state database.

11 Area 0 is the backbone area. All other areas must send their inter-area traffic through the backbone.

12 MaxAge, one hour, is the age at which an LSA is considered to be obsolete.

13 The four OSPF router types are

- Internal Routers, whose OSPF interfaces all belong to the same area
- Backbone Routers, which are Internal Routers in Area 0
- Area Border Routers, which have OSPF interfaces in more than one area
- Autonomous System Boundary Routers, which advertise external routes into the OSPF domain

14 The four OSPF path types are

- Intra-area paths
- Inter-area paths
- Type 1 external paths
- Type 2 external paths

15 What are the five OSPF network types?

The five OSPF network types are

- Point-to-point networks
- Broadcast networks
- Non-broadcast multiaccess (NBMA) networks
- Point-to-multipoint networks
- Virtual links

16 A Designated Router is a router that represents a multiaccess network, and the routers connected to the network, to the rest of the OSPF domain.

17 Cisco IOS calculates the outgoing cost of an interface as $10^8$/BW, where BW is the configured bandwidth of the interface. $10^8$ can be changed with the OSPF command `auto-cost reference-bandwidth`.

18 An area is partitioned if one or more of its routers cannot send a packet to the area's other routers without sending the packet out of the area.

19 A virtual link is a tunnel that extends an OSPF backbone connection through a non-backbone area.

20 A stub area is an area into which no type 5 LSAs are flooded. A totally stubby area is an area into which no type 3, 4, or 5 LSAs are flooded, with the exception of type 3 LSAs to advertise a default route. Not-so-stubby areas are areas through which external destinations are advertised into the OSPF domain, but into which no type 5 LSAs are sent by the ABR.

21 OSPF network entries are entries in the route table, describing IP destinations. OSPF router entries are entries in a separate route table that record only routes to ABRs and ASBRs.

22 Type 2 authentication uses MD5 encryption, whereas type 1 authentication uses clear-text passwords.

23 The three fields in the LSA header that distinguish different LSAs are the Type, Advertising Router, and the Link State ID fields. The three fields in the LSA header that distinguish different instances of the same LSA are the Sequence Number, Age, and Checksum fields.

PREV NEXT

# Chapter 9

1. OSPFv3 cannot support IPv4 at the time of this writing. To route both IPv4 and IPv6 with OSPF, you must run both OSPFv2 and OSPFv3.

2. Multiple instance per link means that there can be separate adjacencies among different routers all connected to the same broadcast link, so that different OSPFv3 routing domains can use the same shared link without interfering with or having knowledge of each other. The Instance ID field in the OSPFv3 header makes this possible.

3. OSPFv3 packets are authenticated using the built-in IPv6 authentication (by means of the IPv6 Authentication extension header). OSPFv3 does not have its own authentication mechanism as OSPFv2 does.

4. The OSPFv3 Next Header number is the same as the OSPFv2 Protocol Number, 89.

5. OSPFv3 uses the reserved multicast addresses FF02::5 (AllSPFRouters) and FF02::6 (AllDRouters).

6. No. OSPFv3 uses the same five message types as OSPFv2.

7. The first bit is the U bit, specifying how the receiving router should treat the LSA if its type is unknown. The second and third bits are the S bits, indicating the flooding scope of the LSA.

8. OSPFv3 supports a link-local flooding scope that is not supported by OSPFv2. The Link LSA uses this flooding scope.

9. OSPFv3 Router and Network LSAs do not advertise prefixes, as OSPFv2 Router and Network LSAs do.

10. The Intra-Area Prefix LSA carries IPv6 prefixes connected to the originating router.

11. The Link LSA carries information that is only significant between two directly connected neighbors.

# Chapter 10

1 An Intermediate System is the ISO term for a router.

2 A Network Protocol Data Unit is the ISO term for a packet.

3 An L1 router has no direct connections to another area. An L2 router has no adjacencies with L1 routers. An L1/L2 router routes both inter-area and intra-area traffic and acts as an inter-area gateway for L1 routers.

4 Cisco routers by default are L1/L2.

5 The borders of IS-IS areas are between routers, on links. The borders of OSPF areas are defined by the routers themselves.

6 Two L1/L2 routers with the same AIDs will form both an L1 and an L2 adjacency. Two L1/L2 routers with different AIDs will form an L2 adjacency.

7 Two L2-only routers will form an L2 adjacency, whether the AIDs are the same or different.

8 The Network Entity Title is an address by which a router identifies both itself and the area in which it resides.

9 The NSAP Selector should be set to 0x00 in a NET.

10 The System ID uniquely identifies a router within an IS-IS domain.

11 The portion of the NET preceding the last seven octets is the area address.

12 IS-IS does not elect a BDR.

13 The Pseudonode ID is the last octet of a LAN ID. Its purpose is to distinguish LAN IDs that are originated by a single router that is the DR on multiple LANs.

14 The MaxAge of an IS-IS LSP is 1200 seconds (20 minutes). The MaxAge (or beginning Remaining Lifetime) can be configured up to 65,535 seconds.

15 OSPF increments the age up to MaxAge; IS-IS decrements the age down to 0. A new OSPF LSA has an age of 0, whereas a new IS-IS LSP has an age of MaxAge.

16 The refresh rate of an IS-IS router is 900 seconds (15 minutes).

17 A Complete Sequence Number Packet contains a full listing of all LSPs in a database. A CSNP is periodically sent by the Designated Router on a broadcast network to maintain database synchronization.

18 A Partial Sequence Number Packet contains a listing of one or more LSPs. It has two uses: On point-to-point networks, it is used to acknowledge the receipt of LSPs. On broadcast networks, it is used to request LSPs.

19 An IS-IS router uses the Overload bit to inform its neighbors that it is experiencing a memory overload and cannot store the entire link-state database.

20 The Attached bit is used by L1/L2 routers to inform L1 routers that it is attached to the L2 backbone.

21 The Up/down bit is used to distinguish between an address that originated within an area, or an address that was leaked into an area.

22 The ISO specifies four metrics: Default, Expense, Delay, and Error. Cisco supports only the Default metric.

23 The two metric styles are narrow and wide. The narrow metric has a maximum value of 63. The wide metric has a maximum value of 16777214.

24 The maximum metric value of an IS-IS route is 1023 for narrow metrics and 4261412864 for wide metrics.

25 L1 IS-IS metrics apply to intra-area routes, and L2 IS-IS metrics apply to inter-area routes.

26 Internal metrics apply to routes to destinations within the IS-IS domain. External metrics apply to routes to destinations external to the IS-IS domain.

27 A single adjacency is formed between two routers, even if both IPv4 and IPv6 are configured in multi-topology mode.

28 L1 areas may be configured on a router. One L2 area is configured.

29 The two active mesh group modes are Blocked and Set (Numbered). Blocked mode offers the most reduced flooding, but at the possible cost of the most reduced redundancy and increased convergence time. Set mode, or numbered mesh groups, do not reduce flooding load as much as Blocked mode, but also have less potential impact on redundancy and convergence time.

PREV NEXT

# Chapter 11

1. Routes that are learned from another routing protocol, between two processes of the same routing protocol, from static routes, or from a direct connection to the destination network can be redistributed into a routing domain. Routes can also be redistributed between IS-IS levels 1 and 2.

2. In contrast to metrics, which are used to determine the best path among multiple routes to the same destination discovered by the same routing protocol, administrative distances are used to determine the best path among multiple routes to the same destination discovered by different routing protocols.

3. A route to a destination within a routing domain with a higher administrative distance can be redistributed into a routing domain with a lower administrative distance. If that route is redistributed back into the higher-distance domain, packets might be misrouted into the lower-distance domain.

4. Redistributing variably subnetted destination addresses from a classless domain into a classful domain can cause problems. The classful domain might not be able to recognize all the subnets attempting to be redistributed from the classless domain.

5. OSPF and IS-IS understand the default metric. RIP, IGRP, and EIGRP do not.

6. The `metric` command assigns a metric to specific redistribution statements. The `default-metric` command assigns a metric to all redistribution commands that do not include the `metric` command.

7. Without the `subnets` keyword, only major network addresses that are not directly connected to the router will be redistributed.

8. A router that originates a summary route should use the null interface as the next hop of the summary route. Any packets that match the summary route, but for which there is no more-specific route to the packet's destination address, will be dropped. This prevents the router from forwarding "lost" packets.

# Chapter 12

1. The IPv4 default route address is 0.0.0.0.
2. The IPv6 default prefix/prefix length is ::/0.
3. EIGRP advertises a default address as an external address type.
4. Yes.
5. A stub router is a router with only a single link to another router. A stub network is a network with only one attached router.
6. Using a default route rather than a full route table can conserve router memory by keeping the table small and can save router processing cycles by limiting the routing information that must be processed.
7. Using a full route table rather than a default route can make routing more accurate.
8. ODR uses Cisco Discovery Protocol (CDP) to discover routes.
9. ODR is available in IOS 11.2 and later.
10. The medium over which ODR is to run must support SNAP.

# Chapter 14

1. Route maps are similar to access lists in that they define match criteria and an action to take in the event of a match. Route maps are different from access lists in that they not only specify match criteria but also specify set criteria. The set action can modify a route, or route a packet according to the parameters of the packet.

2. Policy routes are static routes that use route maps to determine which packets should be routed and where the packets should be routed.

3. Route tags are fields within routing information packets that allow external information to be carried through the routing domain.

4. Route tags have no effect on the routing protocols that carry them.

# Appendix E. Solutions to Configuration Exercises

Chapter 1

Chapter 3

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Chapter 10

Chapter 11

Chapter 13

Chapter 14

# Chapter 1

1 If the first four bits of a class D address are 1110, the lowest first octet is 11100000 and the highest first octet is 11101111. In decimal these two numbers are 224 and 239, respectively. Therefore, the first octet of a class D address will range from 224 to 239.

2 (a) Enough subnet bits *n* are needed so that $2^n$ 2 > = 16,000. There must be enough host bits *h* remaining so that $2^h$ 2 > = 700. A subnet mask of 255.255.252.0 provides 16,382 subnets of the class A address, and 1022 host addresses on each subnet. This mask is the only one that works. If one more bit is used for subnetting (255.255.254.0), there will not be enough host addresses. If one less bit is used for subnetting (255.255.248.0), there will not be enough subnets.

(b) Enough subnet bits *n* are needed so that $2^n$ 2 > = 500. Enough host bits *h* must remain so that $2^h$ 2 > = 100. A subnet mask of 255.255.255.128 provides 510 subnets of the class B address and 126 host addresses on each subnet. Again, this mask is the only one that works.

3 With six bits of subnetting, a class C address will have $2^6$ 2 = 62 subnets and $2^2$ 2 = 2 host addresses per subnet. With this subnetting scheme, a single class C address can be used for 62 point-to-point links. A point-to-point link requires only two host addressesone for each end of the link.

4 A class C address with a 28-bit mask will have 14 subnets and 14 host addresses on each subnet. The subnets are derived first.

The subnets are

```
11111111111111111111111111110000  =  255.255.255.240   (mask)
11000000101010001001001100010000  =  192.168.147.16
11000000101010001001001100100000  =  192.168.147.32
11000000101010001001001100110000  =  192.168.147.48
11000000101010001001001101000000  =  192.168.147.64
11000000101010001001001101010000  =  192.168.147.80
11000000101010001001001101100000  =  192.168.147.96
11000000101010001001001101110000  =  192.168.147.112
11000000101010001001001110000000  =  192.168.147.128
11000000101010001001001110010000  =  192.168.147.144
11000000101010001001001110100000  =  192.168.147.160
11000000101010001001001110110000  =  192.168.147.176
11000000101010001001001111000000  =  192.168.147.192
11000000101010001001001111010000  =  192.168.147.208
11000000101010001001001111100000  =  192.168.147.224
```

Next, the host addresses for each subnet are derived. The broadcast addresses for each subnet are also shown.

The host addresses for each subnet are

```
11000000101010001001001100010000  =  192.168.147.16   (subnet)
11000000101010001001001100010001  =  192.168.147.17
11000000101010001001001100010010  =  192.168.147.18
```

| | | | |
|---|---|---|---|
| 11000000101010001001001100010011 | = | 192.168.147.19 | |
| 11000000101010001001001100010100 | = | 192.168.147.20 | |
| 11000000101010001001001100010101 | = | 192.168.147.21 | |
| 11000000101010001001001100010110 | = | 192.168.147.22 | |
| 11000000101010001001001100010111 | = | 192.168.147.23 | |
| 11000000101010001001001100011000 | = | 192.168.147.24 | |
| 11000000101010001001001100011001 | = | 192.168.147.25 | |
| 11000000101010001001001100011010 | = | 192.168.147.26 | |
| 11000000101010001001001100011011 | = | 192.168.147.27 | |
| 11000000101010001001001100011100 | = | 192.168.147.28 | |
| 11000000101010001001001100011101 | = | 192.168.147.29 | |
| 11000000101010001001001100011110 | = | 192.168.147.30 | |
| 11000000101010001001001100011111 | = | 192.168.147.31 | (broadcast) |
| 11000000101010001001001100100000 | = | 192.168.147.32 | (subnet) |
| 11000000101010001001001100100001 | = | 192.168.147.33 | |
| 11000000101010001001001100100010 | = | 192.168.147.34 | |
| 11000000101010001001001100100011 | = | 192.168.147.35 | |
| 11000000101010001001001100100100 | = | 192.168.147.36 | |
| 11000000101010001001001100100101 | = | 192.168.147.37 | |
| 11000000101010001001001100100110 | = | 192.168.147.38 | |
| 11000000101010001001001100100111 | = | 192.168.147.39 | |
| 11000000101010001001001100101000 | = | 192.168.147.40 | |
| 11000000101010001001001100101001 | = | 192.168.147.41 | |
| 11000000101010001001001100101010 | = | 192.168.147.42 | |
| 11000000101010001001001100101011 | = | 192.168.147.43 | |
| 11000000101010001001001100101100 | = | 192.168.147.44 | |
| 11000000101010001001001100101101 | = | 192.168.147.45 | |
| 11000000101010001001001100101110 | = | 192.168.147.46 | |
| 11000000101010001001001100101111 | = | 192.168.147.47 | (broadcast) |
| 11000000101010001001001100110000 | = | 192.168.147.48 | (subnet) |
| 11000000101010001001001100110001 | = | 192.168.147.49 | |
| 11000000101010001001001100110010 | = | 192.168.147.50 | |
| 11000000101010001001001100110011 | = | 192.168.147.51 | |
| 11000000101010001001001100110100 | = | 192.168.147.52 | |
| 11000000101010001001001100110101 | = | 192.168.147.53 | |
| 11000000101010001001001100110110 | = | 192.168.147.54 | |
| 11000000101010001001001100110111 | = | 192.168.147.55 | |
| 11000000101010001001001100111000 | = | 192.168.147.56 | |
| 11000000101010001001001100111001 | = | 192.168.147.57 | |
| 11000000101010001001001100111010 | = | 192.168.147.58 | |
| 11000000101010001001001100111011 | = | 192.168.147.59 | |
| 11000000101010001001001100111100 | = | 192.168.147.60 | |
| 11000000101010001001001100111101 | = | 192.168.147.61 | |
| 11000000101010001001001100111110 | = | 192.168.147.62 | |
| 11000000101010001001001100111111 | = | 192.168.147.63 | (broadcast) |
| 11000000101010001001001101000000 | = | 192.168.147.64 | (subnet) |
| 11000000101010001001001101000001 | = | 192.168.147.65 | |
| 11000000101010001001001101000010 | = | 192.168.147.66 | |
| 11000000101010001001001101000011 | = | 192.168.147.67 | |
| 11000000101010001001001101000100 | = | 192.168.147.68 | |
| 11000000101010001001001101000101 | = | 192.168.147.69 | |
| 11000000101010001001001101000110 | = | 192.168.147.70 | |
| 11000000101010001001001101000111 | = | 192.168.147.71 | |
| 11000000101010001001001101001000 | = | 192.168.147.72 | |
| 11000000101010001001001101001001 | = | 192.168.147.73 | |
| 11000000101010001001001101001010 | = | 192.168.147.74 | |
| 11000000101010001001001101001011 | = | 192.168.147.75 | |
| 11000000101010001001001101001100 | = | 192.168.147.76 | |
| 11000000101010001001001101001101 | = | 192.168.147.77 | |
| 11000000101010001001001101001110 | = | 192.168.147.78 | |
| 11000000101010001001001101001111 | = | 192.168.147.79 | (broadcast) |

| | | | |
|---|---|---|---|
| 11000000101010001001001101010000 | = | 192.168.147.80 | (subnet) |
| 11000000101010001001001101010001 | = | 192.168.147.81 | |
| 11000000101010001001001101010010 | = | 192.168.147.82 | |
| 11000000101010001001001101010011 | = | 192.168.147.83 | |
| 11000000101010001001001101010100 | = | 192.168.147.84 | |
| 11000000101010001001001101010101 | = | 192.168.147.85 | |
| 11000000101010001001001101010110 | = | 192.168.147.86 | |
| 11000000101010001001001101010111 | = | 192.168.147.87 | |
| 11000000101010001001001101011000 | = | 192.168.147.88 | |
| 11000000101010001001001101011001 | = | 192.168.147.89 | |
| 11000000101010001001001101011010 | = | 192.168.147.90 | |
| 11000000101010001001001101011011 | = | 192.168.147.91 | |
| 11000000101010001001001101011100 | = | 192.168.147.92 | |
| 11000000101010001001001101011101 | = | 192.168.147.93 | |
| 11000000101010001001001101011110 | = | 192.168.147.94 | |
| 11000000101010001001001101011111 | = | 192.168.147.95 | (broadcast) |
| 11000000101010001001001101100000 | = | 192.168.147.96 | (subnet) |
| 11000000101010001001001101100001 | = | 192.168.147.97 | |
| 11000000101010001001001101100010 | = | 192.168.147.98 | |
| 11000000101010001001001101100011 | = | 192.168.147.99 | |
| 11000000101010001001001101100100 | = | 192.168.147.100 | |
| 11000000101010001001001101100101 | = | 192.168.147.101 | |
| 11000000101010001001001101100110 | = | 192.168.147.102 | |
| 11000000101010001001001101100111 | = | 192.168.147.103 | |
| 11000000101010001001001101101000 | = | 192.168.147.104 | |
| 11000000101010001001001101101001 | = | 192.168.147.105 | |
| 11000000101010001001001101101010 | = | 192.168.147.106 | |
| 11000000101010001001001101101011 | = | 192.168.147.107 | |
| 11000000101010001001001101101100 | = | 192.168.147.108 | |
| 11000000101010001001001101101101 | = | 192.168.147.109 | |
| 11000000101010001001001101101110 | = | 192.168.147.110 | |
| 11000000101010001001001101101111 | = | 192.168.147.111 | (broadcast) |
| 11000000101010001001001101110000 | = | 192.168.147.112 | (subnet) |
| 11000000101010001001001101110001 | = | 192.168.147.113 | |
| 11000000101010001001001101110010 | = | 192.168.147.114 | |
| 11000000101010001001001101110011 | = | 192.168.147.115 | |
| 11000000101010001001001101110100 | = | 192.168.147.116 | |
| 11000000101010001001001101110101 | = | 192.168.147.117 | |
| 11000000101010001001001101110110 | = | 192.168.147.118 | |
| 11000000101010001001001101110111 | = | 192.168.147.119 | |
| 11000000101010001001001101111000 | = | 192.168.147.120 | |
| 11000000101010001001001101111001 | = | 192.168.147.121 | |
| 11000000101010001001001101111010 | = | 192.168.147.122 | |
| 11000000101010001001001101111011 | = | 192.168.147.123 | |
| 11000000101010001001001101111100 | = | 192.168.147.124 | |
| 11000000101010001001001101111101 | = | 192.168.147.125 | |
| 11000000101010001001001101111110 | = | 192.168.147.126 | |
| 11000000101010001001001101111111 | = | 192.168.147.127 | (broadcast) |
| 11000000101010001001001110000000 | = | 192.168.147.128 | (subnet) |
| 11000000101010001001001110000001 | = | 192.168.147.129 | |
| 11000000101010001001001110000010 | = | 192.168.147.130 | |
| 11000000101010001001001110000011 | = | 192.168.147.131 | |
| 11000000101010001001001110000100 | = | 192.168.147.132 | |
| 11000000101010001001001110000101 | = | 192.168.147.133 | |
| 11000000101010001001001110000110 | = | 192.168.147.134 | |
| 11000000101010001001001110000111 | = | 192.168.147.135 | |
| 11000000101010001001001110001000 | = | 192.168.147.136 | |
| 11000000101010001001001110001001 | = | 192.168.147.137 | |
| 11000000101010001001001110001010 | = | 192.168.147.138 | |
| 11000000101010001001001110001011 | = | 192.168.147.139 | |
| 11000000101010001001001110001100 | = | 192.168.147.140 | |

| | | | |
|---|---|---|---|
| 11000000101010001001001110001101 | = | 192.168.147.141 | |
| 11000000101010001001001110001110 | = | 192.168.147.142 | |
| 11000000101010001001001110001111 | = | 192.168.147.143 | (broadcast) |
| 11000000101010001001001110010000 | = | 192.168.147.144 | (subnet) |
| 11000000101010001001001110010001 | = | 192.168.147.145 | |
| 11000000101010001001001110010010 | = | 192.168.147.146 | |
| 11000000101010001001001110010011 | = | 192.168.147.147 | |
| 11000000101010001001001110010100 | = | 192.168.147.148 | |
| 11000000101010001001001110010101 | = | 192.168.147.149 | |
| 11000000101010001001001110010110 | = | 192.168.147.150 | |
| 11000000101010001001001110010111 | = | 192.168.147.151 | |
| 11000000101010001001001110011000 | = | 192.168.147.152 | |
| 11000000101010001001001110011001 | = | 192.168.147.153 | |
| 11000000101010001001001110011010 | = | 192.168.147.154 | |
| 11000000101010001001001110011011 | = | 192.168.147.155 | |
| 11000000101010001001001110011100 | = | 192.168.147.156 | |
| 11000000101010001001001110011101 | = | 192.168.147.157 | |
| 11000000101010001001001110011110 | = | 192.168.147.158 | |
| 11000000101010001001001110011111 | = | 192.168.147.159 | (broadcast) |
| 11000000101010001001001110100000 | = | 192.168.147.160 | (subnet) |
| 11000000101010001001001110100001 | = | 192.168.147.161 | |
| 11000000101010001001001110100010 | = | 192.168.147.162 | |
| 11000000101010001001001110100011 | = | 192.168.147.163 | |
| 11000000101010001001001110100100 | = | 192.168.147.164 | |
| 11000000101010001001001110100101 | = | 192.168.147.165 | |
| 11000000101010001001001110100110 | = | 192.168.147.166 | |
| 11000000101010001001001110100111 | = | 192.168.147.167 | |
| 11000000101010001001001110101000 | = | 192.168.147.168 | |
| 11000000101010001001001110101001 | = | 192.168.147.169 | |
| 11000000101010001001001110101010 | = | 192.168.147.170 | |
| 11000000101010001001001110101011 | = | 192.168.147.171 | |
| 11000000101010001001001110101100 | = | 192.168.147.172 | |
| 11000000101010001001001110101101 | = | 192.168.147.173 | |
| 11000000101010001001001110101110 | = | 192.168.147.174 | |
| 11000000101010001001001110101111 | = | 192.168.147.175 | (broadcast) |
| 11000000101010001001001110110000 | = | 192.168.147.176 | (subnet) |
| 11000000101010001001001110110001 | = | 192.168.147.177 | |
| 11000000101010001001001110110010 | = | 192.168.147.178 | |
| 11000000101010001001001110110011 | = | 192.168.147.179 | |
| 11000000101010001001001110110100 | = | 192.168.147.180 | |
| 11000000101010001001001110110101 | = | 192.168.147.181 | |
| 11000000101010001001001110110110 | = | 192.168.147.182 | |
| 11000000101010001001001110110111 | = | 192.168.147.183 | |
| 11000000101010001001001110111000 | = | 192.168.147.184 | |
| 11000000101010001001001110111001 | = | 192.168.147.185 | |
| 11000000101010001001001110111010 | = | 192.168.147.186 | |
| 11000000101010001001001110111011 | = | 192.168.147.187 | |
| 11000000101010001001001110111100 | = | 192.168.147.188 | |
| 11000000101010001001001110111101 | = | 192.168.147.189 | |
| 11000000101010001001001110111110 | = | 192.168.147.190 | |
| 11000000101010001001001110111111 | = | 192.168.147.191 | (broadcast) |
| 11000000101010001001001111000000 | = | 192.168.147.192 | (subnet) |
| 11000000101010001001001111000001 | = | 192.168.147.193 | |
| 11000000101010001001001111000010 | = | 192.168.147.194 | |
| 11000000101010001001001111000011 | = | 192.168.147.195 | |
| 11000000101010001001001111000100 | = | 192.168.147.196 | |
| 11000000101010001001001111000101 | = | 192.168.147.197 | |
| 11000000101010001001001111000110 | = | 192.168.147.198 | |
| 11000000101010001001001111000111 | = | 192.168.147.199 | |
| 11000000101010001001001111001000 | = | 192.168.147.200 | |
| 11000000101010001001001111001001 | = | 192.168.147.201 | |

```
11000000101010001001001111001010  =  192.168.147.202
11000000101010001001001111001011  =  192.168.147.203
11000000101010001001001111001100  =  192.168.147.204
11000000101010001001001111001101  =  192.168.147.205
11000000101010001001001111001110  =  192.168.147.206
11000000101010001001001111001111  =  192.168.147.207   (broadcast)
11000000101010001001001111010000  =  192.168.147.208   (subnet)
11000000101010001001001111010001  =  192.168.147.209
11000000101010001001001111010010  =  192.168.147.210
11000000101010001001001111010011  =  192.168.147.211
11000000101010001001001111010100  =  192.168.147.212
11000000101010001001001111010101  =  192.168.147.213
11000000101010001001001111010110  =  192.168.147.214
11000000101010001001001111010111  =  192.168.147.215
11000000101010001001001111011000  =  192.168.147.216
11000000101010001001001111011001  =  192.168.147.217
11000000101010001001001111011010  =  192.168.147.218
11000000101010001001001111011011  =  192.168.147.219
11000000101010001001001111011100  =  192.168.147.220
11000000101010001001001111011101  =  192.168.147.221
11000000101010001001001111011110  =  192.168.147.222
11000000101010001001001111011111  =  192.168.147.223   (broadcast)
11000000101010001001001111100000  =  192.168.147.224   (subnet)
11000000101010001001001111100001  =  192.168.147.225
11000000101010001001001111100010  =  192.168.147.226
11000000101010001001001111100011  =  192.168.147.227
11000000101010001001001111100100  =  192.168.147.228
11000000101010001001001111100101  =  192.168.147.229
11000000101010001001001111100110  =  192.168.147.230
11000000101010001001001111100111  =  192.168.147.231
11000000101010001001001111101000  =  192.168.147.232
11000000101010001001001111101001  =  192.168.147.233
11000000101010001001001111101010  =  192.168.147.234
11000000101010001001001111101011  =  192.168.147.235
11000000101010001001001111101100  =  192.168.147.236
11000000101010001001001111101101  =  192.168.147.237
11000000101010001001001111101110  =  192.168.147.238
11000000101010001001001111101111  =  192.168.147.239   (broadcast)
```

5 This solution shows a shorter technique than the last solution in which every host address of every subnet was written out.

A class C address with a 29-bit mask means that there will be 30 subnets and 6 host addresses on each subnet.

The subnets are

```
11111111111111111111111111111000  =  255.255.255.248   (mask)
11000000101010001001001100001000  =  192.168.147.8
11000000101010001001001100010000  =  192.168.147.16
11000000101010001001001100011000  =  192.168.147.24
11000000101010001001001100100000  =  192.168.147.32
11000000101010001001001100101000  =  192.168.147.40
11000000101010001001001100110000  =  192.168.147.48
11000000101010001001001100111000  =  192.168.147.56
11000000101010001001001101000000  =  192.168.147.64
11000000101010001001001101001000  =  192.168.147.72
11000000101010001001001101010000  =  192.168.147.80
11000000101010001001001101011000  =  192.168.147.88
11000000101010001001001101100000  =  192.168.147.96
```

11000000101010001001001101101000 = 192.168.147.104
11000000101010001001001101100000 = 192.168.147.112
11000000101010001001001101111000 = 192.168.147.120
11000000101010001001001110000000 = 192.168.147.128
11000000101010001001001110001000 = 192.168.147.136
11000000101010001001001110010000 = 192.168.147.144
11000000101010001001001110011000 = 192.168.147.152
11000000101010001001001110100000 = 192.168.147.160
11000000101010001001001110101000 = 192.168.147.168
11000000101010001001001110110000 = 192.168.147.176
11000000101010001001001110111000 = 192.168.147.184
11000000101010001001001111000000 = 192.168.147.192
11000000101010001001001111001000 = 192.168.147.200
11000000101010001001001111010000 = 192.168.147.208
11000000101010001001001111011000 = 192.168.147.216
11000000101010001001001111100000 = 192.168.147.224
11000000101010001001001111101000 = 192.168.147.232
11000000101010001001001111110000 = 192.168.147.240

Next, the broadcast addresses of each subnet are derived by setting the host bits of each subnet to all ones.

The subnet broadcast addresses are

11000000101010001001001100001111 = 192.168.147.15
11000000101010001001001100010111 = 192.168.147.23
11000000101010001001001100011111 = 192.168.147.31
11000000101010001001001100100111 = 192.168.147.39
11000000101010001001001100101111 = 192.168.147.47
11000000101010001001001100110111 = 192.168.147.55
11000000101010001001001100111111 = 192.168.147.63
11000000101010001001001101000111 = 192.168.147.71
11000000101010001001001101001111 = 192.168.147.79
11000000101010001001001101010111 = 192.168.147.87
11000000101010001001001101011111 = 192.168.147.95
11000000101010001001001101100111 = 192.168.147.103
11000000101010001001001101101111 = 192.168.147.111
11000000101010001001001101100111 = 192.168.147.119
11000000101010001001001101111111 = 192.168.147.127
11000000101010001001001110000111 = 192.168.147.135
11000000101010001001001110001111 = 192.168.147.143
11000000101010001001001110010111 = 192.168.147.151
11000000101010001001001110011111 = 192.168.147.159
11000000101010001001001110100111 = 192.168.147.167
11000000101010001001001110101111 = 192.168.147.175
11000000101010001001001110110111 = 192.168.147.183
11000000101010001001001110111111 = 192.168.147.191
11000000101010001001001111000111 = 192.168.147.199
11000000101010001001001111001111 = 192.168.147.207
11000000101010001001001111010111 = 192.168.147.215
11000000101010001001001111011111 = 192.168.147.223
11000000101010001001001111100111 = 192.168.147.231
11000000101010001001001111101111 = 192.168.147.239
11000000101010001001001111110111 = 192.168.147.247

Finally, the host addresses for each subnet are derived. They will be all addresses between the subnet addresses and the subnet broadcast addresses.

| Subnet | Broadcast | Host Addresses |
|---|---|---|
| 192.168.147.8 | 192.168.147.15 | 192.168.147.9192.168.147.14 |
| 192.168.147.16 | 192.168.147.23 | 192.168.147.17192.168.147.22 |
| 192.168.147.24 | 192.168.147.31 | 192.168.147.25192.168.147.30 |
| 192.168.147.32 | 192.168.147.39 | 192.168.147.33192.168.147.38 |
| 192.168.147.40 | 192.168.147.47 | 192.168.147.41192.168.147.46 |
| 192.168.147.48 | 192.168.147.55 | 192.168.147.49192.168.147.54 |
| 192.168.147.56 | 192.168.147.63 | 192.168.147.57192.168.147.62 |
| 192.168.147.64 | 192.168.147.71 | 192.168.147.65192.168.147.70 |
| 192.168.147.72 | 192.168.147.79 | 192.168.147.73192.168.147.78 |
| 192.168.147.80 | 192.168.147.87 | 192.168.147.81192.168.147.86 |
| 192.168.147.88 | 192.168.147.95 | 192.168.147.89192.168.147.94 |
| 192.168.147.96 | 192.168.147.103 | 192.168.147.97192.168.147.102 |
| 192.168.147.104 | 192.168.147.111 | 192.168.147.105192.168.147.110 |
| 192.168.147.112 | 192.168.147.119 | 192.168.147.113192.168.147.118 |
| 192.168.147.120 | 192.168.147.127 | 192.168.147.121192.168.147.126 |
| 192.168.147.128 | 192.168.147.135 | 192.168.147.129192.168.147.134 |
| 192.168.147.136 | 192.168.147.143 | 192.168.147.137192.168.147.142 |
| 192.168.147.144 | 192.168.147.151 | 192.168.147.145192.168.147.150 |
| 192.168.147.152 | 192.168.147.159 | 192.168.147.153192.168.147.158 |
| 192.168.147.160 | 192.168.147.167 | 192.168.147.161192.168.147.166 |
| 192.168.147.168 | 192.168.147.175 | 192.168.147.169192.168.147.174 |

6 A class B address with a 20-bit mask will have 14 subnets and 4,094 host addresses on each subnet.

The subnets are

```
11111111111111111111000000000000  =  255.255.240.0   (mask)
10101100001000000001000000000000  =  172.16.16.0
10101100001000000010000000000000  =  172.16.32.0
10101100001000000011000000000000  =  172.16.48.0
10101100001000000100000000000000  =  172.16.64.0
10101100001000000101000000000000  =  172.16.80.0
10101100001000000110000000000000  =  172.16.96.0
10101100001000000111000000000000  =  172.16.112.0
10101100001000001000000000000000  =  172.16.128.0
10101100001000001001000000000000  =  172.16.144.0
10101100001000001010000000000000  =  172.16.160.0
10101100001000001011000000000000  =  172.16.176.0
10101100001000001100000000000000  =  172.16.192.0
10101100001000001101000000000000  =  172.16.208.0
10101100001000001110000000000000  =  172.16.224.0
```

The subnet broadcast addresses are

```
10101100000100000001111111111111  =  172.16.31.255
10101100000100000010111111111111  =  172.16.47.255
10101100000100000011111111111111  =  172.16.63.255
10101100000100000100111111111111  =  172.16.79.255
10101100000100000101111111111111  =  172.16.95.255
10101100000100000110111111111111  =  172.16.111.255
10101100000100000111111111111111  =  172.16.127.255
10101100000100001000111111111111  =  172.16.143.255
10101100000100001001111111111111  =  172.16.159.255
10101100000100001010111111111111  =  172.16.175.255
10101100000100001011111111111111  =  172.16.191.255
10101100000100001100111111111111  =  172.16.207.255
10101100000100001101111111111111  =  172.16.223.255
10101100000100001110111111111111  =  172.16.239.255
```

Using the derived subnet and broadcast addresses, the host addresses are as follows.

| Subnet | Broadcast | Host Addresses |
|---|---|---|
| 172.16.16.0 | 172.16.31.255 | 172.16.16.1172.16.31.254 |
| 172.16.32.0 | 172.16.47.255 | 172.16.32.1172.16.47.254 |
| 172.16.48.0 | 172.16.63.255 | 172.16.48.1172.16.63.254 |
| 172.16.64.0 | 172.16.79.255 | 172.16.64.1172.16.79.254 |
| 172.16.80.0 | 172.16.95.255 | 172.16.80.1172.16.95.254 |
| 172.16.96.0 | 172.16.111.255 | 172.16.96.1172.16.111.254 |
| 172.16.112.0 | 172.16.127.255 | 172.16.112.1172.16.127.254 |
| 172.16.128.0 | 172.16.143.255 | 172.16.128.1172.16.143.254 |
| 172.16.144.0 | 172.16.159.255 | 172.16.144.1172.16.159.254 |
| 172.16.160.0 | 172.16.175.255 | 172.16.160.1172.16.175.254 |
| 172.16.176.0 | 172.16.191.255 | 172.16.176.1172.16.191.254 |
| 172.16.192.0 | 172.16.207.255 | 172.16.192.1172.16.207.254 |
| 172.16.208.0 | 172.16.223.255 | 172.16.208.1172.16.223.254 |
| 172.16.224.0 | 172.16.239.255 | 172.224.16.1172.16.239.254 |

PREV NEXT

# Chapter 3

1 First determine the subnet addresses of each link and then write the static routes. Remember that the routers will already have entries in their route tables for any directly connected subnet. The static routes are as follows:

RTA:

```
ip route 192.168.2.64 255.255.255.224 192.168.2.131
ip route 192.168.2.160 255.255.255.224 192.168.2.131
ip route 192.168.1.128 255.255.255.240 192.168.2.131
ip route 192.168.1.16 255.255.255.240 192.168.2.131
ip route 192.168.2.32 255.255.255.224 192.168.2.131
ip route 192.168.1.160 255.255.255.240 192.168.2.131
ip route 10.1.1.0 255.255.255.0 192.168.2.131
ip route 10.1.3.0 255.255.255.0 192.168.2.131
ip route 10.1.2.0 255.255.255.0 192.168.2.131
```

RTB:

```
ip route 10.1.4.0 255.255.255.0 192.168.2.132
ip route 192.168.1.128 255.255.255.240 192.168.2.174
ip route 192.168.1.16 255.255.255.240 192.168.2.174
ip route 192.168.2.32 255.255.255.224 192.168.2.174
ip route 192.168.1.160 255.255.255.240 192.168.2.174
ip route 10.1.1.0 255.255.255.0 192.168.2.174
ip route 10.1.3.0 255.255.255.0 192.168.2.174
ip route 10.1.2.0 255.255.255.0 192.168.2.174
```

RTC:

```
ip route 10.1.4.0 255.255.255.0 192.168.2.185
ip route 192.168.2.128 255.255.255.224 192.168.2.185
ip route 192.168.2.64 255.255.255.224 192.168.2.185
ip route 192.168.2.32 255.255.255.224 192.168.1.20
ip route 10.1.1.0 255.255.255.0 192.168.1.173
ip route 10.1.3.0 255.255.255.0 192.168.1.173
ip route 10.1.2.0 255.255.255.0 192.168.1.173
```

RTD:

```
ip route 10.1.4.0 255.255.255.0 192.168.1.29
ip route 192.168.2.128 255.255.255.224 192.168.1.29
ip route 192.168.2.64 255.255.255.224 192.168.1.29
ip route 192.168.2.160 255.255.255.224 192.168.1.29
ip route 192.168.1.128 255.255.255.240 192.168.1.29
ip route 192.168.1.160 255.255.255.240 192.168.1.29
ip route 10.1.1.0 255.255.255.0 192.168.1.29
ip route 10.1.3.0 255.255.255.0 192.168.1.29
ip route 10.1.2.0 255.255.255.0 192.168.1.29
```

RTE:

```
ip route 10.1.4.0 255.255.255.0 192.168.1.163
ip route 192.168.2.128 255.255.255.224 192.168.1.163
ip route 192.168.2.64 255.255.255.224 192.168.1.163
ip route 192.168.2.160 255.255.255.224 192.168.1.163
ip route 192.168.1.128 255.255.255.240 192.168.1.163
ip route 192.168.1.16 255.255.255.240 192.168.1.163
ip route 192.168.2.32 255.255.255.224 192.168.1.163
ip route 10.1.2.0 255.255.255.0 10.1.3.2
```

RTF:

```
ip route 10.1.4.0 255.255.255.0 10.1.3.1
ip route 192.168.2.128 255.255.255.224 10.1.3.1
ip route 192.168.2.64 255.255.255.224 10.1.3.1
ip route 192.168.2.160 255.255.255.224 10.1.3.1
ip route 192.168.1.128 255.255.255.240 10.1.3.1
ip route 192.168.1.16 255.255.255.240 10.1.3.1
ip route 192.168.2.32 255.255.255.224 10.1.3.1
ip route 192.168.1.160 255.255.255.240 10.1.3.1
ip route 10.1.1.0 255.255.255.0 10.1.3.1
```

2 The static routes are as follows:

RTA:

```
ip route 192.168.0.0 255.255.0.0 192.168.2.131
ip route 10.1.0.0 255.255.0.0 192.168.2.131
```

RTB:

```
ip route 10.1.4.0 255.255.255.0 192.168.2.132
ip route 192.168.0.0 255.255.0.0 192.168.2.174
ip route 10.1.0.0 255.255.0.0 192.168.2.174
```

RTC:

```
ip route 10.1.4.0 255.255.255.0 192.168.2.185
ip route 192.168.2.32 255.255.255.224 192.168.1.20
ip route 10.1.0.0 255.255.0.0 192.168.1.173
ip route 192.168.2.64 255.255.255.224 192.168.2.185
ip route 192.168.2.128 255.255.255.224 192.168.2.185
```

RTD:

```
ip route 10.1.0.0 255.255.0.0 192.168.1.29
ip route 192.168.0.0 255.255.05.0 192.168.1.29
```

RTE

```
ip route 10.1.4.0 255.255.255.0 192.168.1.163
ip route 192.168.0.0 255.255.0.0 192.168.1.163
ip route 10.1.2.0 255.255.255.0 10.1.3.2
```

RTF:

```
ip route 10.1.0.0 255.255.0.0 10.1.3.1
ip route 192.168.0.0 255.255.0.0 10.1.3.1
```

3 The static routes are as follows:

RTA:

```
ip route 172.16.7.0 255.255.255.0 172.16.2.2
ip route 172.16.7.0 255.255.255.0 172.16.4.2 50
ip route 172.16.6.0 255.255.255.0 172.16.2.2
ip route 172.16.6.0 255.255.255.0 172.16.4.2 50
ip route 172.16.8.0 255.255.255.0 172.16.4.2
ip route 172.16.8.0 255.255.255.0 172.16.2.2 50
ip route 172.16.5.0 255.255.255.0 172.16.4.2
ip route 172.16.5.0 255.255.255.0 172.16.2.2 50
ip route 172.16.9.0 255.255.255.0 172.16.2.2
ip route 172.16.9.0 255.255.255.0 172.16.4.2
```

RTB:

```
ip route 172.16.1.0 255.255.255.0 172.16.2.1
ip route 172.16.1.0 255.255.255.0 172.16.6.1 50
ip route 172.16.4.0 255.255.255.0 172.16.2.1
ip route 172.16.4.0 255.255.255.0 172.16.6.1 50
ip route 172.16.9.0 255.255.255.0 172.16.6.1
ip route 172.16.9.0 255.255.255.0 172.16.2.1 50
ip route 172.16.5.0 255.255.255.0 172.16.6.1
ip route 172.16.5.0 255.255.255.0 172.16.2.1 50
ip route 172.16.8.0 255.255.255.0 172.16.6.1
ip route 172.16.8.0 255.255.255.0 172.16.2.1
```

RTC:

```
ip route 172.16.1.0 255.255.255.0 172.16.6.2
ip route 172.16.1.0 255.255.255.0 172.16.5.1
ip route 172.16.4.0 255.255.255.0 172.16.5.1
ip route 172.16.4.0 255.255.255.0 172.16.6.2 50
ip route 172.16.2.0 255.255.255.0 172.16.6.2
ip route 172.16.2.0 255.255.255.0 172.16.5.1 50
ip route 172.16.7.0 255.255.255.0 172.16.6.2
ip route 172.16.7.0 255.255.255.0 172.16.5.1 50
ip route 172.16.8.0 255.255.255.0 172.16.5.1
ip route 172.16.8.0 255.255.255.0 172.16.6.2 50
```

RTD:

```
ip route 172.16.1.0 255.255.255.0 172.16.4.1
ip route 172.16.1.0 255.255.255.0 172.16.5.2 50
ip route 172.16.2.0 255.255.255.0 172.16.4.1
ip route 172.16.2.0 255.255.255.0 172.16.5.2 50
ip route 172.16.9.0 255.255.255.0 172.16.5.2
ip route 172.16.9.0 255.255.255.0 172.16.4.1 50
ip route 172.16.6.0 255.255.255.0 172.16.5.2
ip route 172.16.6.0 255.255.255.0 172.16.4.1 50
ip route 172.16.7.0 255.255.255.0 172.16.5.2
```

```
ip route 172.16.7.0 255.255.255.0 172.16.4.1
```

PREV NEXT

# Chapter 5

1 In addition to the RIP configurations shown here, a subnet of 192.168.5.0 must be configured between RTE and RTF, using secondary addresses. Otherwise, subnets 192.168.5.192/27 and 192.168.5.96/27 are discontiguous. The RIP configurations are

RTA:

```
router rip
  network 192.168.2.0
```

RTB:

```
router rip
  network 192.168.2.0
```

RTC:

```
router rip
  network 192.168.2.0
  network 192.168.3.0
```

RTD:

```
interface ethernet 0
  ip address 192.168.4.3 255.255.255.0
  ip address 192.168.5.3 255.255.255.224 secondary
router rip
  network 192.168.3.0
  network 192.168.4.0
```

RTE:

```
interface ethernet 0
  ip address 192.168.4.1 255.255.255.0
  ip address 192.168.5.1 255.255.255.224 secondary
router rip
  network 192.168.4.0
  network 192.168.5.0
```

RTF:

```
interface ethernet 0
  ip address 192.168.4.2 255.255.255.0
  ip address 192.168.5.2 255.255.255.224 secondary
router rip
  network 192.168.4.0
  network 192.168.6.0
```

2 To unicast RIP updates between RTC and RTD, the configurations are

RTC:

```
router rip
  network 192.168.2.0
  neighbor 192.168.3.2
```

RTD:

```
router rip
  network 192.168.3.0
  neighbor 192.168.3.1
```

3 The update time applies to the entire RIP process. If the update time is changed for the serial link, it will also be changed for the router's other links. That, in turn, means that the timers must be changed on the neighboring routers, which means those neighbors' neighbors must be changed, and so on. The cascade effect of changing the update timer on a single router means that the timers must be changed on every router in the RIP domain. The command to increase the RIP update period to two minutes, configured on every router, is

```
timers basic 120 360 360 480
```

Because the update timer is changed, the invalid, holddown, and flush timers must also be changed. Setting the invalid and holddown timers to six times the update period, as the default timers are, would make the conversion time of the network extremely high. Therefore, the invalid and holddown timers are set to three times the update period. The flush timer must be longer than the holddown timer, so it is set to 60 seconds longer.

4 Network 192.168.4.0 is two hops from RTA, so adding 14 to the metric will give the route an unreachable metric of 16. Remember that in Configuration Exercise 1 a subnet of 192.168.5.0 had to be configured on the same link as 192.168.4.0 using secondary addresses, so that the subnets of 192.168.5.0 are contiguous. Therefore, 192.168.5.0 is also two hops from RTB. Assuming the interfaces of RTA and RTB connected to RTC are E0 on both routers, the configurations are

RTA:

```
router rip
offset-list 1 in 14 Ethernet0
network 192.168.2.0
!
access-list 1 permit 192.168.4.0 0.0.0.0
```

RTB:

```
router rip
offset-list 1 in 14 Ethernet0
network 192.168.2.0
!
access-list 1 permit 192.168.5.0 0.0.0.0
```

5 RTB, with its longer mask, can interpret all subnets correctly. The problem is at RTA.

With a 27-bit mask, RTA interprets RTB's subnets 192.168.20.40/29 and 192.168.20.49/29 as 192.168.20.32/27the same subnet as its directly connected link. Therefore, RTA does not have a correct "view" of the network.

However, packets can still be routed if Proxy ARP is enabled. For example, suppose RTA has a packet to forward with a destination address of 192.168.20.50. RTA incorrectly interprets this address as a member of its subnet 192.168.20.32/27, and ARPs for the MAC identifier of 192.168.20.50 on that subnet. RTB hears the ARP; it correctly interprets 192.168.20.50 as being a member of its subnet 192.168.20.48/29 and responds with the MAC identifier of its interface on 192.168.20.32/29. RTA then forwards the packet to RTB, and RTB forwards the packet to the correct destination. If Proxy ARP is disabled, packets will not be delivered correctly from RTA to RTB.

.

PREV NEXT

# Chapter 6

1. The statement neighbor 172.25.150.206 can be added to Taos's RIP configuration, causing Taos to unicast RIP updates to that address. This approach works only if Pojoaque's RIPv1 process complies with the rule that the unused fields of RIP messages with version numbers higher than 1 will be ignored and the rest of the packet processed.

2. Begin by calculating the subnets with the highest number of hosts. From the unused subnet bits, calculate the subnet(s) with the next-highest number of hosts, and so on. Remember that as a group of bits are used for a subnet, no subsequent subnet can begin with that same bit combination. For example, if the first subnet begins with 00, all subsequent subnets must begin with 01, 10, or 11. If the second subnet begins with 010, no subsequent subnet can begin with 010.

   One solution (with the subnet bits shaded) is

| | |
|---|---|
| 00000000 | 192.168.100.0/26 (62 hosts) |
| 01000000 | 192.168.100.64/27 (30 hosts) |
| 01100000 | 192.168.100.96/28 (14 hosts) |
| 01110000 | 192.168.100.112/28 (14 hosts) |
| 10000000 | 192.168.100.128/28 (14 hosts) |
| 10010000 | 192.168.100.144/28 (14 hosts) |
| 10100000 | 192.168.100.160/28 (14 hosts) |
| 10110000 | 192.168.100.176/29 (8 hosts) |
| 10111000 | 192.168.100.184/29 (8 hosts) |
| 11000000 | 192.168.100.192/29 (8 hosts) |
| 11001000 | 192.168.100.200/29 (8 hosts) |
| 11010000 | 192.168.100.208/29 (8 hosts) |
| 11011000 | 192.168.100.216/30 (2 hosts) |
| 11011100 | 192.168.100.220/30 (2 hosts) |

| | |
|---|---|
| 11100000 | 192.168.100.224/30 (2 hosts) |
| 11100100 | 192.168.100.228/30 (2 hosts) |
| 11101000 | 192.168.100.232/30 (2 hosts) |
| 11101100 | 192.168.100.236/30 (2 hosts) |
| 11110000 | 192.168.100.240/30 (2 hosts) |
| 11110100 | 192.168.100.244/30 (2 hosts) |
| 11111000 | 192.168.100.248/30 (2 hosts) |
| 11111100 | 192.168.100.252/30 (2 hosts) |

3 RTA, RTB, and RTD have the statement version 2 in their RIP configurations. Additionally, the RIP configurations of RTA and RTB include the statement no auto-summary. The interfaces of RTA and RTB connected to subnet 192.168.2.64/28 have the statements ip rip send version 1 2 and ip rip receive version 1 2. The interface of RTD connected to subnet 192.168.2.128/28 has the statements ip rip send version 1 and ip rip receive version 1.

4 The following solution uses a key chain name of *CCIE* and a key string named *exercise4* on RTB and RTD. Assuming both routers' interfaces are S0, the configuration of both RTB and RTD is

```
key chain CCIE
  key 1
    key-string exercise4
!
interface Serial0
  ip address 192.168.1.15X 255.255.255.252
  ip rip authentication mode md5
  ip rip authentication key-chain CCIE
```

5 The configuration here assumes that the authentication key in Configuration Exercise 4 was enabled on October 31, 2004, at midnight. The second key string used here is *exercise5a*, and the third key string is *exercise5b*:

```
key chain CCIE
  key 1
    key-string exercise4
```

PREV NEXT

# Chapter 7

1 The EIGRP configurations are

RTA:

```
router eigrp 5
network 172.16.0.0
network 172.18.0.0
```

RTB:

```
router eigrp 5
network 172.16.0.0
```

RTC:

```
router eigrp 5
network 172.16.0.0
network 172.17.0.0
```

2 This solution uses a key chain named *CCIE* and key strings named *exercise2a* and *exercise2b*. Assuming today's date is November 30, 2004, and the first key begins being used at 8:30 AM, the configurations of the serial interfaces are

```
key chain CCIE
key 1
key-string exercise2a
accept-lifetime 08:30:00 Dec 2 2004 08:30:00 Jan 1 2005
send-lifetime 08:30:00 Dec 2 2004 08:30:00 Jan 1 2005
key 2
key-string exercise2b
accept-lifetime 08:30:00 Jan 1 2005 infinite
send-lifetime 08:30:00 Jan 1 2005 infinite
!
interface Serial0
ip address 172.16.3.19X 255.255.255.252
ip authentication key-chain eigrp 5 CCIE
ip authentication mode eigrp 5 md5
```

3 The EIGRP configuration of RTD is

```
router eigrp 5
network 172.16.0.0
network 172.17.0.0
no auto-summary
```

Auto-summarization must also be turned off on RTC.

4 The EIGRP configuration of RTF is

```
router eigrp 5
network 172.16.0.0
network 172.18.0.0
no auto-summary
```

The statement ip summary-address eigrp 5 172.18.10.192 255.255.255.224 is added to RTA's interface on subnet 172.18.10.96/27. Also, note that auto-summarization must be turned off on RTA, because of the appearance of the 172.16.0.0 network on RTF.

5 RTA can send a summary address of 172.16.3.128/25 to RTF and a summary address of 172.16.3.0/25 to RTB. All other summarization is performed automatically.

PREV NEXT

# Chapter 8

1 The OSPF configurations are

RTA:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
```

RTB:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.5.0.0 0.0.255.255 area 5
area 5 virtual-link 10.100.100.9
```

RTC:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.10.0.0 0.0.255.255 area 10
network 10.30.0.0 0.0.255.255 area 30
```

RTD:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.20.0.0 0.0.255.255 area 20
```

RTE:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.15.0.0 0.0.255.255 area 15
```

RTF:

```
router ospf 1
network 10.5.0.0 0.0.255.255 area 5
```

RTG:

```
router ospf 1
network 10.10.1.58 0.0.0.0 area 10
```

RTH:

```
router ospf 1
```

```
network 10.20.100.100 0.0.0.0 area 20

network 10.20.1.0 0.0.0.255 area 20
```

RTI:

```
router ospf 1
network 10.5.0.0 0.0.255.255 area 5
network 10.35.0.0 0.0.255.255 area 35
area 5 virtual-link 10.100.100.2
```

RTJ:

```
router ospf 1
network 10.15.0.0 0.0.255.255 area 15
```

RTK through RTN have Frame Relay interfaces. The four interfaces to the Frame Relay network are all on the same subnet, so the OSPF network type must be either broadcast or point-to-multipoint:

RTK:

```
interface Serial0
encapsulation frame-relay
ip address 10.30.254.193 255.255.255.192
ip ospf network point-to-multipoint
!
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
```

RTL:

```
encapsulation frame-relay
ip address 10.30.254.194 255.255.255.192
ip ospf network point-to-multipoint
!
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
```

RTM:

```
encapsulation frame-relay
ip address 10.30.254.195 255.255.255.192
ip ospf network point-to-multipoint
!
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
```

RTN:

```
encapsulation frame-relay
ip address 10.30.254.196 255.255.255.192
ip ospf network point-to-multipoint
!
router ospf 1
```

```
network 10.30.0.0 0.0.255.255 area 30
```

2 The ABR configurations are

RTB:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.5.0.0 0.0.255.255 area 5
area 5 virtual-link 10.100.100.9
area 0 range 10.0.0.0 255.255.0.0
area 5 range 10.5.0.0 255.255.0.0
```

RTC:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.10.0.0 0.0.255.255 area 10
network 10.30.0.0 0.0.255.255 area 30
area 0 range 10.0.0.0 255.255.0.0
area 10 range 10.10.0.0 255.255.0.0
area 30 range 10.30.0.0 255.255.0.0
```

RTD:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.20.0.0 0.0.255.255 area 20
area 0 range 10.0.0.0 255.255.0.0
area 20 range 10.20.0.0 255.255.0.0
```

RTE:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.15.0.0 0.0.255.255 area 15
area 0 range 10.0.0.0 255.255.0.0
area 15 range 10.15.0.0 255.255.0.0
```

RTI:

```
router ospf 1
network 10.5.0.0 0.0.255.255 area 5
network 10.35.0.0 0.0.255.255 area 35
area 5 virtual-link 10.100.100.2
area 0 range 10.0.0.0 255.255.0.0
area 5 range 10.5.0.0 255.255.0.0
area 35 range 10.35.0.0 255.255.0.0
```

3 The configurations are

RTE:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
```

```
network 10.15.0.0 0.0.255.255 area 15
area 15 stub
```

RTJ:

```
router ospf 1
network 10.15.0.0 0.0.255.255 area 15
area 15 stub
```

4 The configurations are

RTC:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.10.0.0 0.0.255.255 area 10
network 10.30.0.0 0.0.255.255 area 30
area 0 range 10.0.0.0 255.255.0.0
area 10 range 10.10.0.0 255.255.0.0
area 30 stub no-summary
area 30 range 10.30.0.0 255.255.0.0
```

RTK:

```
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
area 30 stub
```

RTL:

```
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
area 30 stub
```

RTM:

```
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
area 30 stub
```

RTN:

```
router ospf 1
network 10.30.0.0 0.0.255.255 area 30
area 30 stub
```

5 The configurations are

RTD:

```
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.20.0.0 0.0.255.255 area 20
area 20 nssa
```

RTH:

```
router ospf 1
network 10.20.100.100 0.0.0.0 area 20
area 20 nssa
```

6 Only one end of a point-to-point circuit needs to be configured as a demand circuit. In this solution, RTC is chosen:

```
interface Serial0
ip address 10.30.255.249 255.255.255.252
ip ospf demand-circuit
!
router ospf 1
network 10.0.0.0 0.0.255.255 area 0
network 10.10.0.0 0.0.255.255 area 10
network 10.30.0.0 0.0.255.255 area 30
area 0 range 10.0.0.0 255.255.0.0
area 10 range 10.10.0.0 255.255.0.0
area 30 range 10.30.0.0 255.255.0.0
```

PREV NEXT

# Chapter 9

1. No additional OSPFv3 command is needed to include a second IPv6 prefix that has been configured on an interface in the OSPFv3 process. If OSPFv3 is configured on an interface, and a second IPv6 address is added to the interface, the second address is automatically included in the OSPFv3 process.

2. Yes, two routers can become adjacent if they are configured with different prefixes. No prefix information is included in Hellos, the source address of the Hello packet is the Link-local address. Other IPv6 addresses configured on the interfaces have no bearing on whether two routers become adjacent or not.

3. The basic router configurations are

RtrA:

```
ipv6 unicast-routing
int e 0/0
 ipv6 address 2001:db8:0:4::3/64
 ipv6 ospf 1 area 1
int e 0/1
 ipv6 address 2001:db8:0:5::1/64
 ipv6 ospf 1 area 1
```

RtrB:

```
ipv6 unicast-routing
int e 0/0
 ipv6 address 2001:db8:0:4::2/64
 ipv6 ospf 1 area 1

int e 0/1
 ipv6 address 2001:db8:0:6::1/64
 ipv6 ospf 1 area 1
```

RtrC:

```
ipv6 unicast-routing
int e 0/0
 ipv6 address 2001:db8:0:4::1/64
 ipv6 ospf 1 area 1

int serial 0/0
 ipv6 address 2001:db8:0:8::1/64
 ipv6 ospf 1 area 0
```

RtrD:

```
ipv6 unicast-routing
int s 0/0
 ipv6 address 2001:db8:0:8::2/64
 ipv6 ospf 1 area 0
```

4 To change area 1 to a totally stubby area and summarize into area 0, new router configurations are

RtrA:

```
ipv6 unicast-routing
int e 0/0
 ipv6 address 2001:db8:0:5::3/64
 ipv6 ospf 1 area 1
ipv6 router ospf 1
 area 1 stub
```

RtrB:

```
ipv6 unicast-routing
int e 0/0
 ipv6 address 2001:db8:0:4::2/64
 ipv6 ospf 1 area 1
ipv6 router ospf 1
 area 1 stub
```

RtrC:

```
ipv6 unicast-routing
int e 0/0
 ipv6 address 2001:db8:0:4::1/64
 ipv6 ospf 1 area 1
int serial 0/0
 ipv6 address 2001:db8:0:8::1/64
 ipv6 ospf 1 area 0
ipv6 router ospf 1
 area 1 stub no-summary
 area 1 range 2001:db8:0:10::/61
```

PREV NEXT

# Chapter 10

1 This solution uses a System ID of 0000.1234.abcX, where X is a number that makes the router unique within the IS-IS domain. The System ID is six octets, which is required by Cisco IOS. The minimum NET length is eight octets, one of which is the SEL, so the area address is a one-octet number corresponding to the area number shown in Table 10-6. The configurations are

RTA:

```
interface Ethernet0
ip address 192.168.1.17 255.255.255.240
ip router isis
!
interface Ethernet1
ip address 192.168.1.50 255.255.255.240
ip router isis
!
router isis
net 00.0000.1234.abc1.00
is-type level-1
```

RTB:

```
interface Ethernet0
ip address 192.168.1.33 255.255.255.240
ip router isis
!
interface Ethernet1
ip address 192.168.1.51 255.255.255.240
ip router isis
!
router isis
net 00.0000.1234.abc2.00
is-type level-1
```

RTC:

```
interface Ethernet0
ip address 192.168.1.49 255.255.255.240
ip router isis
!
interface Serial0
ip address 192.168.1.133 255.255.255.252
ip router isis
!
router isis
net 00.0000.1234.abc3.00
```

RTD:

```
interface Serial0
```

```
ip address 192.168.1.134 255.255.255.252
ip router isis
!
interface Serial1
ip address 192.168.1.137 255.255.255.252
ip router isis
!
router isis
net 02.0000.1234.abc4.00
is-type level-2-only
```

RTE:

```
interface Serial0
ip address 192.168.1.142 255.255.255.252
ip router isis
!
interface Serial1
ip address 192.168.1.145 255.255.255.252
ip router isis
!
interface Serial2
ip address 192.168.1.138 255.255.255.252
ip router isis
!
router isis
net 02.0000.1234.abc5.00
is-type level-2-only
```

RTF:

```
interface Serial0
ip address 192.168.1.141 255.255.255.252
ip router isis
!
interface Serial1
ip address 192.168.1.158 255.255.255.252
ip router isis
!
router isis
net 02.0000.1234.abc6.00
is-type level-2-only
```

RTG:

```
interface Ethernet0
ip address 192.168.1.111 255.255.255.224
ip router isis
!
interface Serial0
ip address 192.168.1.157 255.255.255.252
ip router isis
!
router isis
net 01.0000.1234.abc7.00
```

RTH:

```
interface Ethernet0
ip address 192.168.1.73 255.255.255.224
ip router isis
!
interface Ethernet1
ip address 192.168.1.97 255.255.255.224
ip router isis
!
router isis
net 01.0000.1234.abc8.00
is-type level-1
```

RTI:

```
interface Ethernet0
ip address 192.168.1.225 255.255.255.248
ip router isis
!
interface Ethernet1
ip address 192.168.1.221 255.255.255.248
ip router isis
!
interface Serial0
ip address 192.168.1.249 255.255.255.252
ip router isis
!
interface Serial1
ip address 192.168.1.146 255.255.255.252
ip router isis
!
router isis
net 03.0000.1234.abc9.00
```

RTJ:

```
interface Ethernet0
ip address 192.168.1.201 255.255.255.248
ip router isis
!
interface Ethernet1
ip address 192.168.1.217 255.255.255.248
ip router isis
!
router isis
net 03.0000.1234.abca.00
is-type level-1
```

RTK:

```
interface Ethernet0
ip address 192.168.1.209 255.255.255.248
ip router isis
!
interface Serial0
ip address 192.168.1.250 255.255.255.252
ip router isis
!
```

```
router isis
net 03.0000.1234.abcb.00
is-type level-1
```

2 The configurations are

RTA:

```
ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.17 255.255.255.240
ip router isis
ipv6 address 2001:db8:0:10::1/64
ipv6 router isis
!
interface Ethernet1
ip address 192.168.1.50 255.255.255.240
ip router isis
ipv6 address 2001:db8:0:30::2/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc1.00
is-type level-1
metric-style wide
address-family ipv6
multi-topology
```

RTB:

```
ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.33 255.255.255.240
ip router isis
ipv6 address 2001:db8:0:20::1/64
ipv6 router isis
!
interface Ethernet1
ip address 192.168.1.51 255.255.255.240
ip router isis
ipv6 address 2001:db8:0:30::3/64
ipv6 router isis

!
router isis
net 00.0000.1234.abc2.00
is-type level-1
metric-style wide
address-family ipv6
multi-topology
```

RTC:

```
ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.49 255.255.255.240
ip router isis
ipv6 address 2001:db8:0:30::1/64
```

```
ipv6 router isis
!
interface Serial0
ip address 192.168.1.133 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:84::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc3.00
metric-style wide
address-family ipv6
multi-topology
```

RTD:

```
ipv6 unicast-routing
interface Serial0
ip address 192.168.1.134 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:84::2/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.137 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:88::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc4.00
is-type level-2-only
metric-style wide
address-family ipv6
multi-topology
```

RTE:

```
ipv6 unicast-routing
interface Serial0
ip address 192.168.1.142 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:8c::2/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.145 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:90::1/64
ipv6 router isis
!
interface Serial2
ip address 192.168.1.138 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:88::2/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc5.00
```

```
is-type level-2-only
metric-style wide
address-family ipv6
multi-topology
```

RTF:

```
ipv6 unicast-routing
interface Serial0
ip address 192.168.1.141 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:8c::1/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.158 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:9c::2/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc6.00
is-type level-2-only
metric-style wide
address-family ipv6
multi-topology
```

RTG:

```
ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.111 255.255.255.224
ip router isis
ip router clns
ipv6 address 2001:db8:0:60::f/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.157 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:9c::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc7.00
metric-style wide
address-family ipv6
multi-topology
```

RTH:

```
ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.73 255.255.255.224
ip router isis
ipv6 address 2001:db8:0:40::9/64
ipv6 router isis
!
```

```
interface Ethernet1
ip address 192.168.1.97 255.255.255.224
ip router isis
ip router clns
ipv6 address 2001:db8:0:60::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc8.00
is-type level-1
metric-style wide
address-family ipv6
multi-topology
```

RTI:

```
ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.225 255.255.255.248
ip router isis
ipv6 address 2001:db8:0:e0::1/64
ipv6 router isis
!
interface Ethernet1
ip address 192.168.1.221 255.255.255.248
ip router isis
ipv6 address 2001:db8:0:d8::5/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.249 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:f8::1/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.146 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:90::2/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc9.00
metric-style wide
address-family ipv6
multi-topology
```

RTJ:

```
ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.201 255.255.255.248
ip router isis
ipv6 address 2001:db8:0:c8::1/64
ipv6 router isis
!
interface Ethernet1
ip address 192.168.1.217 255.255.255.248
ip router isis
```

```
ipv6 address 2001:db8:0:d8::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abca.00
is-type level-1
metric-style wide
address-family ipv6
multi-topology
```

RTK:

```
ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.209 255.255.255.248
ip router isis
ipv6 address 2001:db8:0:d0::1/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.250 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:f8::2/64
ipv6 router isis
!
router isis
net 00.0000.1234.abcb.00
is-type level-1
metric-style wide
address-family ipv6
multi-topology
```

3 The configurations are

RTD:

```
Ipv6 unicast-routing
Key chain Fair
 Key 1
 Key-string Eiffel

interface Serial0
ip address 192.168.1.134 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:84::2/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.137 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:88::1/64
ipv6 router isis
authentication key-chain Fair level-2
authentication mode md5 level-2

!
router isis
net 00.0000.1234.abc4.00
is-type level-2-only
```

```
metric-style wide
address-family ipv6
multi-topology
```

RTE:

```
Ipv6 unicast-routing
Key chain Fair
 Key 1
 Key-string Eiffel
Key chain Paris
Key 1
 Key-string Tower
interface Serial0
ip address 192.168.1.142 255.255.255.252
ip router isis
ipv6 address 2001:db7:0:8c::2/64
ipv6 router isis
authentication key-chain Paris level-2
authentication mode md5 level-2
!
interface Serial1
ip address 192.168.1.145 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:90::1/64
ipv6 router isis
!
interface Serial2
ip address 192.168.1.138 255.255.255.252
ip router isis ipv6 address 2001:db8:0:88::2/64
ipv6 router isis
authentication key-chain Fair level-2
authentication mode md5 level-2
!
router isis
net 00.0000.1234.abc5.00
is-type level-2-only
metric-style wide
address-family ipv6
multi-topology
```

RTF:

```
Ipv6 unicast-routing
Key chain Paris
Key 1
 Key-string Tower
interface Serial0
ip address 192.168.1.141 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:8c::1/64
ipv6 router isis
authentication key-chain Paris level-2
authentication mode md5 level-2
!
interface Serial1
ip address 192.168.1.158 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:9c::2/64
```

```
ipv6 router isis
!
router isis
net 00.0000.1234.abc6.00
is-type level-2-only
metric-style wide
address-family ipv6
multi-topology
```

4 The configurations are

RTG:

```
Ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.111 255.255.255.224
ip router isis
ipv6 address 2001:db8:0:96::f/64
ipv6 router isis
ip router clns
!
interface Serial0
ip address 192.168.1.157 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:9c::1/64
ipv6 router isis!
router isis
net 00.0000.1234.abc7.00
area-password Scotland
metric-style wide
address-family ipv6
multi-topology
```

RTH:

```
Ipv6 unicast-routing
interface Ethernet0
ip address 192.168.1.73 255.255.255.224
ip router isis
ipv6 address 2001:db8:0:40::9/64
ipv6 router isis
!
interface Ethernet1
ip address 192.168.1.97 255.255.255.224
ip router isis
ip router clns
ipv6 address 2001:db8:0:60::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc8.00
is-type level-1
area-password Scotland
metric-style wide
address-family ipv6
multi-topology
```

5 The configurations are

RTC:

```
Ipv6 unicast-routing
Key chain Austria
Key 1
 Key-string Vienna

interface Ethernet0
ip address 192.168.1.49 255.255.255.240
ip router isis
ipv6 address 2001:db8:0:30::1/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.133 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:84::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc3.00

authentication key-chain Austria level-2
authentication mode text
metric-style wide
address-family ipv6
multi-topology
```

RTD:

```
Ipv6 unicast-routing
Key chain Fair
 Key 1
 Key-string Eiffel

Key chain Austria
Key 1
 Key-string Vienna

interface Serial0
ip address 192.168.1.134 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:84::2/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.137 255.255.255.252
ip router isis
authentication key-chain Fair level-2
authentication mode md5 level-2
ipv6 address 2001:db8:0:88::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc4.00
is-type level-2-only
authentication key-chain Austria level-2
authentication mode text
metric-style wide
```

```
address-family ipv6
multi-topology
```

RTE:

```
Ipv6 unicast-routing
Key chain Fair
 Key 1
 Key-string Eiffel

Key chain Paris
Key 1
 Key-string Tower

Key chain Austria
Key 1
 Key-string Vienna
interface Serial0
ip address 192.168.1.142 255.255.255.252
ip router isis
authentication key-chain Paris level-2
authentication mode md5 level-2
 ipv6 address 2001:db8:0:8c::2
 ipv6 routing isis
!
interface Serial1
ip address 192.168.1.145 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:90::1/64
ipv6 router isis
!
interface Serial2
ip address 192.168.1.138 255.255.255.252
ip router isis
authentication key-chain Fair level-2
authentication mode md5 level-2
ipv6 address 2001:db8:0:88::2/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc5.00
is-type level-2-only
authentication key-chain Austria level-2
authentication mode text
metric-style wide
address-family ipv6
multi-topology
```

RTF:

```
Ipv6 unicast-routing
Key chain Paris
Key 1
 Key-string Tower

Key chain Austria
Key 1
 Key-string Vienna
```

```
interface Serial0
ip address 192.168.1.141 255.255.255.252
ip router isis
authentication key-chain Paris level-2
authentication mode md5 level-2
ipv6 address 2001:db8:0:8c::1/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.158 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:9c::2/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc6.00
is-type level-2-only
authentication key-chain Austria level-2
authentication mode text
metric-style wide
address-family ipv6
multi-topology
```

RTG:

```
Ipv6 unicast-routing
Key chain Austria
Key 1
 Key-string Vienna
interface Ethernet0
ip address 192.168.1.111 255.255.255.224
ip router isis
ip router clns
ipv6 address 2001:db8:0:96::f/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.157 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:9c::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc7.00
area-password Scotland
authentication key-chain Austria level-2
authentication mode text
metric-style wide
address-family ipv6
multi-topology
```

RTI:

```
Ipv6 unicast-routing
Key chain Austria
Key 1
 Key-string Vienna
interface Ethernet0
ip address 192.168.1.225 255.255.255.248
```

```
ip router isis
ipv6 address 2001:db8:0:e0::1/64
ipv6 router isis
!
interface Ethernet1
ip address 192.168.1.221 255.255.255.248
ip router isis
ipv6 address 2001:db8:0:d8::5/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.249 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:f8::1/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.146 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:90::2/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc9.00
authentication key-chain Austria level-2
authentication mode text
metric-style wide
address-family ipv6
multi-topology
```

6 The configurations are

RTC:

```
Ipv6 unicast-routing
Key chain Austria
Key 1
 Key-string Vienna
interface Ethernet0
ip address 192.168.1.49 255.255.255.240
ip router isis
ipv6 address 2001:db8:0:30::1/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.133 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:84::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc3.00
summary-address 192.168.1.0 255.255.255.192
authentication key-chain Austria level-2
authentication mode text
metric-style wide
address-family ipv6
multi-topology
summary-prefix 2001:db8::/58
```

RTG:

```
Ipv6 unicast-routing
Key chain Austria
Key 1
 Key-string Vienna
interface Ethernet0
ip address 192.168.1.111 255.255.255.224
ip router isis
ip router clns
ipv6 address 2001:db8:0:96::f/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.157 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:9c::1/64
ipv6 router isis
!
router isis
net 00.0000.1234.abc7.00
area-password Scotland
authentication key-chain Austria level-2
authentication mode text
metric-style wide
summary-address 192.168.1.64 255.255.255.192
address-family ipv6
multi-topology
summary-prefix 2001:db8:0:40::/58
```

RTI:

```
Ipv6 unicast-routing
Key chain Austria
Key 1
 Key-string Vienna
interface Ethernet0
ip address 192.168.1.225 255.255.255.248
ip router isis
ipv6 address 2001:db8:0:e0::1/64
ipv6 router isis
!
interface Ethernet1
ip address 192.168.1.221 255.255.255.248
ip router isis
ipv6 address 2001:db8:0:d8::5/64
ipv6 router isis
!
interface Serial0
ip address 192.168.1.249 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:f8::1/64
ipv6 router isis
!
interface Serial1
ip address 192.168.1.146 255.255.255.252
ip router isis
ipv6 address 2001:db8:0:90::2/64
ipv6 router isis
```

```
!
router isis
net 00.0000.1234.abc9.00
authentication key-chain Austria level-2
authentication mode text
metric-style wide
summary-address 192.168.1.192 255.255.255.192
address-family ipv6
multi-topology
summary-prefix 2001:db8:0:c0::/58
```

PREV NEXT

# Chapter 11

1 RIP does not advertise subnet 172.16.1.144/28. To remedy this problem, a static route is entered to that subnet but with a 27-bit mask. Because the mask references RTB's E0 interface, it is redistributed into RIP automatically.

```
interface Ethernet0
ip address 172.16.1.146 255.255.255.240
ipv6 address 2001:db8:0:190::2/64
ipv6 ospf 1 area 0
!
interface Ethernet1
ip address 172.16.1.98 255.255.255.224
ipv6 address 2001:db8:0:160::2/64
ipv6 rip ripdomain enable
!
router ospf 1
redistribute rip metric 50 subnets
network 172.16.1.0 0.0.0.255 area 0
!
router rip
redistribute ospf 1 metric 2
passive-interface Ethernet0
network 172.16.0.0
!
ipv6 router rip ripdomain
redistribute ospf 1 metric 2
ipv6 router ospf
redistribute rip ripdomain metric 50

ip classless
ip route 172.16.1.128 255.255.255.224 Ethernet0
```

2 RTB's configuration is

```
ipv6 router rip ripdomain
redistribute ospf 1 metric 2

ipv6 router ospf
redistribute rip ripdomain metric 50
summary-prefix 2001:db8:0:200::/48
```

3 RTB's configuration is

```
interface Ethernet0
ip address 172.16.1.146 255.255.255.240
ip router isis
ipv6 address 2001:db8:0:190::2/64
ipv6 router isis
!
interface Ethernet1
ip address 172.16.1.98 255.255.255.224
```

```
ip summary-address eigrp 1 172.16.1.128 255.255.255.128
ipv6 address 2001:db8:0:160::2/64
ipv6 rip ripdomain enable
!
router eigrp 1
redistribute isis level-1 metric 10000 1000 255 1 1500
passive-interface Ethernet0
network 172.16.0.0
!
router isis
net 00.0000.1234.abc1.00
summary-address 172.16.2.0 255.255.255.0
redistribute eigrp 1 metric 20 metric-type external level-1
address-family ipv6
summary-prefix 2001:db8:0:200::/48
redistribute rip metric 20 metric-type external level-1
!
ipv6 router rip ripdomain
redistribute isis metric 5
```

PREV NEXT

# Chapter 13

1 RTA's configuration is

```
router rip
redistribute eigrp 1 metric 3
passive-interface Ethernet0
passive-interface Ethernet1
network 172.16.0.0
distribute-list 1 in Ethernet3
!
router eigrp 1
redistribute rip metric 10000 1000 255 1 1500
passive-interface Ethernet2
passive-interface Ethernet3
network 172.16.0.0!
access-list 1 deny 172.16.12.0
access-list 1 permit any
```

2 RTA's configuration is

```
router rip
redistribute eigrp 1 metric 3
passive-interface Ethernet0
passive-interface Ethernet1
network 172.16.0.0
distribute-list 2 out Ethernet2
!
router eigrp 1
redistribute rip metric 10000 1000 255 1 1500
passive-interface Ethernet2
passive-interface Ethernet3
network 172.16.0.0
!
access-list 2 deny 172.16.10.0
access-list 2 permit any
```

3 RTA's configuration is

```
router rip
redistribute eigrp 1 metric 3
passive-interface Ethernet0
passive-interface Ethernet1
network 172.16.0.0
distribute-list 3 out eigrp 1
!
router eigrp 1
redistribute rip metric 10000 1000 255 1 1500
passive-interface Ethernet2
passive-interface Ethernet3
network 172.16.0.0
!
```

```
access-list 3 permit 172.16.2.0
access-list 3 permit 172.16.8.0
access-list 3 permit 172.16.9.0
```

4 RTA's configuration is

```
router rip
redistribute eigrp 1 metric 3
passive-interface Ethernet0
passive-interface Ethernet1
network 172.16.0.0
!
router eigrp 1
redistribute rip metric 10000 1000 255 1 1500
passive-interface Ethernet2
passive-interface Ethernet3
network 172.16.0.0
distribute-list 4 out Ethernet0
!
access-list 4 permit 172.16.1.0
access-list 4 permit 172.16.2.0
access-list 4 permit 172.16.3.0
access-list 4 permit 172.16.7.0
access-list 4 permit 172.16.8.0
access-list 4 permit 172.16.9.0
```

5 The distribute list configured under the RIPng process is

```
ipv6 router rip ripname
distribute-list prefix-list listname out ethernet2
!
ipv6 prefix-list listname seq 5 deny 2001:DB8:0:A::/63 le 64
ipv6 prefix-list listname seq 10 permit ::/0 le 128
```

6 EIGRP assigns a higher distance (170) to external routes than to internal routes (90). If any destination within the EIGRP domain is advertised into EIGRP from IS-IS, it will be ignored unless the internal route to the destination fails. Therefore, the EIGRP distances do not have to be manipulated. The distance statements under the IS-IS configurations of RTC and RTD follow:

RTC:

```
distance 115
distance 170 192.168.10.254 0.0.0.0 1
!
access-list 1 permit any
```

RTD:

```
distance 115
distance 170 192.168.10.249 0.0.0.0 1
distance 170 192.168.10.241 0.0.0.0 1
!
access-list 1 permit any
```

7 The distance statement under the IS-IS configuration of RTD is

```
distance 115
distance 255 192.168.10.241 0.0.0.0 1
!
access-list 1 permit any
```

8 The distance statement under RTC's EIGRP configuration is distance eigrp 90 90

PREV NEXT

# Chapter 14

1 RTA's configuration is

```
interface Serial0
ip address 172.16.14.6 255.255.255.252
ip policy route-map Exercise1
!
interface Serial1
ip address 172.16.14.10 255.255.255.252
ip policy route-map Exercise1
!
access-list 1 permit 172.16.1.0 0.0.0.127
access-list 2 permit 172.16.1.128 0.0.0.127
!
route-map Exercise1 permit 10
match ip address 1
set ip next-hop 172.16.14.17
!
route-map Exercise1 permit 20
match ip address 2
set ip next-hop 172.16.14.13
```

2 RTA's configuration is

```
interface Serial0
ip address 172.16.14.6 255.255.255.252
ip policy route-map Exercise2A
!
interface Serial1
ip address 172.16.14.10 255.255.255.252
ip policy route-map Exercise2B
!
access-list 1 permit 172.16.1.64 0.0.0.63
!
route-map Exercise2 permit 10
match ip address 1
set ip next-hop 172.16.14.13
!
route-map Exercise2B permit 10
match ip address 1
set ip next-hop 172.16.14.17
```

3 RTA's configuration is

```
interface Serial2
ip address 172.16.14.18 255.255.255.252
ip access-group 101 in
ip policy route-map Exercise3
!
interface Serial3
ip address 172.16.14.14 255.255.255.252
```

```
ip access-group 101 in
ip policy route-map Exercise3
!
access-list 101 permit udp any 172.168.1.0 0.0.0.255
access-list 101 permit tcp any eq smtp 172.16.1.0 0.0.0.255
access-list 102 permit tcp any eq smtp 172.16.1.0 0.0.0.255
access-list 103 permit udp any 172.168.1.0 0.0.0.255
!
route-map Exercise3 permit 10
match ip address 102
set ip next-hop 172.16.14.9
!
route-map Exercise3 permit 20
match ip address 103
set ip next-hop 172.16.14.5
```

4 The OSPF configuration is

```
router ospf 1
redistribute eigrp 1 route-map Exercise4
network 192.168.1.0 0.0.0.255 area 16
!
access-list 1 permit 10.201.100.0

!
route-map Exercise4 deny 10
match ip address 1
!
route-map Exercise4 permit 20
match route-type internal
set metric 10
set metric-type type-1
!
route-map Exercise4 permit 30
match route-type external
set metric 50
set metric-type type-2
```

5 The EIGRP configuration is

```
router eigrp 1
redistribute ospf 1 route-map Exercise5
network 192.168.100.0
!
access-list 1 permit 192.168.1.0
access-list 1 permit 192.168.2.0
access-list 1 permit 192.168.3.0
!
route-map Exercise5 permit 10
match ip address 1
match route-type internal
set metric 10000 100 255 1 1500
!
route-map Exercise5 permit 30
match ip address 1
match route-type external
set metric 10000 10000 255 1 1500
```

# Appendix F. Solutions to Troubleshooting Exercises

Chapter 1

Chapter 3

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 10

Chapter 11

Chapter 13

Chapter 14

# Chapter 1

1 Subnet: 10.14.64.0

Host Addresses: 10.14.64.110.14.95.254

Broadcast: 10.14.95.255

Subnet: 172.25.0.224

Host Addresses: 172.25.0.225175.25.0.254

Broadcast: 172.25.0.255

Subnet: 172.25.16.0

Host Addresses: 172.25.16.1172.25.16.126

Broadcast: 172.25.16.127

2 192.168.13.175/28 is the broadcast address of subnet 192.168.13.160/28.

# Chapter 3

1. Subnet 192.168.1.64/27 will no longer be reachable from Piglet. Subnets 10.4.6.0/24 and 10.4.7.0/24 are also no longer reachable from Piglet.

2. Mistakes occur in the following:

   - RTA, the second entry, should be ip route 172.20.80.0 255.255.240.0 172.20.20.1
   - RTB, the third entry, should be ip route 172.20.208.0 255.255.240.0 172.20.16.50
   - RTC, the second entry, should be ip route 172.20.208.0 255.255.240.0 172.20.16.50 and fifth entry, should be ip route 172.20.80.0 255.255.240.0 172.20.20.1

3. The mistakes are as follows:

   - RTC: The route to 10.5.8.0/24 points to the wrong next-hop address.
   - RTC: The route to 10.1.1.0/24 should be 10.5.1.0/24. (There is no 10.1.1.0/24 in the network.)
   - RTC: There is no route to 10.5.4.0/24.
   - RTD: The route to 10.4.5.0/24 should be 10.5.4.0/24.

# Chapter 5

1. The new access list causes two hops to be added to every route except 10.33.32.0.

2. RTB interprets all subnets of 172.16.0.0 according to its misconfigured masks. The consequences, as exhibited in each of the four entries in its route table, are

   Entry 1: This entry is correct because 172.16.24.0 can be masked with either 22 bits or 23 bits.

   Entry 2: Subnet 172.16.26.0 is advertised by RTC. Because the 23rd bit of this address is one and RTB is using a 22-bit mask, from RTB's perspective this one appears in the host address space. Therefore, RTB interprets the advertisement of 172.16.26.0 as a host route and marks it with a 32-bit mask in the route table.

   Entry 3: RTB interprets its interface address 172.16.22.5 as being a member of subnet 172.16.20.0/22, instead of 172.16.22.0/23. When RTB receives RTA's advertisement of subnet 172.16.20.0/23, RTB, thinking it has a directly connected link to that subnet, ignores the advertisement. Notice that subnet 172.16.22.0 is not in the route tables of RTA and RTC for the same reason: RTB is advertising it as 172.16.20.0.

   Entry 4: RTB interprets its interface address 172.16.18.4 as being a member of subnet 172.16.16.0/22 instead of 172.16.18.0/23.

3. The answer is in RTC's route table in Example 5-30. Notice that the route to 172.16.26.0/23 has not been updated in 2 minutes, 42 seconds. RTC's invalid timer is expiring before it hears a new update from RTD, and it is declaring the route to 172.16.26.0/23 invalid. Because no routes from RTA or RTB are being invalidated, the problem is with RTD's update timerthe update period is too long. When RTD finally sends an update, it is re-entered in RTC's route table and remains until RTC's invalid timer again expires.

# Chapter 6

1. RTA and RTB send and receive only RIPv2 messages. RTC receives both RIPv1 and RIPv2 but sends only RIPv1. Consequently, RTA and RTB do not have subnet 192.168.13.75/27 in their route tables. Although it receives the updates from RTA and RTB, RTC does not have subnets 192.168.13.90/28 and 192.168.13.86/29 in its route table because it interprets both of them as 192.168.13.64/27, which is directly connected to RTC's E0 interface.

2. Yes. Subnet 192.168.13.64/27 will be added to the route tables of RTA and RTB because they can now receive RIPv1 updates from RTC.

# Chapter 7

1. RTG is RTF's successor to subnet A.
2. RTC's feasible distance to subnet A is 309760.
3. RTG's feasible distance to subnet A is 2214319.
4. RTG's topology table shows RTD and RTE as feasible successors to subnet A.
5. RTA's feasible distance to subnet B is 2198016.

# Chapter 8

1. Either the IP address or the mask is misconfigured on one of the neighboring interfaces.
2. One router is configured as a stub area router, and the other is not.
3. The receiving router is configured for MD5 (type 2) authentication, and the neighbor has no authentication configured (type 0).
4. The passwords configured on the two routers do not match.
5. The neighboring interfaces are not configured with the same area ID.
6. RTA's network statements are in the wrong order. The first statement matches IP address 192.168.50.242 and puts it in area 192.168.50.0.
7. Link ID 10.8.5.1 is the most likely suspect because its sequence number is substantially higher than the sequence numbers of the other links.

# Chapter 10

1. Although IS-IS has formed an adjacency, its type is IS rather then L1, L2, or L1L2, indicating there is a problem. The IP addresses are not on the same subnet.

2. The router sends only L1 Hellos, indicating that it is an L1 router. It receives only L2 Hellos from 0000.3090.c7df, indicating that it is an L2 router.

# Chapter 11

1. Split horizon will prevent 192.168.10.0/24 from being advertised into the RIP domain from the IGRP domain.

2. No, because not all subnets of 10.0.0.0 are directly connected to Mantle on the RIP side.

3. The `network` statement under Robinson's EIGRP 1 configuration matches interface 192.168.3.32, even though no EIGRP Hellos are being forwarded out that interface. Therefore, the subnet is advertised as internal within the EIGRP 1 domain.

4. EIGRP automatically entered this summary route because subnets of 192.168.3.0 are being redistributed into EIGRP from OSPF.

5. A level-1 router in area 1 will not know of the summary route. RIP is being redistributed into level-2 (this is the default distribution level). The `summary-prefix` command is summarizing routes that are getting advertised into level-1.

# Chapter 13

1. The inverse mask of the first line of the access list is wrong. With this mask, all routes are matched. The line should be `access-list 1 deny 0.0.0.0 0.0.0.0`.

2. All routes not matched by the access lists are given a distance of 255 (unreachable). If a preferred route fails, Grimwig will not use an alternative path.

3. OSPF calculates its routes from the information learned from LSAs. The route filter has no effect on LSAs, and so the filter affects only the router on which it is configured.

4. The two route filters refer to the wrong interfaces. `distribute-list 1` should refer to E1, and `distribute-list 2` should refer to E0.

# Chapter 14

1 Both errors are errors of sequence. First, the `telnet` keyword in access list 101 is associated with the destination port; the correct association is with the source port. Second, the route map statements are in the wrong order. Telnet packets from 192.168.10.5 match the first statement and are forwarded to 192.168.16.254.

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

# Index

[**SYMBOL**] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**3-way handshaking**
**90-second timer (IGRP)**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**A class IP addresses**
  address masks
  subnetting
**ABRs (Area Border Routers)** 2nd
**access-class command**
**access-list compiled command**
**Aceves, J.J. Garcia Luna**
**Acknowledgment Number field (TCP)**
**acknowledgments**
  delayed
  direct
**ACLs (access control lists)**
  calling
    *access-class command*
  categories of
  comments, adding
  dynamic
  editing
  effect on switching
  effect on system performance
  extended IP 2nd
    *effect on system performance*
    *example of*
    *features*
    *sequence numbers*
  filter layers, numbering 2nd
  fragments, handling
  ICMP
  implicit deny any filter
  keywords
  named 2nd
  placement of
  reflexive
    *example*
    *parameters*
  sequence number
  sequentiality
  standard IP
    *Boolean AND/OR functions*
    *inverse masks*
  TCP
  Turbo ACLs
  UDP
**acquiring hands-on experience**
**active state**
  SIA
**adding comments to ACLs**
**address aggregation**
  CIDR
**Address Family Identifier field (RIP)**

**Address Family Identifier field (RIPv2)**
**address masks** [See also **subnet masks**]
**Address Resolution Protocol**. [See **ARP**]
**address spaces, designing lollipop-shaped**
**address summarization**
EIGRP, configuring
OSPF
*configuring*
*troubleshooting*
**addresses, IP**
internetworks
masks
reverse ARP
RIP
secondary RIP
subnet mask
subnets
**adjacencies** 2nd 3rd
building process
formation of
IS-IS 2nd
*troubleshooting*
troubleshooting
**adjusting RIPng metrics**
**administrative distance** 2nd
effect on redistribution
setting route preferences
**advertising default networks**
**aging process (LSAs)**
**alternative routes, creating for static routes**
**always keyword (default-information originate command)**
**AND operator**
Boolean AND function
**anycast addresses**
**application layer (TCP/IP)**
**Area ID**
**areas** 2nd
Area ID
autonomous systems
backbone
IS-IS
L1 routers
maximum size of
multiple L1 areas, IS-IS configuration
NSSAs
*configuring*
partitioned areas
path types
stub areas
*configuring*
totally stubby areas
*configuring*
traffic types
troubleshooting
**ARP (Address Resolution Protocol)** 2nd
gratuitous ARP
packets
*Hardware Address Length field*
*Hardware Type field*
*Operation field*
*Protocol Address Length field*

proxy ARP
reverse ARP
**arp command** 2nd
**arp timeout command**
**AS-External LSAs**
**ASBR Summary LSAs**
**ASBRs (Autonomous System Border Routers)**
**assigning metrics to redistributed routes**
**asynchronous updates**
**ATT (Attached bit)**
**Attempt state (OSPF neighbors)**
**authentication**
IS-IS, configuring
OSPF
*configuring*
RIPng
RIPv2
*configuring*
**Authentication Information TLV**
**auto-cost reference-bandwidth command**
**automatic summarization, disabling on EIGRP**
**Autonomous System External LSAs** 2nd
**autonomous systems**

PREV NEXT

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**B class IP addresses**
  address masks
  subnetting
**backbone**
  virtual links
**backbone routers**
**bandwidth** 2nd
  IGRP metric, modifying
  reference bandwidth
**BDRs (Backup Designated Routers), election process**
**best path determination, convergence**
**binary numbering system**
  converting to decimal
  converting to hex
**bitcount format**
**bits, IP packet flags**
**Boolean operators**
  AND
  OR
**boundaries**
**boundary routers**
**bounded updates**
**broadcast networks**
  IPv6 static routes, configuring
**broadcast updates**
**building process for OSPF adjacencies**
**bytes**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**C class IP addresses**
- address masks
- octet boundaries
- subnetting

**calculating**
- host addresses for subnets
- hosts per subnet requirements
- IGRP default metric
- place values

**calling ACLs**
- access-class command

**case studies**
- IPv6 redistribution with route maps
- policy routing
  - *and QoS*
  - *and redistribution*
  - *router configuration*
  - *standard IP access lists*
- protocol migration 2nd
  - *routing loops*
- RIP configuration
- route filtering
  - *filtering by routing process*
- route tagging

**categories of ACLs**

**CCIE exam, preparing for**
- certification path
- exam day
- hands-on experience
- study materials
- training classes

**CDP (Cisco Discovery Protocol)** 2nd
- enabling

**cdp enable command**

**cdp run command**

**CEF (Cisco Express Forwarding)**
- per destination load sharing

**Cerf, Vint** 2nd

**certification path for CCIE exam**

**characteristics of distance vector protocols**
- broadcast updates
- full routing table updates
- neighbors

**Checksum field**
- LSAs
- TCP

**CIDR (Classless Interdomain Routing)**

**circular sequence number space**

**Cisco IOS Software, switching method determination**

**classes of IP addresses**

**classful routing**
RIP 2nd
*replies*
*requests*
*subnetting*
*summarization*
**classless route lookups** 2nd
**classless routing** 2nd
redistribution into classful domains
*unmatching masks*
**clear arp-cache command** 2nd
**clear ip accounting command**
**CLNP (Connectionless Network Protocol)**
**columns in route tables**
**command field (RIPv2)**
**commands** 2nd
access class
access-list compiled
arp 2nd
arp timeout
auto-cost reference-bandwidth
cdp enable
cdp run
clear arp-cache 2nd
clear ip accounting
config-router
debug arp
debug iegpr neighbors
debug ip icmp
debug ip packet
debug ip policy
debug ip rip 2nd
debug ipv6 packet
default metric
default-information originate
*always keyword*
*for IPv6 addresses*
distance 2nd 3rd
distribute-list 2nd 3rd
eigrp stub
ip access-group
ip accounting access-violations
ip address 2nd
ip cef
ip classless 2nd
ip default network
ip default-network
ip load-sharing per packet
ip netmask-format 2nd
ip ospf cost
ip ospf dead-interval
ip ospf priority
ip ospf transmit-delay
ip policy route-map
ip policy-route map
ip proxy-arp 2nd
ip rip triggered
ip route
ip route cache
ip router isis
ipv6 cef

ipv6 nd prefix
ipv6 route command
ipv6 unicast-routing
isis isp-interval
lsp-gen-interval
maximum-paths
metric weights
metric-style wide
neighbor
network 2nd
network area
no ip proxy-arp
no ip redirects
offset-list 2nd
output-delay
passive-interface 2nd
redistribute connected
redistribute eigrp
router
router odr
router rip 2nd
set ip next-hop verify-availability
show arp
show cdp
show cdp neighbor detail
show clns is-neighbors
show clns is-neighbors detail
show interface
show ip access list
show ip accounting access-violations
show ip cef
show ip eigrp topology
show ip ospf database
show ip ospf database nssa-external
show ip ospf interface
show ip ospf neighbor
show ip route 2nd
show ipv6 cef
show ipv6 detail
show ipv6 interface
show ipv6 rip
show ipv6 route
show isis database
summary address
timers active time
timers basic
trace
variance
which route
**comparing**
OSPFv2 and OSPFv3
RIPng and RIPv2
**compatibility of RIPv2 and RIPv1**
**compatibility switches** 2nd
**config-router command**
**configuring**
ACLs, keywords
default routes
*in IPv6*
*ip default-network command*
dual-protocol routing

EIGRP
*address summarization*
*authentication*
*maximum paths for load sharing*
*multiple processes*
*stub routing*
*unequal-cost load balancing*
extended IP ACLs
*sequence number*
floating static routes
ICMP ACLs
IPv6 floating static routes
ISIS
*authentication*
*IPv4*
*IPv6*
*multiple area addresses*
*multiple L1 areas*
*multiple topology mode*
*route leaking*
*route summarization* 2nd
*routers*
load sharing
*CEF*
named ACLs
ODR
OSPF
*address summarization*
*authentication*
*Domain Name Service lookups*
*LSA filtering*
*network area command*
*NSSAs*
*on NBMA networks*
*over demand circuits*
*Router IDs*
*secondary addresses*
*stub areas*
*totally stubby areas*
*virtual links*
OSPFv3
*multiple instances on a link*
*on NBMA networks*
*stub area*
policy routing
*and QoS*
*and redistribution*
*global router configuration*
*standard IP access lists*
redistribution
*EIGRP into OSPF*
*IGRP into RIP*
*metrics*
reflexive ACLs
*example*
*parameters*
RIP
*discontiguous subnets*
*metrics*
*passive interfaces*
*unicast updates*

RIPng
*global parameters*
RIPv2
*authentication*
route maps
route tagging
standard IP ACLs
*Boolean AND/OR functions*
*inverse masks*
static routes
*IPv4*
*IPv6*
TCP ACLs
UDP ACLs
virtual links
**connection-oriented services, TCP**
**control protocols, ICMPv6 header format**
**convergence**
after router reboot in redistributed protocol environment
**converting**
binary to hex
decimal to binary
routing protocols
**CoS (Class of Service)**
**cost metric** 2nd 3rd 4th
**counting to infinity**
**cryptographic checksum, MD5**
**CSNPs (Complete SNPs)** 2nd

PREV NEXT

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**data links**
  address masks
    IP addresses
    *masks*
    *subnets*
  TCP/IP
**data structure of OSPF neighbors**
  components 2nd
**data-link identifiers, ARP**
**Database Description packets (OSPF)**
**DC-bit**
**DD (Database Description) packets**
**debug arp command**
**debug eigrp neighbors command**
**debug ip icmp command**
**debug ip packet command**
**debug ip policy command**
**debug ip rip command** 2nd
**debug ipv6 packet command**
**debugging route table entries**
**decimal numbering system**
  converting to binary
  converting to hexadecimal
  IP addresses
**decision points for OSPF adjacency formation**
**decision process (IS-IS)**
**default IGRP timers, modifying**
**default metrics, IGRP**
  calculating
  modifying
**default networks**
  advertising
**default routes** 2nd
  classless route lookups
  configuring
    *in IPv6*
    *ip default-network command*
    *IPv4*
  filtering
  flapping
  gateway of last resort
  in hub-and-spoke topologies
  large-scale use of
  redistribution
  risk of suboptimal routing
  static
    *"don't care" bits*
    *redistribution*
**default-information originate command**
  always keyword

for IPv6 addresses
**default-metric command**
**delay metric** 2nd
modifying
**delayed acknowledgments**
**demand circuits**
OSPF, configuring
presumption of reachability
**depletion of IPv4 addresses**
**designated routers (IS-IS)**
**designing**
lollipop-shaped address spaces
subnets
**Destination Address field (IPv6 packets)**
**destination ports, TCP**
**devices, boundary routers**
**DF bit**
**DiffServ (Differentiated Services framework)**
**diffusing computations**
feasible successor selection process
**digits**
**Dijkstra, E.W.**
**Dijkstra's algorithm**
**direct acknowledgments**
**directly connected subnets**
**disabling**
automatic summarization on EIGRP
split horizon
summarization on RIPv2
**discontiguous subnets** 2nd
configuring
RI
summarizing RIPv2 routes across
**displaying**
IGRP delay
IPv6 addresses
IS-IS link state database
IS-IS neighbor table
RIPng processes
**distance command** 2nd 3rd
**distance vector routing protocols** 2nd
asynchronous updates
characteristics of
*broadcast updates*
*full routing table updates*
*neighbors*
*periodic updates*
counting to infinity
holddown timers
RIP
*configuring*
*metrics, configuring*
route filtering
route invalidation timers
routing by rumor
split horizon
triggered updates
**distribute-list command** 2nd 3rd
**distributed database protocols.** [See **link-state protocols**]
**domain authentication, IS-IS configuration**
**Domain Name Service lookups, OSPF configuration**

**domains**
IS-IS, domain-wide prefix distribution extension
OSPF, areas
**Don't Fragment (DF) bit**
**dotted-decimal format** 2nd
**Down state (OSPF neighbors)**
**DRs (Designated Routers)**
DRothers
election process
**DSCP (Differentiated Services Code Point)**
**DUAL (Diffusing Update Algorithm)**
feasible successors, selection process
input events
**dual-protocol routing, configuring**
**Duplicate Address Detection**
**dynamic ACLs**
**Dynamic Hostname Exchange**
**dynamic routing protocols, RIP.** [See **RIP (Routing Information Protocol)**]

PREV NEXT

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**echo packets, tracing**
**Echo Request/Reply messages (ICMP)**
**ECN (Explicit Congestion Notification)**
**editing**
ACLs
route maps
**EGPs (Exterior Gateway Protocols)**
**EIGRP (Enhanced Interior Gateway Routing Protocol)**
address aggregation
*CIDR*
address summarization, configuring
adjacencies
authentication, configuring
automatic summarization, disabling
configuring
diffusing computations
DUAL
feasible successors, selection process 2nd
input events
load sharing, configuring maximum paths
metrics
*delay*
*FC*
*FD*
*feasible successors*
*variance*
missing neighbors, troubleshooting
multiple processes, configuring
neighbor discovery/recovery
neighbor table
packets
*header format*
*Hello*
*sequence numbers*
*TLVs*
protocol-dependent modules
Q Count
redistribution into OSPF
*configuring*
RTO
RTP
SRTT
stub routing, configuring
stuck-in-active neighbors, troubleshooting 2nd
summarization
timers, mulicast flow timer
topological table, contents of
troubleshooting
unequal-cost load balancing, configuring
updates

**eigrp stub command**
**election process for DRs/BDRs**
**electrical protocols**
**embedded IPv4 addresses**
**enabling**
CDP
classless route lookups
per packet load sharing
process switching
**encapsulation (SNAP), ARP**
**environments**
**EPHOS (European Procurement Handbook for Open Systems)**
**equal-cost load balancing** 2nd
**error correction in IP packets**
**ES (End System)**
**establishing adjacencies**
decision points
OSPF
**Ethernet**
**examples**
of diffusing computations 2nd
of extended IP ACLs
of reflexive ACLs
**exams, CCIE**
certification path
preparing for
training classes
**Exchange state (OSPF neighbors)**
**expiration timer, RIP**
**explicit acknowledgments** 2nd
**ExStart state (OSPF neighbors)**
**extended IP ACLs** 2nd
effect on system performance
example of
features
placement of
sequence numbers
**extended metrics**
**Extended Ping utility**
**extension headers**
functions performed
**extensions to IS-IS**
3-way handshaking
domain-wide prefix distribution
Dynamic Hostname Exchange
IPv6 routing
mesh groups
Multi Topology routing
wide metrics
**exterior routes**
**External Attributes LSAs**
**external routes**
summarization
**external traffic**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**failed routes, tracing**
  IPv6
**fast switching, per destination load sharing**
**FC (feasible condition)**
**FD (feasible distance)** 2nd
**feasible successors**
  diffusing computations 2nd
  selection process
**features**
  of extended IP ACLs
  of OSPF
  of TCP ACLs
**fields**
  of IS-IS Hello PDUs
  of IS-IS LSPs
  of RIPng messages
  of TCP header
**filter layers, numbering** 2nd
**filtering**
  implicit deny any
  IPv6 routes
  prefix lists
  sequentiality
**first octet rule**
**Flags field (TCP)**
**flapping routes**
**flash updates.** [See **triggered updates**]
**Fletcher algorithm**
**floating static routes**
  IPv6
**flooding**
  LSAs
    *acknowledgments*
    *checksum*
  sequence numbers 2nd
    *circular sequence number space*
    *linear sequence number space*
    *lollipop-shaped sequence number space*
**flooding delays (IS-IS)**
**Flow Label field (IPv6 packets)**
**Floyd, Sally**
**flush timer (RIP)**
**format**
  of IPv6 multicast addresses
  of OSPFv3 LSAs
    *AS-External LSAs*
    *Inter-Area Prefix LSAs*
    *Inter-Area Router LSAs*
    *Intra-Area Prefix LSAs*
    *Link LSAs*

*Network LSAs*
*Router LSAs*
of RA messages
of Redirect messages
of RIPng messages
of RIPv2 messages
**formation of OSPF adjacencies**
decision points
**Fragment Offset field (IP packets)**
**fragmenting IP packets**
**fragments, processing through ACLs**
**Frame-Relay mapping, OSPFv3**
**full routing table updates**
**Full state (OSPF neighbors)**
**functional protocols (TCP/IP)**

PREV NEXT

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**garbage collection (RIP)**
**gateway of last resort**
**general TLV fields**
**global parameters for RIPng configuration**
**global unicast addresses**
**GOSIP (Government Open Systems Interconnection Profile)**
**gratuitous ARP**
**Group Membership LSAs**
**group pacing**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [**H**] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**Hardware Address Length field (ARP packets)**
**Hardware Type field (ARP packets)**
**Header Checksum field (IP packets)**
**Header Length field**
IP packets
TCP packets
**headers**
EIGRP packet format
IP packets
*flags*
*Fragment Offset field*
*Header Checksum field*
*identifiers*
*Options field*
*options field*
*size*
*TOS*
*total length*
*TTL field*
IS-IS PDU fields
TCP
UDP
**Hedrick, Charles**
**Hello packets**
EIGRP
*hold time*
OSFP 2nd
**Hello PDUs**
Area Addresses TLV
Authentication Information TLV
fields
Intermediate System Neighbors TLV
IP Interface Address TLV
Padding TLV
Protocols Supported TLV
**Hello protocol** 2nd
**HelloInterval**
**hexadecimal numbering system**
converting to decimal
hexadecimal IPv6 representation
IP address format
**hippity bit**
**holddown**
IGRP
RIP
**holddown timers**
**hop count metric** 2nd
**Hop Limit field (IPv6 packets)**
**host addresses, subnet mask octet boundaries**
**host-to-host layer (TCP/IP)**

TCP
UDP
**Host-to-Host Layer Protocol**
**hosts**
IP addresses
quantity per subnet requirement, calculating
**"hot potato" routing**
**hub-and-spoke topologies**
ODR
stub routers
**Huitema, Christian**

PREV NEXT

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**ICMP (Internet Control Message Protocol)**
- ACLs
- echo packets, tracing
- Echo Request/Reply messages
- redirect packets
- Router Advertisement packets
- Router Selection packets

**ICMPv6 header format**
**identifying IPv6 address types**
**IGPs (Interior Gateway Protocols)**
**IGRP (Interior Gateway Routing Protocol)**
- default timers, modifying
- delay
- holddown
- load
- metrics
  - *modifying*
- process domains
- redistribution into RIP, configuring
- reliability
- similarities to RIP
  - timers
  - *90-second*
  - *sleeptime*

**Implicit Acknowledgments**
**implicit acknowledgments**
**implicit deny any filter**
**improving SPF efficiency on IS-IS networks**
**InfTransDelay**
**Init state (OSPF neighbors)**
**input events**
- for OSPF interface state machine
- for OSPF neighbors

**inter-area paths**
**Inter-Area Prefix LSAs**
**Inter-Area Router LSAs**
**inter-area summarization**
**inter-area traffic**
**Inter-Domain Routing Protocol Information TLV**
**Interface ID**
**interface state machine (OSPF)**
- input events

**interface-specific summarization**
**interfaces, OSPF**
- data structure
- input events
- states

**interior routes**
**Intermediate System Neighbors TLV**
**internal routers**

**internal routes**
**Internet layer (TCP/IP)**
**internetwork IP addresses**
**intra-area paths**
**Intra-Area Prefix LSAs**
**intra-area traffic**
**Intradomain Routing Protocol Discriminator**
**invalid timers**
IGRP
RIP
**invalid updates, sources of**
**inverse masks**
**ip access-group command**
**ip accounting access-violations command**
**ip address command** 2nd
**IP Address field**
RIP
RIPv2
**IP addresses**
bitcount format
dotted decimal format
dotted-decimal format
first octet rule
hexadecimal format
internetworks
masks
prefix lists
reverse ARP
RIP
subnet masks, troubleshooting
subnets 2nd
*designing*
summary routes
**ip cef command**
**ip classless command** 2nd
**ip default-network command** 2nd
**IP External Reachability TLV**
U/D bit
**IP External Routes TLV**
**IP Interface Address TLV**
**IP Internal Reachability TLV**
U/D bit
**IP Internal Routes TLV**
**ip load-sharing**
**ip load-sharing command**
**ip load-sharing per-packet command**
**ip netmask-format**
**ip netmask-format command** 2nd
**ip ospf cost command**
**ip ospf dead-interval command**
**ip ospf priority command**
**ip ospf transmit-delay command**
**IP packets**
flags
Fragment Offset field
Header Checksum field
header length
identifiers
Options field
Protocol field
TOS (Type of Service)

total length
TTL (Time to Live) field
versions of
**ip policy route-map command** 2nd
**ip proxy-arp command** 2nd
**ip redirects command**
**ip rip triggered command**
**ip route command**
**ip route-cache command**
**ip router isis command**
**IP-specific TLV fields**
**IPv4**
configuring on IS-IS
depletion of
static routes, configuring
**IPv6**
address representation
anycast addresses
configuring on IS-IS
default-information originate command, default route configuration
double colon representation
embedded IPv4 addresses
extension headers
failed routes, tracing
floating static routes
global unicast addresses
identifying address types
local unicast addresses
modifying static entries on replaced routers
multicast addresses
NDP
*address autoconfiguration*
*Duplicate Address Detection*
*messages*
*Neighbor Address Resolution*
*Neighbor Unreachability Detection*
*privacy addresses*
*router discovery*
need for
packet header format
prefixes
redistribution with route tags
RIPng
*configuring*
*next-hop route entries*
*route summarization*
route filtering
routing with IS-IS
static routes, configuring
unspecified addresses
**ipv6 cef command**
**ipv6 nd prefix command**
**IPv6 Prefix field (RIPng messages)**
**ipv6 route command**
**ipv6 unicast-routing command**
**IS (intermediate system)**
**IS-IS (intermediate System-to-Intermediate System)**
adjacencies
*troubleshooting*
areas
authentication, configuring

extensions
*3-way handshaking*
*domain-wide prefix distribution*
*Dynamic Hostname Exchange*
*IPv6 routing*
*mesh groups*
*Multi Topology routing*
*wide metrics*
flooding delay
*controlling*
IPv4, configuring
IPv6, configuring
layered network architecture
*subnetwork dependent functions*
*subnetwork independent functions*
link-state database
*displaying*
*troubleshooting* 2nd
LSPs
*fields*
*Inter-Domain Routing Protocol Information TLV*
*Intermediate System Neighbors TLV*
*IP External Reachability TLV*
*IP Internal Reachability TLV*
*TLVs*
multiple area addresses, configuring
multiple L1 areas, configuring
multiple topology mode, configuring
neighbor table, displaying
NETs
*format*
*selecting*
on NBMA networks, troubleshooting
PDUs
*format*
*header fields*
*Hello PDU format*
*Hello PDUs*
*IS-IS LSPs*
*IS-IS Sequence Number PDU format*
*TLV fields*
redistribution into RIP/RIPng
route leaking, configuring
route summarization, configuring
router types, configuring
similarities with OSPF
SNPs
SPF efficiency, improving
System ID
troubleshooting
virtual link support
**IS-IS Sequence Number PDUs**
LSP Entries TLV
**isis lsp-interval command**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**Jacobson, Van**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [**K**] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**Kahn, Bob** 2nd
**key chains, configuring for RIPv2 authentication**
**keywords for ACL configuration**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**L1 LSPs, TLVs**
**L1 routers**
**L1/L2 routers**
**L2 LSPs, TLVs**
**L2 routers**
**LAN Hellos**
**layered network architecture, IS-IS**
**Level 3 Routing**
**linear sequence number space**
**link costs**
**Link LSAs** 2nd
**Link State Acknowledgment packets (OSPF)**
**link state refresh process**
**Link State Request packets (OSPF)**
**Link State Update packets (OSPF)**
**link-local unicast addresses**
**link-state advertisements** [See **LSAs**]
**link-state database** 2nd 3rd 4th 5th
  ISIS
  *displaying*
  *troubleshooting* 2nd
  LSA group pacing
**link-state routing protocols** 2nd
  areas
  Dijkstra's algorithm
    *router adaptation*
  flooding
    *sequence numbers*
    IS-IS.[See IS-IS]
  LSAs
    *aging process*
  neighbor discovery
    OSPF.[See OSPF]
  route filtering
**links, OSPF**
**LIRs (Local Internet Registries)**
**load (IGRP)**
**load balancing** 2nd
  unequal-cost, EIGRP configuration
**load sharing**
  CEF
  configuring
  maximum paths, EIGRP configuration
  per destination 2nd
    perpacket
    *enabling*
    *process switching*
**Loading state (OSPF neighbors)**
**local computations**
**local unicast addresses**

**lollipop-shaped sequence number space**
**lookups**
classless
recursive
**loopback interfaces** 2nd
**LSAs (link-state advertisements)** 2nd 3rd 4th
acknowledgments
*during flooding process*
aging process
ASBR Summary LSAs
Autonomous System External LSAs
External Attributes LSAs
filtering, OSPF configuration
flooding 2nd 3rd
*checksum*
*sequence numbers* 2nd
Group Membership LSAs
group pacing
header format
Network LSAs 2nd
Network Summary LSAs 2nd
NSSA External LSAs 2nd
Opaque LSAs
Options field
OSPFv3
*format*
*Inter-Area AS-External LSAs*
*Inter-Area Prefix LSAs*
*Inter-Area Router LSAs*
*Intra-Area Prefix LSAs*
*Link LSAs*
*Network LSAs*
*Router LSAs*
Router LSAs 2nd
sequence numbers
types of
**LSP Entries TLV**
**lsp-gen-interval command**
**LSPs**
ATT
controlling
flooding delays

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [**M**] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**MAC addresses, ARP**
**MAC-to-EUI64 conversion**
**masks** [See also **subnet masks**]
  octet boundaries
**MaxAge timer, link state refresh**
**maximum-paths command**
**MD5**
**mechanical protocols (TCP/IP)**
**medium internetworks**
**medium networks**
**mesh groups**
**messages**
  ICMP, Echo Request/Reply
  NDP
  OSPFv3
  RIP
  RIPng
  RIPv2
**Metric field**
  RIP
  RIPng
**metric weights command**
**metric-style wide command**
**metrics** 2nd [See also **administrative distance**]
  and redistribution
  bandwidth
  cost 2nd 3rd 4th
  delay
  EIGRP
    *delay*
    *FC*
    *FD* 2nd
    *feasible successors*
    *variance*
  hop-count
  IGRP
    *default metric, calculating*
    *load*
    *modifying*
    *reliability*
  redistribution, configuring
  reliability
  RIP 2nd
  RIPng, adjusting
  triggered updates
**MF bit (More Fragments)**
**microflows**
**migrating to new routing protocols**
**minimizing impact of updates on RIP**
**minimumLSPTransmissionInterval**

**misconfigured OSPF summarization, troubleshooting**
**missing EIGRP neighbors, troubleshooting**
**missing route table entries**
**modifying**
cost metric
IGRP default metrics
IGRP default timers
RouterDeadInterval
**More Fragments (MF) bit**
**MOSPF (Multicast OSPF)**
**MT (Multi Topology) routing**
**MTU (Maximum Transmission Unit)**
IP packets
**multiaccess**
**multicast addresses**
**multicast flow timer**
**multiple area addresses, configuring on IS-IS**
**multiple EIGRP processes, configuring**
**multiple L1 areas, IS-IS configuration**
**multiple topology mode, IS-IS configuration**
**mutual redistribution** 2nd
at multiple redistribution points

PREV NEXT

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**NA (Neighbor Advertisement) messages**
  format
**named ACLs** 2nd
**NAT (Network Address Translation)**
**NBMA networks**
  IS-IS, troubleshooting
  OSPF, configuring 2nd
**NDP (Neighbor Discovery Protocol)**
  address autoconfiguration, MAC-to-EUI64 conversion
  Duplicate Address Detection
  messages
  Neighbor Address Resolution
  neighbor discovery
  Neighbor Unreachability Detection
  privacy addresses
**Neighbor Address Resolution**
**neighbor command**
**neighbor state machine (OSPF)**
**neighbor table**
  EIGRP
**Neighbor Unreachability Detection**
**neighbors**
    adjacencies
    *formation of*
    *troubleshooting*
    discovery/recovery
    *adjacencies*
    *EIGRP*
  EIGRP, adjacencies
  IS-IS
  OSPF
    *adjacency building process*
    *data structure*
    *input events*
**NETs (Network Entity Titles)**
  format
  selecting
**network area command**
**network command** 2nd
**Network LSAs** 2nd 3rd
**Network Summary LSAs** 2nd
**Next Header field (IPv6 packets)**
**Next Hop field (RIPv2)**
**next-hop addresses on replaced routers, troubleshooting**
**next-hop route entries (RIPng)**
**NLA (Next-Level Aggregator)**
**no ip proxy-arp command**
**no ip redirects command**
**nonperiodic updates**
**not-so-stubby areas**

**NPDUs (Network Protocol Data Units)**
**NS (Neighbor Solicitation) messages**
**NS messages, format**
**NSSA External LSAs** 2nd
**NSSAs (not-so-stubby areas)**
configuring
**null interface**
**numbering ACL filter layers**
**numbering systems**
binary
hexadecimal
place values

PREV NEXT

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [**O**] [P] [Q] [R] [S] [T] [U] [V] [W]

**octets**
 binary
 boundary subnet masks
 hexadecimal
 subnetting
**ODR (On-Demand Routing)**
 CDP
 configuring
 redistribution
**offset lists, RIP**
**offset-list command** 2nd
**OL (Overload) bit**
**opaque LSAs**
**Operation field (ARP packets)**
**operation of OSPFv3**
**operators**
 Boolean AND function 2nd
 Boolean OR function
**optical protocols**
**Options field**
 IP packets
 LSAs
 TCP
**OSPF (Open Shortest Path First).** [See also **OSPFv3**]
 ABRs
 address summarization, configuring
 adjacencies
  *formation of, decision points*
  *troubleshooting*
 area-wide problems, troubleshooting
 areas
  *Area ID*
  *backbone*
  *maximum size of*
  *not-so-stubby areas*
  *partitioned areas*
  *path types*
  *stub areas*
  *totally stubby areas*
  *traffic types*
 ASBRs
 authentication
  *configuring*
 backbone routers
 BDRs
  *election process*
 broadcast networks
 configuring
  *on NBMA networks*
  *over demand circuits*

cost metric, modifying
DD packets
Domain Name Service lookups, configuring
DRs
*election process*
features of
Hello packets
Hello protocol
interface state machine
*input events*
interfaces, data structure
internal routers
isolated areas, troubleshooting
link state database
*LSA group pacing*
link state refresh process
link-state database 2nd
links
loopback interfaces
LSAs
*ASBR Summary LSAs*
*Autonomous System External LSAs*
*External Attributes LSAs*
*filtering*
*flooding process*
*Group Membership LSAs*
*group pacing*
*header format*
*Network LSAs* 2nd
*Network Summary LSAs* 2nd
*NSSA External LSAs* 2nd
*Opaque LSAs*
*Options field*
*Router LSAs* 2nd
*sequence numbers*
*types of*
metrics, cost 2nd
misconfigured summarization, troubleshooting
MOSPF
NBMA networks
neighbor table
neighbors
*data structure*
*input events*
*state machine*
NSSAs
*configuring*
over demand circuits
oversubscription
packets
*Database Description packets*
*header format*
*Hello packets*
*Link State Acknowledgment packets*
*Link State Request packets*
*Link State Update packets*
point-to-multipoint networks
point-to-point networks
PollInterval
presumption of reachability
pseudonodes

redistribution into EIGRP, configuring
reference bandwidth
refresh process versus IS-IS Update process
route table entries, lookups
Router IDs
*configuring*
RouterDeadInterval
routingtableentries
*destination types*
*path types*
secondary addresses, configuring
similarities with IS-IS
stub areas, configuring
stub networks
totally stubby areas, configuring 2nd
transit networks
troubleshooting
virtual links 2nd
*configuring* 2nd
**OSPFv3**
configuring
Frame-Relay mapping
LSAs
*AS-External LSAs*
*format*
*Inter-Area Prefix LSAs*
*Inter-Area Router LSAs*
*Intra-Area Prefix LSAs*
*Link LSAs*
*Network LSAs*
*Router LSAs* 2nd
messages
multiple instances on a link, configuring
on NBMA networks, configuring
operation
redistribution into RIPng
stub areas configuring
troubleshooting
versus OSPFv2
**out keyword**
**output-delay command**
**oversubscription**

PREV NEXT

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**P-bit**
**packet filtering routers**
**packets**
ARP
*Hardware Address Length field*
*Hardware Type field*
*Operation field*
*Protocol Address Length field*
EIGRP
*header format*
*Hello*
*sequence numbers*
*TLVs*
fragments, processing through ACLs
Hello (OSPF)
ICMP
*Echo Reply*
*Echo Request*
*redirect*
*Router Advertisement*
*Router Selection*
IP
*flags*
*Fragment Offset field*
*Header Checksum field*
*header length*
*identifiers*
*Options field*
*Protocol field*
*TOS (Type of Service)*
*total length*
*TTL field*
IPv6
*Destination Address field*
*Flow Label field*
*Hop Limit field*
*Next Header field*
*Payload Length field*
*Source Address field*
*Traffic Class field*
LSAs
*acknowledgment during flooding process*
*AS-External LSAs*
*ASBR Summary LSAs*
*Autonomous System External LSAs*
*External Attributes LSAs*
*Group Membership LSAs*
*header format*
*Inter-Area Prefix LSAs*
*Inter-Area Router LSAs*

*Intra-Area Prefix LSAs*
*Link LSAs*
*Network LSAs* 2nd 3rd
*Network Summary LSAs* 2nd
*NSSA External LSAs* 2nd
*Opaque LSAs*
*Options field*
*OSPFv3*
*Router LSAs* 2nd 3rd
*sequence numbers*
*types of*
OSPF
*Database Description*
*Database Description packets*
*header format*
*Hello packets*
*Link State Acknowledgment packets*
*Link State Request packets*
*Link State Update packets*
PDUs [See PDUs (Protocol Data Units)]
**Padding TLV**
**parameters**
of extended IP ACLs
of reflexive ACLs
**Partial Route Calculation**
**partial updates**
**partitioned areas**
**passive interfaces, configuring**
**passive state**
**passive-interface command** 2nd
**path types (OSPF)**
**Payload Length field (IPv6 packets)**
**PDUs (Protocol Data Units)**
format
header fields
Hello PDUs
*Area Addresses TLV*
*Authentication Information TLV*
*fields*
*format*
*Intermediate System Neighbors TLV*
*IP Interface Address TLV*
*Padding TLV*
*Protocols Supported TLV*
IS-IS LSPs
*fields*
*Inter-Domain Routing Protocol Information TLV*
*Intermediate System Neighbors TLV*
*IP External Reachability TLV*
*IP Internal Reachability TLV*
*TLVs*
IS-IS Sequence Number PDU format
*LSP Entries TLV*
TLV fields
**per destination load sharing** 2nd
CEF
**per packet load sharing**
enabling
process switching
**periodic updates**
**Perlman, Radia** 2nd

**PHB (Per-Hop Behavior)**
**physical layer (TCP/IP)**
**place values of numbering systems**
**placement of ACLs**
**point-to-multipoint networks**
**point-to-point connections, TCP**
**point-to-point Hellos**
**point-to-point networks**
**policy routing**
and QoS
and redistribution
case study
match commands
set commands
standard IP access lists, case study
**PollInterval**
**port numbers**
**ports**
sockets
TCP
**Prefix Length field (RIPng messages)**
**prefix lists**
**preparing for CCIE exam**
certification path
exam day
hands-on experience
study materials
training classes
**presumption of reachability**
**privacy addresses**
**procedural protocols**
**process domains**
**process switching**
**promiscuous ARP**
**Protocol Address Length field (ARP packets)**
**protocol conflicts, troubleshooting** 2nd
**Protocol field**
**protocol migration, case study**
**protocol-dependent modules**
**Protocols Supported TLV**
**proxy ARP**
resolving protocol conflicts
**pseudonodes**
**PSNPs (Partial SNPs)** 2nd

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [**Q**] [R] [S] [T] [U] [V] [W]

**Q Count**
**QoS and policy routing**
**queries, EIGRP**
**query origin flags**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [**R**] [S] [T] [U] [V] [W]

**RA (Router Advertisement) messages**
**RARP (Reverse ARP)**
**recursive table lookups**
**Redirect messages**
  format
**redirects (ICMP)**
**redistribute connected command**
**redistribute eigrp command**
**redistribution**
  administrative distances
  and policy routing
  and summarization
  case study
  classless to classful protocols
    *unmatching masks*
  configuring
  conflicting metrics
  convergence after router reboot
  EIGRP into OSPF
    *configuring*
  IGRP into RIP, configuring
  IPv6 addresses with route maps
  IS-IS into RIP/RIPng
  match commands
  metrics, configuring
  mutual redistribution
    *at multiple redistribution points*
  of default routes
  of default static routes
  of ODR-discovered routes
  of static routes
    *into dynamic routing protocols*
  of static routes into OSPF
  OSPFv3 into RIPng
  route feedback
  route tagging
    *tagged routes, filtering*
  routing loops
  set commands
  SIN routing
**reference bandwidth**
**reflexive ACLs**
  example
  parameters
**reliability as IGRP metric** 2nd
**reliable multicast**
**remarks, adding to ACLs**
**replaced routers, troubleshooting next-hop addresses**
**replies**
  EIGRP

gratuitous ARP
ICMP
RIP
**representation of IPv6 addresses**
double colon
prefixes
**Request messages (RIP)** 2nd
**requests**
ARP
EIGRP
gratuitous ARP
ICMP
RIP
**Reserved field, TCP**
**Response messages, RIP**
**reverse ARP**
**reverse routes**
**RFCs, ICMP-related**
**RIB (routing information base).** [See **route tables**]
**RIP (Routing Information Protocol)** 2nd [See also **RIPng**, **RIPv1**, **RIPv2**]
classful routing
*subnetting*
*summarization*
configuring
*discontiguous subnets*
*metrics*
equal-cost load balancing
hop counts
messages
metrics, configuring
offset lists
passive interfaces, configuring
redistribution into IGRP, configuring
redistribution into IS-IS
replies
Request messages
requests
silent hosts
similarities to IGRP
timers
triggered updates
troubleshooting
updates
*minimizing impact of*
*unicast, configuring*
**RIP_JITTER variable**
**RIPng (Routing Information Protocol next generation)**
authentication
configuring
global parameters, configuring
message format
metrics, adjusting
misconfigured VLSM, troubleshooting
next-hop route entries
processes, displaying
redistribution
*into IS-IS*
*into OSPFv3*
route summarization
**RIPv1, compatibility with RIPv2**
**RIPv2**

authentication
*configuring*
compability switches
compatibility with RIPv1
configuring
messages
misconfigured VLSM, troubleshooting
similarity to RIPng
summarization
VLSM
**RIRs (Regional Internet Registries)**
**route feedback**
**route filtering**
case study
filtering by routing process, case study
specific routes, filtering
tagged routes, filtering
**route flapping**
**route invalidation timers**
**route leaking** 2nd
IS-IS configuration
**route lookups, classless**
**route maps**
configuring 2nd
editing
redistribution of IPv6 addresses
sequence number, configuring
**route redistribution.** [See **redistribution**]
**route summarization.** [See **summarization**]
**route tables**
administrative distance
columns
entries
*debugging*
*of static routes*
floating static routes, IPv6
metrics
missing entries
static routes
*configuring*
*creating alternative routes*
*floating static routes*
*load sharing, configuring*
*recursive table lookups*
*summarizing*
*troubleshooting*
**Route Tag field (RIPng messages)**
**Route Tag field (RIPv2)**
**route tagging** 2nd
**Router Advertisement packets (ICMP)**
**router command**
**Router IDs**
configuring
**Router LSAs** 2nd 3rd
**router odr command**
**router preferences, setting with administrative distance**
**router rip command** 2nd
**Router Selection packets (ICMP)**
**RouterDeadInterval**
**routers**
boundary routers

configuring on IS-IS
IS-IS
*designated routers* 2nd
*hold time*
*System ID*
**"routing by rumor,"**
**routing domains versus process domains**
**Routing Information Protocol.** [See **RIP**]
**routing loops**
counting to infinity
in redistributed protocol environments
troubleshooting
**routing policies**
**routing protocols**
classless
*redistribution into classful domains*
distance vector
*asynchronous updates*
*characteristics of*
*counting to infinity*
*holddown timers*
*route invalidation timers*
*routing by rumor*
*split horizon*
*triggered updates*
link state
*areas*
*autonomous systems*
*Dijkstra's algorithm*
*flooding*
*LSAs*
*neighbor discovery*
metrics
*bandwidth*
*cost*
*delay*
*hop count*
*reliability*
selecting
**routing table entries (OSPF)**
destination types
lookups
path types
**RS (Router Solicitation) messages**
**RTO (retransmission timeout)**
**RTP (Reliable Transport Protocol)**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**Scholten, C.S.**
**secondary addresses**
OSPF configuration
RIP
**security, authentication**
configuring on EIGRP
configuring on IS-IS
OSPF configuration 2nd
RIPv2 2nd
**selecting**
NETs
routing protocols
**Sequence Number field (TCP)**
**sequence numbers** 2nd 3rd 4th 5th
circular sequence number space
EIGRP
linear sequence number space
lollipop-shaped sequence number space
TCP
**sequentiality of access list filters, importance of**
**set ip next-hop verify-availability command**
**shortest path first protocols.** [See **link-state routing protocols**]
**show arp command**
**show cdp command**
**show cdp neighbor detail command**
**show clns is-neighbors command**
**show clns is-neighbors detail command**
**show interface command**
**show ip access-list command**
**show ip accounting access-violations command**
**show ip cef command**
**show ip eigrp topology command**
**show ip osfp interface command**
**show ip ospf database command**
**show ip ospf database nssa-external command**
**show ip ospf neighbor command**
**show ip route command** 2nd
**show ipv6 cef command**
**show ipv6 cef detail command**
**show ipv6 interface command**
**show ipv6 rip command**
**show ipv6 route command**
**show isis database command**
**SIA (stuck-in-active)**
troubleshooting
**silent hosts**
**Simple Internet Protocol (SIP)**
**SIN (ships in the night) routing**
**SIP (Simple Internet Protocol)**
**site-local unicast addresses**

**sizing IP packet headers**
**sleeptime, IGRP**
**small internetworks**
**SNAP (subnetwork access protocol)**
ARP encapsulation
**SNPA (Subnetwork Point of Attachment)**
**sockets**
**solicited-node multicast addresses**
**Source Address field (IPv6 packets)**
**source ports, TCP**
**sources of invalid updates**
**SPF efficiency, improving on IS-IS networks**
**split horizon**
disabling
resolving routing loops in redistributed environments
with poisoned reverse 2nd
**spoofing attacks**
**SRTT (smooth round-trip time)**
**standard IP ACLs**
Boolean AND/OR functions
inverse masks
placement of
policy routing configuration
**static default routes**
"don't care" bits
**static routes**
administrative distance
configuring
failed routes, tracing
floating static routes, creating
IPv4, configuring
IPv6
*configuring*
*floating static routes*
*troubleshooting on replaced routers*
load sharing
*CEF*
*configuring*
*per destination*
*per packet*
policy routes
policy routing
*match commands*
*set commands*
protocol conflicts, resolving 2nd
recursive table lookups
redistribution
redistribution into dynamic routing protocols
redistribution into OSPF
summarizing
troubleshooting
**static routing**
creating alternative routes
per destination load sharing
**stub areas** 2nd
configuring
**stub networks**
**stub routers**
configuring on EIGRP
ODR
unreachable destinations

**stuck-in-active EIGRP neighbors, troubleshooting**
**study materials for CCIE exam**
**sub-subnetting**
**subinterfaces**
**subnet**
**subnet hiding**
**Subnet ID field**
**Subnet Mask field (RIPv2)**
**subnet masks**
octect boundaries
troubleshooting
unmatching, advertising
VLSM
**subnets** 2nd 3rd
bitcount format
classful routing protocols
designing
discontiguous 2nd
*summarizing RIPv2 routes across*
dotted decimal format
hexadecimal format
IP addresses
number of required hosts, calculating
RIP, classful routing
**subnetting, VLSM**
and RIPv2 routing
troubleshooting
**subnetwork dependent functions of IS-IS layered network architecture**
Decision process
designated routers 2nd
neighbors
Update process
**Subnetwork Independent sublayer (IS-IS)**
**summarization**
address aggregation
*CIDR*
EIGRP
in redistributed environments
interface-specific
null interfaces
of RIPv2 routes, disabling
OSPF
*configuring*
*troubleshooting*
RIP
RIPng
**summary addresses**
**summary routes**
**summary-address command**
**supernetting**
**System ID (IS-IS)**
**system routes**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**TCP (Transmission Control Protocol)**
ACLs
header information
**TCP/IP (Transmission Control Protocol/Internet Protocol)**
application layer
ARP
*proxy ARP*
data link layer
host-to-host layer 2nd
*TCP* 2nd
*UDP*
ICMP Echo Requests/Replies
Internet layer
physical layer
**three-way handshaking**
**timers**
EIGRP, multicast flow timer
holddown timers
IGRP
OSPF, group pacing timer
RIP
*expiration timer*
route invalidation
**timers active time command**
**timers basic command**
**timing jitter**
**TLA (Top Level Aggregator)**
**TLV (Type/Length/Value) fields**
general fields
IP External Routes TLV
IP Internal Reachability, U/D bit
IP Internal Routes TLV
IP-specific fields
of IS-IS LSPs
*Inter-Domain Routing Protocol Information*
*Intermediate System Neighbors*
*IP External Reachability*
*IP Internal Reachability*
of IS-IS PDUs
*Area Addresses TLVs*
*Authentication Information TLV*
*Intermediate System Neighbors TLVs*
*IP Interface Address TLV*
*Padding TLV*
*Protocols Supported TLV*
**topological table, contents of**
**totally stubby areas**
configuring 2nd
**trace command**
**tracing failed routes**

IPv6
**Traffic Class field (IPv6 packets)**
**traffic, load balancing**
**training classes for CCIE exam**
**transit networks**
**transition technologies**
**triggered updates**
RIP
**troubleshooting**
EIGRP
*missing neighbors*
*stuck-in-active neighbors*
IS-IS
*adjacencies*
*link-state database* 2nd
*on NBMA networks*
OSPF
*adjacencies*
*area-wide problems*
*isolated areas*
*misconfigured summarization*
OSPFv3, Frame-Relay mapping
RIP
RIPng, misconfigured VLSM
RIPv2, misconfigured VLSM
static routes
*failed routes*
*protocol conflicts* 2nd
subnet masks
**TTL (Time to Live) field**
**Type 1 external paths**
**Type 2 external paths**
**types of LSAs**
ASBR Summary LSAs
External Attributes LSAs
Group Membership LSAs
Network LSAs
Network Summary LSAs
NSSA External LSAs
Opaque LSAs
Router LSAs
**types of traffic in OSPF areas**

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [**U**] [V] [W]

**U/D (Up/Down) bit**
**UDP (User Datagram Protocol)**
 ACLs
 header information
**unequal-cost load balancing, EIGRP configuration**
**unequal-cost load sharing**
**unicast addresses**
 global addresses
 updates, configuring
**unreachable networks**
**unspecified addresses**
**Update process (IS-IS)**
**update timers, RIP**
**updates**
 EIGRP
 invalid, sources of
 minimizing impact of
 nonperiodic
 RIP
**uptime**
**Urgent Pointer field, TCP**
**User Datagram Protocol.** [See **UDP**]

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**variables, RIP_JITTER**
**variance command**
**verifying EIGRP stub router configuration**
**Version field**
  RIP
  RIPv2
**versions of IP**
**viewing**
  IS-IS link-state database
  IS-IS neighbor table
**virtual links** 2nd
  configuring
  IS-IS support
  OSPF, configuring
**VLSM (variable-length subnet masking)**
  and RIPv2 routing
  troubleshooting

# Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

**which-route command**
**wide metrics**
**Window Size field, TCP header**
**windowing, TCP**