



CCNP Routing and Switching ROUTE 300-101 Quick Reference

Denise Donohue

Cisco Press

CCNP

Routing and Switching ROUTE 300-101 Quick Reference

Denise Donohue, CCIE No. 9566

Cisco Press

800 East 96th Street

Indianapolis, Indiana 46240 USA

CCNP Routing and Switching ROUTE 300-101 Quick Reference

Denise Donohue

Copyright © 2015 Pearson Education, 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

First Printing November 2014

ISBN-13: 978-0-13-392947-8

ISBN-10: 0-13-392947-7

Warning and Disclaimer

This book is designed to provide information about networking. 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.

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.

Publisher

Paul Boger

Associate Publisher

Dave Dusthimer

Business Operation Manager, Cisco Press

Jan Cornelissen

Executive Editor

Brett Bartow

Managing Editor

Sandra Schroeder

Development Editor

Marianne Bartow

Senior Project Editor

Tonya Simpson

Copy Editor

Paula Lowell

Technical Editor

Sean Wilkins

Editorial Assistant

Vanessa Evans

Cover Designer

Mark Shirar

Composition

Studio Galou

Indexer

Brad Herriman

Proofreader

Megan Wade-Taxter

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact 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 email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.



Americas 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: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
199 Robinson Road
#28-01 Capital Tower
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Europe Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: +31 0 800 020 0791
Fax: +31 0 20 357 1100

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

©2007 Cisco Systems, Inc. All rights reserved. CCVP, the Cisco logo, and the Cisco Square Bridge logo are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, IQ Expertise, the IQ logo, IQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networking Academy, Network Registrar, Packet, PIX, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0609R)

About the Author

Denise Donohue, CCIE No. 9566, is a senior solutions architect with Chesapeake NetCraftsmen. She has worked in IT since the mid-1990s. Focusing on network design since 2004, she has consulted on a wide range of networks, private and public, of all sizes, across most industries. She is author or co-author of numerous Cisco Press networking, voice, and design books. Denise lives in Maryland with her wonderfully patient and supportive husband, Kevin, and their two much less patient dogs.

About the Technical Reviewer

Sean Wilkins is an accomplished networking consultant for SR-W Consulting and has been in the field of IT since the mid-1990s, working with companies such as Cisco, Lucent, Verizon, and AT&T, as well as several other private companies. Sean currently holds certifications with Cisco (CCNP/CCDP), Microsoft (MCSE), and CompTIA (A+ and Network+). He also has a Master of Science in information technology with a focus in network architecture and design, a Master of Science in organizational management, a Master's Certificate in network security, a Bachelor of Science in computer networking, and Associates of Applied Science in computer information systems. In addition to working as a consultant, Sean spends most of his time as a technical writer and editor for various companies; check out his work at his author website: www.infodispersion.com.

Contents at a Glance

How This Book Is Organized xiii

Chapter 1 Networking Overview 1

Chapter 2 IPv6 Overview 9

Chapter 3 RIP 22

Chapter 4 EIGRP 25

Chapter 5 OSPF 43

Chapter 6 Advanced Routing Techniques 65

Chapter 7 BGP and Internet Connectivity 86

Chapter 8 Infrastructure Security 105

Chapter 9 Infrastructure Services 114

Contents

How This Book Is Organized xiii

Chapter 1 Networking Overview 1

IP, TCP, and UDP Operations 1

IP Operations 1

TCP Operations 2

UDP Operations 2

Routing Fundamentals 3

The Routing Table 3

Administrative Distance 4

Choosing a Routing Protocol 5

Distance Vector Protocols 5

Link State Protocols 5

Path Vector Protocols 5

Packet Forwarding 6

Configuring and Troubleshooting CEF 7

Loop Prevention Mechanisms 8

Split Horizon 8

Route Poisoning 8

Chapter 2 IPv6 Overview 9

IPv6 Addressing 10

Simplifying an IPv6 Address 10

Special Addresses 11

IPv6 Host Addressing 12

Neighbor Discovery Protocol 12

Manual IP Address Assignment 12

Manual Network Assignment 13

Stateless Address Autoconfiguration 14

Securing NDP 15

DHCPv6 15

Renumbering 15

IPv6 Routing 15

Static Routing 16

IPv6 Route Summarization 16

	Integrating IPv4 and IPv6	17
	Tunneling IPv6 over IPv4	18
	<i>Manual Tunnels</i>	18
	<i>GRE Tunnels</i>	18
	<i>6to4 Tunnels</i>	19
	<i>ISATAP Tunnels</i>	19
	IPv6 Link Types	20
	<i>Point-to-Point Links</i>	20
	<i>Point-to-Multipoint Links</i>	20
	<i>Multiaccess Links</i>	21
Chapter 3	RIP	22
	RIP Version 2	22
	RIPv2 Configuration	23
	RIPng for IPv6	23
	RIPng Configuration	24
Chapter 4	EIGRP	25
	EIGRP Overview	25
	EIGRP Neighbor Establishment	26
	Packet Types	26
	Neighbor Discovery and Route Exchange	26
	<i>Unicast Neighbors</i>	27
	<i>Troubleshooting and Verifying EIGRP Neighbors</i>	27
	EIGRP Route Selection	27
	EIGRP Metric	27
	<i>Wide Metrics</i>	28
	Diffusing Update Algorithm	29
	EIGRP for IPv4	30
	Basic Configuration	30
	Optimizing the EIGRP Configuration	31
	<i>Passive Interface</i>	31
	<i>Advertising a Default Route</i>	31
	<i>Summarization</i>	31
	<i>Load Balancing</i>	32
	<i>EIGRP Authentication</i>	33
	EIGRP Stub Routing	34
	EIGRP for IPv6	35
	Verifying and Troubleshooting EIGRP	36

EIGRP Named Mode 37

EIGRP Over WAN Links 38

Layer 3 MPLS WAN 38

Layer 2 WAN 39

Frame Relay 39

EIGRP Over the Top 42

Chapter 5 OSPF 43

OSPF Overview 43

OSPF Network Structure 43

OSPF Metric 45

Link-State Advertisements 45

LSA Operation 46

LSA Types 46

OSPF Operation 47

OSPF Packets 47

OSPF Neighbor Relationships 48

Basic OSPFv2 Configuration 49

Router ID 50

Verifying and Troubleshooting OSPF 50

OSPF Network Types 51

Designated Routers 52

Nonbroadcast Multiaccess Networks 53

OSPF over Layer 2 and Layer 3 MPLS 53

Advanced OSPF Configuration 54

OSPF Summarization 54

Passive Interface 55

OSPF Default Routes 55

Stub and Not-So-Stubby Areas 56

Virtual Links 56

OSPF Authentication 57

OSPF for IPv6 59

OSPFv3 LSAs 60

Traditional OSPFv3 Configuration 61

New OSPFv3 Configuration 62

Chapter 6	Advanced Routing Techniques 65
	Controlling Routing Updates 65
	Route Maps 66
	<i>Route Map Syntax</i> 66
	<i>Route Map Match and Set Conditions</i> 66
	Controlling Route Redistribution Using Route Maps 67
	Tagging Routes Using a Route Map 68
	Prefix Lists 69
	Distribute Lists 70
	Passive Interfaces 71
	Using Multiple Routing Protocols 71
	Configuring Route Redistribution 71
	<i>Seed Metric</i> 72
	<i>Administrative Distance</i> 73
	Planning Route Redistribution 74
	Redistribution Techniques 75
	Path Control 76
	Policy-Based Routing 77
	Using IOS IP SLA 79
	VRF-Lite 82
Chapter 7	BGP and Internet Connectivity 86
	Planning an Internet Connection 86
	Types of ISP Connections 86
	IP Addressing and AS Numbering 87
	BGP Route Options 87
	BGP Overview 88
	BGP Databases 89
	BGP Message Types 89
	BGP Next-Hop Selection 90
	BGP Next Hop on a Multiaccess Network 91
	BGP Synchronization Rule 91
	Configuring Basic BGP 91
	BGP Network Command 92

	BGP Peering	93
	<i>BGP Peering States</i>	93
	<i>Troubleshooting BGP</i>	94
	BGP Path Selection	94
	BGP Attributes	94
	BGP Path Selection Criteria	95
	Influencing BGP Path Selection	96
	Filtering BGP Routes	97
	Prefix Lists	97
	AS Path Access List	98
	Order of Operations	99
	BGP Authentication	99
	Verifying BGP	100
	Multiprotocol BGP	103
Chapter 8	Infrastructure Security	105
	Device Access Control	105
	Router Security Features	106
	Access Control Lists	106
	<i>Configuring IPv4 ACLs</i>	107
	<i>Configuring an IPv6 Access List</i>	108
	Unicast Reverse Path Forwarding	108
	Tunneling Technologies	109
	GRE Tunnels	110
	<i>Configuring a GRE Tunnel</i>	110
	DMVPN	111
	Easy VPN	111
	PPPoE	112
Chapter 9	Infrastructure Services	114
	Simple Network Management Protocol	114
	SNMPv2c Configuration	115
	SNMPv3 Configuration	115
	Logging	116
	Debug	117
	Syslog	117
	<i>Configuring Syslog</i>	118

Network Time Protocol 119**DHCP 120**

DHCP for IPv4 120

*Configuring DHCP for IPv4 120**IPv4 DHCP Relay Agent 121*

DHCP for IPv6 121

*DHCPv6 Process 122**Configuring Stateful DHCPv6 122**Configuring Stateless DHCPv6 123**IPv6 DHCP Relay Agent 123***NetFlow 124****Network Address Translation 125**

Configuring Traditional NAT for IPv4 126

*Static NAT 126**Dynamic NAT 127**PAT 127*

Configuring NAT Virtual Interface 128

NAT64 129

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).
- *Italic* indicates arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets ([]) indicate an optional element.
- Braces ({ }) indicate a required choice.
- Braces within brackets ([{ }]) indicate a required choice within an optional element.

How This Book Is Organized

- **Chapter 1, “Networking Overview”:** This chapter provides a review of basic IP, TCP, and UDP operations, such as fragmentation and windowing. It also reviews routing fundamentals including AD, types of protocols, packet forwarding, and loop prevention.
- **Chapter 2, “IPv6 Overview”:** This chapter provides an overview of IPv6 addressing, routing, and route summarization. This rather in-depth introduction to IPv6 covers the IPv6 address format, ways for hosts to acquire their addresses, and IPv6 routing. It also includes strategies for integrating IPv4 and IPv6, such as in various types of tunnels. More detailed applications of IPv6 are covered in the chapters for each routing protocol and for the various technologies.
- **Chapter 3, “RIP”:** Configuring and verifying RIPv2 and RIPv6 are described in this chapter.
- **Chapter 4, “EIGRP”:** This chapter contains an in-depth description of EIGRP for IPv4 and IPv6 operation and configuration, including neighbor establishment and route exchange. It covers using EIGRP with Frame Relay, Ethernet over MPLS (EoMPLS), and Layer 3 MPLS VPNs. It also includes EIGRP named mode, ways to optimize EIGRP, and securing EIGRP through authentication.
- **Chapter 5, “OSPF”:** Chapter 5 describes OSPF’s structure and operation. It covers OSPF design requirements, neighbor establishment, and LSA information for both OSPFv2 and OSPFv3. The configuration portion provides OSPF configuration for LAN and WAN networks. The chapter additionally covers optimizing and securing OSPF.
- **Chapter 6, “Advanced Routing Techniques”:** This chapter examines various methods of controlling routing updates, such as route maps, prefix lists, and distribute lists. It describes how to configure route maps and how to use them for policy-based routing, controlling route redistribution, and tagging routes. Additionally, techniques such as IP SLA and VRF Lite are covered.
- **Chapter 7, “BGP and Internet Connectivity”:** This chapter gives an overview of BGP operation and basic configuration. BGP path selection is covered, along with ways to influence the path selection and filter routes. Additionally, methods to verify BGP operation are shown. Multi-protocol BGP, using BGP with IPv6 routing, is covered.

- **Chapter 8, “Infrastructure Security”:** This chapter examines ways to secure the routing infrastructure and the routers themselves, as well as the data transmitted. It looks at IPv4 and IPv6 ACLs, device access control, and various types of traffic tunneling techniques.
- **Chapter 9, “Infrastructure Services”:** This chapter describes useful network management services, such as SNMP, logging, debugging, and NetFlow. It covers DHCP for both IPv4 and IPv6, NAT for both IPv4 and IPv6, and NAT virtual interface.

CHAPTER 1

Networking Overview

IP, TCP, and UDP Operations

This book assumes that the reader is already familiar with basic IP, TCP, and UDP operations; their header structures; and IPv4 addressing. This section briefly discusses some additional points about these protocols.

IP Operations

Packet fragmentation is handled differently in IPv4 and IPv6. Each link has a maximum transmission unit (MTU) size associated with it. If the IPv4 packet is larger than the interface MTU, it is fragmented (unless the Don't Fragment bit is set in the IP header, in which case it will be dropped). Fragments of the same original packets are marked identically in the Identifier field of the IP header, so that the receiving router knows they belong together. The Fragment Offset field gives that fragment's place in the original packet, so that it can be reassembled correctly.

IPv6 uses Path MTU Discovery to determine the minimum MTU along the path and so does not fragment packets by default. When a too-large packet is sent, ICMPv6 returns a packet-too-big message. TCP reduces its packet size accordingly, but UDP cannot do that so the source host must fragment the packet. If fragmentation is required, the fields needed to reassemble the packet are placed in an Extension header between the IPv6 header and the TCP/UDP header. IPv4 can also use Path MTU Discovery to ensure that all packets fit the minimum path size and thus will not be fragmented.

The Internet Control Message Protocol (ICMP) has many types of messages and numerous uses in an IP network. Its "TTL Exceeded" message can be used along with the Time to Live (TTL) field in an IP header for performing trace routes. Its "Redirect" message notifies a host to use an alternate default gateway, saving an unneeded hop. Probably ICMP's most common uses are the "Echo Request" and "Echo Reply" messages used, along with the "Destination Unreachable" messages, by PING.

TCP Operations

TCP, as a connection-oriented protocol, uses fields in its header to negotiate various settings with its peers. Window size is one of these settings. Rather than sending one segment at a time and waiting for an acknowledgement, TCP groups multiple segments and sends them. The size of this group is called its *window*. The Window Size field in the TCP header tells how many bytes of data the host can accept at a time, in one window. The sending router adjusts the amount of data it sends to fit its peer's window size.

The default maximum window size is 65,535 bytes. You might want to increase that size to get greater throughput from TCP applications. For instance, on any given link the greatest throughput you can get is expressed by the window size divided by link latency. Increasing window size is generally much easier than decreasing link latency! A larger window size is especially useful on links with a high Bandwidth Delay Product (BDP) to be able to send a large amount of data before having to wait for an acknowledgement. Bandwidth Delay Product is equal to the link bandwidth multiplied by end-to-end delay; slower links have higher BDPs.

The TCP header also contains an Options field, and a most commonly used option is maximum segment size (MSS). This option helps avoid fragmentation at the IP layer. Hosts check the link MTU, subtract 40 bytes to account for the IP and TCP headers, make sure that value does not overrun the interface buffer, and then send that value to their peer as the MSS value. The peer must then set its “Send MSS” to that value.

You should be aware of one other issue with TCP—that of global synchronization. When a TCP segment is dropped, it reduces its send rate and then gradually ramps back up. If a sudden event caused all the TCP traffic on an interface to be dropped, then every flow would back off at the same time, and ramp back up at the same time. At some point, the interface would become congested and the flows would all drop again, repeating the cycle. Thus every flow's sending pattern becomes synchronized, which results in inefficient link utilization. Some QoS tools such as Weighted Random Early Detection (WRED) can help prevent global synchronization.

UDP Operations

UDP does not behave in the same way as TCP when its traffic is dropped. It continues sending at the same rate because, being connectionless, it does not know whether or not anything was dropped. On an interface with both TCP and UDP flows, UDP uses the bandwidth whereas TCP backs off and then ramps back up. In fact, TCP's operation might allow even more UDP flows, which can result in *TCP starvation*. Output queues fill with UDP traffic, and the TCP traffic continues getting dropped.

Latency is another concern with UDP traffic. Most applications that use UDP send small packets of interactive traffic such as voice. The quality of the application performance depends on few packet drops and low, predictable delay. However, large packets such as an FTP file transfer could hold up the transmission of a small UDP traffic, causing higher latency. Variable latency, or *jitter*, is a common problem with UDP traffic and one that can be addressed with QoS.

Routing Fundamentals

Routers must know how to reach remote networks to forward packets toward them. They can learn this either by manual configuration (*static route*) or by using a routing protocol to exchange information about network reachability. To configure a static route, use the global command:

```
(IPv4) ip route [destination_network] [subnet_mask] [next_hop_IP]
(IPv6) ipv6 route [destination_network]/prefix-length [next_hop_IP]
```

Another special type of route is a *default route*, designated in IPv4 by 0.0.0.0 with a subnet mask of 0.0.0.0 and in IPv6 by ::/0. Default routes are used when no other route exists to the destination network.

Cisco routers support multiple routing protocols, but the ROUTE exam covers only Routing Information Protocol (RIP), Enhanced Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First (OSPF), and Border Gateway Protocol (BGP). This section gives an overview of routing protocols and some key differences between them.

The Routing Table

Routers maintain a database of the routes they will use to forward a packet. To build this routing table, the router rules out invalid routes, and then considers the remaining options. The procedure is defined in the following steps:

1. For each route received, verify the next hop. If invalid, discard the route.
2. If multiple identical, valid routes are received by a routing protocol, choose the lowest metric. Routes are identical if they advertise the same prefix and mask, so 192.168.0.0/16 and 192.168.0.0/24 are separate paths and may each be placed into the routing table.
3. If the same valid route is advertised by different routing protocols, choose the path with the lowest AD.

After the routing table is built, routers choose which route to use for a specific packet by looking for a match to the destination IP address. Rarely will a route match the destination IP address exactly, so the router looks for the longest match. For instance, suppose a packet is bound for the IP address 10.1.1.1. The routing table has a route for 10.1.0.0/16, one for 10.1.1.0/24, and a default route of 0.0.0.0. The default route matches 0 bits of the destination address, the 10.1.0.0 route matches 16 bits of the destination address, and the 10.1.1.0 route matches 24 bits of the destination address. The 10.1.1.0 route is the longest match, so it will be used to forward the packet.

To view the routes in the routing table, and verify any static or default routes, use the command **show ip route**.

Administrative Distance

Cisco routers are capable of supporting several IP routing protocols concurrently. When identical prefixes are learned from two or more separate sources, Administrative Distance (AD) is used to discriminate between the paths. Cisco routers assign an AD value to each routing protocol; in some cases different AD values are assigned to routes learned natively by the protocol versus those redistributed into the protocol. All other things being equal, routers choose paths advertised by the protocol with the lowest AD. AD can be manually adjusted.

Table 1-1 lists the default values for various routing protocols.

Table 1-1 Routing Protocols and Their Default Administrative Distance

Information Source	AD
Connected	0
Static	1
External BGP	20
Internal EIGRP	90
OSPF	110
IS-IS	115
RIP	120
External EIGRP	170
Internal BGP	200
Unknown	255

Choosing a Routing Protocol

A group of network devices (routers, switches, and so on) under one common administration is termed an *autonomous system*, or AS. This is typically a company or organization, or perhaps one division of an organization. Routing protocols that are used inside an AS are called *Interior Gateway Protocols*, or IGP, and those used to exchange routes between AS's are termed *Exterior Gateway Protocols*, or EGP. IGP's need to respond quickly to changes, converging rapidly. They should be fairly easy to figure and optimize. They might need to scale to manage very large networks, but not as large as EGP's can handle. An EGP can operate in very large networks and are used on the Internet. The only EGP in use today is Border Gateway Protocol (BGP). An AS might want to tightly control what routes it advertises outside its borders, what routes it accepts, and the metrics of those routes. EGP's do not need to converge as quickly as IGP's but they must be very stable, extremely scalable, and support customizable policies.

Classify routing protocols based on the method they use to calculate the best paths. Three types of protocols are discussed in this section.

Distance Vector Protocols

Distance vector protocols have only two pieces of information about a network: the distance (or metric) to reach it, and the next-hop neighbor on the path (or vector) to that network. RIP and EIGRP are distance vector protocols. "Distance" for RIP is hop count, whereas for EIGRP it is a more complex metric. Distance vector protocols in general can be slow to converge because they have to query their neighbors for an alternate route when there is a topology change, but EIGRP compensates for this by proactively selecting an alternate route when possible.

Link State Protocols

Link state protocols pass along information about individual links connected to the routers in their portion of the network. Each link-state router then builds a map of the logical topology and chooses the best path using the Shortest Path First (SPF) algorithm. In a topology change, link state routers do not need to query their neighbors; they can choose an alternate path from their topology database. OSPF and IS-IS are link state protocols.

Path Vector Protocols

Path vector protocols are similar to distance vector protocols in that they do not maintain a complete picture of the network, only the next-hop router address. However, they do exchange information about the path to the

destination network as part of their updates, using it as part of the metric and as a loop prevention mechanism. An example of this is the AS-path part of BGP updates. BGP is the only path vector protocol in common use today.

Table 1-2 compares routing protocols.

Table 1-2 Comparison of Routing Protocols

Property	RIP v2	EIGRP	OSPF	BGP
Method	Distance vector	Distance vector	Link state	Path vector
Summarization	Auto/Manual	Auto/Manual	Manual	Auto/Manual
Metric	Hop count	Bandwidth + sum of path delays	Sum of link costs	Complex metric
Timers: Update (hello/dead)	Every 30 sec and Triggered	Triggered (LAN 5/15, WAN 60/180)	Triggered, but LSA refreshes every 30 minutes (NBMA 30/120, LAN 10/40)	Triggered (60/180)
Network Size	Small	Large	Large	Very large

Packet Forwarding

Routing a packet essentially involves two functions: deciding the best path for the packet, and then forwarding it out the exit interface. The exchange of routing information and building of the route table are handled by the control plane. Forwarding the packet and placing the correct Layer 2 header on is handled by the data plane.

Three methods of forwarding packets are process switching, fast switching, and Cisco Express Forwarding.

Process switching is performed completely in software, so it is the most CPU intensive and thus the slowest method. For each frame, the router strips off the incoming Layer 2 header, sends it to the CPU for the Layer 3 next hop, outgoing interface, and Layer 2 header.

Fast switching is accomplished partially in hardware, so it is faster than process switching. The router builds a fast-switching cache. When a frame comes in, the destination is checked against this cache. If an entry is present, then the Layer 2 header is added and the frame is sent to the outgoing interface. If no entry is present, the packet is sent to the router and processed

as with process switching. An entry is added to the fast-switching cache and subsequent packets are fast switched.

Cisco Express Forwarding (CEF) is the default switching mode for modern platforms. It is also the fastest mode. The router proactively determines the outgoing interface, next-hop adjacency, and Layer 2 information for every route in the routing table. Packets are then forwarded completely in hardware, without the need for CPU intervention. CEF caches routing information in the forwarding information base (FIB) and next-hop Layer 2 information in an adjacency table.

Not all types of traffic can be handled by CEF. Some forms that are *punted* (sent to the processor for handling) are

- Packets with IP header options
- Tunneled traffic
- Unsupported encapsulation types
- Packets with an expiring TTL
- Packets that exceed the MTU and must be fragmented

Configuring and Troubleshooting CEF

By default, CEF is enabled for IPv4 and disabled for IPv6. Enabling IPv6 routing automatically enables CEF for IPv6. The default load sharing type is per destination. To disable CEF on a per-interface basis, use the interface-level command **no ip route-cache cef**. To disable CEF for the entire router, use the global command **no ip cef**.

Verify that CEF is enabled or disabled on an interface with the following command:

```
# show ip interface [interface]
```

You should see output that says either “IP CEF switching is enabled” or “IP CEF switching is disabled.”

View the FIB with the following command:

```
# show [ip | ipv6] cef {interface} {detail}
```

View or troubleshoot CEF adjacencies with the following command:

```
# show adjacency
```

Loop Prevention Mechanisms

Incorrect routing information can result in traffic looping throughout the network, being sent back and forth between the same routers. The traffic might be dropped or blackholed. Looping is especially prevalent in distance vector protocols. Route advertisements can cause this problem in two ways:

- A router can hear about a network from its neighbor, and then advertise that network back to the same neighbor.
- A router can keep sending traffic to a network that is down.

Two techniques to prevent this type of looping are split horizon and route poisoning.

Split Horizon

Enabling *split horizon* causes the router to never advertise a route out of the interface where it learned that route. Some protocol implementations add a poison reverse to that, by advertising the route back to its neighbor with a metric of “unreachable,” thus ensuring the neighbor does not consider it a viable path to the network. Split horizon is useful on point-to-point links but can cause problems on multipoint links such as Frame Relay. It is enabled by default, so it must be manually disabled on a multipoint link.

Route Poisoning

To speed up convergence and prevent traffic from being blackholed, distance vector protocols implement route poisoning. As soon as a router learns that a network is down, it advertises that network out all of its interfaces, including the one where it received the route advertisement, with an “unreachable” metric. The logic behind this breaking of the split horizon rule is a multi-access network (such as Ethernet) with several routers on it. If one router did not receive the initial update for some reason, this provides another chance for it to be processed.

CHAPTER 2

IPv6 Overview

This chapter assumes the reader is very familiar with IPv4 addressing and operation. IPv6 is an extension of IP addressing with several advanced features:

- Larger address space.
- Simpler header for increased router efficiency.
- No more broadcasts.
- Stateless autoconfiguration.
- Built-in support for Mobile IP.
- Built-in support for IPsec security.
- Rich transition features.
- Easy IP address renumbering.
- Support for multiple addresses per interface.
- Routers create link-local addresses for use by Interior Gateway Protocols (IGPs).
- As with IPv4, the addresses can be obtained from an Internet service provider (ISP) or can be provider independent.

The primary adoption of IPv6 is driven by the need for more addresses. Given the growth in Internet use and the emergence of large groups of Internet users worldwide, this is a significant requirement. Another reason to use IPv6 is growth in the size of the current Internet routing table. IPv4 addresses are not summarized enough to keep the size down, increasing the load on Internet routers. Additionally, although the use of Network Address Translation (NAT) has postponed the need for IPv6, it breaks TCP/IP's end-to-end networking model.

IPv6 is not enabled by default on Cisco routers. To enable IPv6 routing, the command is **ipv6 unicast-routing** at the global configuration mode.

IPv6 Addressing

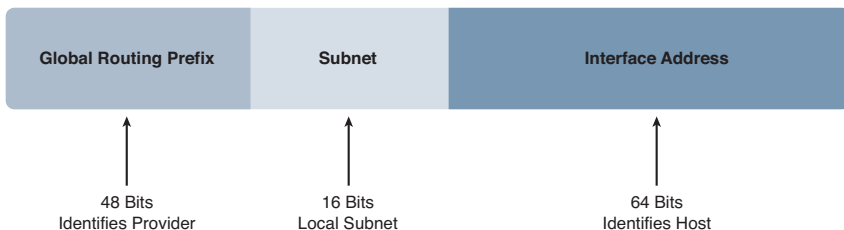
IPv4 addresses are 32 bits long and written in dotted decimal, whereas IPv6 addresses are 128 bits and written in hexadecimal. IPv6 addresses are written in groups of four numbers, separated by colons. Each number represents 4 bits and thus has a value of 0–F. Addresses are typically divided into a 64-bit network portion and a 64-bit host portion. An IPv6 address might look something like 2001:db80:aabb:1111:2222:3333:4444:ffff/64. There is no equivalent of the IPv4 dotted decimal subnet mask—the prefix length (network portion of the address) is written in CIDR notation. Having host addresses that end in 0 or F is legal.

The first 48 bits of the network portion are considered *global address space*. These bits consist of the following elements:

- The first three bits (/3) of a unicast address are always 001.
- The next 13 bits (/16) identify the top-level aggregator (TLA), the upstream ISP or address authority.
- The subsequent 24 bits (/40) identify the next-level aggregator, a regional ISP or address authority.

The next 16 bits are available for creating subnetworks within the enterprise. The host interface portion of the address is the last 64 bits. Because IPv6 does not use broadcasts and separates the interface address portion from the network portion, interface addresses of all zeroes and all ones are legal. Figure 2-1 shows the address components.

Figure 2-1 IPv6 Address Structure



Simplifying an IPv6 Address

There are two ways to shorten the representation of an IPv6 address. Take the example address 2001:0000:0001:0002:0000:0000:0000:ABCD.

- Leading zeros can be omitted, which reduces the preceding address to 2001:0:1:2:0:0:0:ABCD.
- Sequential zeros can be shown as double colons. *This is allowed only once per address.* This simplifies the aforementioned address even further, to 2001:0:1:2::ABCD.

Be sure that you can distinguish between correct and incorrect IPv6 addresses. For instance, the address 2001::1:2::ABCD is incorrect because it uses double colons twice.

Special Addresses

IPv6 does not support broadcasts but replaces them with multicasts. IPv6 also uses *anycast*, which involves using the same address on multiple devices. Anycast is used to implement redundancy and has been backported to IPv4.

Each IPv6 system must recognize the following addresses:

- Its unicast addresses.
- Link local address (begins with FE80/10).
- Loopback (::1/128).
- All-nodes multicast (FF00::1).
- Site-local multicast (FF02::2).
- Solicited-nodes multicast (FF02::1:FF00/104).
- Default route (::/0).
- Routers must support subnet-router anycast (all zeros EUI-64).
- Routers must support local all-routers multicast (FF01::2), link-local (FF02::2), and site-local (FF05:2).
- Routers must support routing protocol multicast groups.

Additionally, some systems also use the following addresses:

- IPv4-compatible address (::/96 | 32-bit, IPv4 address)
- Second unicast address shared with another system (anycast)
- Additional multicast groups

IPv6 Host Addressing

IPv6 uses one special type of address called a *link-local address*. In IPv6, this is composed of the network prefix FE80:: and the host or interface MAC address. This address is valid only on the local network segment. To make sure that the address is not already being used, a host sends a neighbor solicitation message to that address to make sure no other device answers. Even if you assign a routable, global address to the interface, it still has its link-local address. IGP routing protocols use the link-local address to form neighbor relationships. The link-local address is also the next-hop address that is installed in the routing table by IGPs. You can see an example of a link-local address later in Example 2-2.

An IPv6 host can obtain a routable IP address by manual assignment; physically designating the network part of the address only using Stateless Address Autoconfiguration (SLAAC) or by using DHCPv6. DHCPv6 is covered in Chapter 9, “Infrastructure Services.”

To ping any IPv6 address, including link-local addresses, use the command **ping [ipv6] destination-address [source exit-interface]**. Note that the source interface is required when pinging a link-local address.

Neighbor Discovery Protocol

Neighbor Discovery Protocol uses ICMPv6 and assists in IPv6 addressing in several ways:

- **Duplicate address discovery (DAD):** The host uses neighbor solicitation (NS) to send a message to its own address. No response means that the address is unique. This is used when a host creates its link-local address.
- **Neighbor discovery:** Similarly to ARP, the host discovers the link-local address of neighbors using an NS message. This is ICMP type 135. Neighbors respond with an ICMP type 136 message.
- **Router discovery:** IPv6 routers periodically send router advertisements (RAs) listing the network prefix. When a host comes online, it immediately sends a router solicitation (RS) message asking for prefix information rather than waiting for the RA. This is sent to the all-routers multicast address.

Manual IP Address Assignment

To manually assign an IPv6 address to a router interface, use the command **ipv6 address ipv6-address/prefix-length**. Example 2-1 shows two router

interfaces configured with IPv6 addresses. In the first address, note that leading zeroes are omitted in two of the quartets. In the second address, note the use of the double colons.

Example 2-1 IPv6 Address Configuration

```
Router(config)# ipv6 unicast-routing
!
Router(config)# interface gigabitethernet0/0
Router(config-if)# ipv6 address 2001:db8:aabb:1:2222:3333:4444:f
fff/64
!
Router(config)# interface gigabitethernet0/1
Router(config-if)# ipv6 address 2001:db8:aabb:2::1 /64
```

Manual Network Assignment

Although statically assigning IP addresses to router interfaces is a best practice, routers can create their own IPv6 address when they know the network prefix. Assuming the end system has a 48-bit MAC address, the router or host flips the global/local bit (the seventh bit) and inserts 0xFFEE into the middle of the MAC address. The resulting 64-bit number is called the *EUI-64 address*. The prefix and EUI-64 address are concatenated to form the host IPv6 address. The command is **ipv6 address *ipv6-prefix::/prefix-length* *eui-64***.

Example 2-2 shows this command and the resulting link-local and global unicast address. Note the interface MAC address and how it relates to the IPv6 addresses.

Example 2-2 Configuring an EUI-64 Address

```
Router(config)# interface ethernet0/0
Router(config-if)# ipv6 address 2001:db8:1234:aabb::/64 eui-64
!
Router# show int e0/0
Ethernet0/0 is up, line protocol is up
  Hardware is AMDP2, address is 001d.a188.33c1 (bia 001d.a188.33c1)
<output omitted>
!
Router# show ipv6 int e0/0
Ethernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::21D:A1FF:FE88:33C1
  No Virtual link-local address(es):
```

Example 2-2 Continued

```
Global unicast address(es):
  2001:DB8:1234:AABB:21D:A1FF:FE88:33C1, subnet is
  2001:DB8:1234:AABB::/64 [EUI]
Joined group address(es):
  FF02::1
  FF02::2
  FF02::1:FF00:1A00
<output omitted>
```

Stateless Address Autoconfiguration

One big benefit of IPv6 is the capability of a host to automatically acquire an IP address without DHCP, called *Stateless Address Autoconfiguration* or *SLAAC*. To enable stateless autoconfiguration, use the interface command **ipv6 address autoconfig**. Acquiring an address involves the following steps:

- Step 1.** The host creates a link-local address.
- Step 2.** It sends an NS message to its link-local address out the interface.
- Step 3.** If there is no reply, DAD declares the address unique.
- Step 4.** If the host doesn't receive an RA, it sends an RS.
- Step 5.** A router on the subnet sends an RA, listing its interface prefix.
- Step 6.** The host uses that prefix and the interface MAC address to create its IPv6 address.

Use the command **show ipv6 interface** to verify your configuration. Example 2-3 shows this command and the resulting IPv6 address.

Example 2-3 IPv6 Autoconfiguration

```
Router(config)# int e0/0
Router(config-if)# ipv6 address autoconfig
!
Router# show ipv6 int e0/0
Ethernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::21D:A1FF:FE6C:D238
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8::21D:A1FF:FE6C:D238, subnet is 2001:DB8::/64 [EUI/CAL/
    PRE]
    valid lifetime 2591828 preferred lifetime 604628
<output omitted>
```

Securing NDP

NDP is an essential part of IPv6, enabling SLAAC and allowing MAC addresses to be learned and hosts to be redirected to better routes. But because it starts operating when IPv6 unicast routing is enabled, it can expose the router to problems such as man-in-the-middle attacks and black-holing of traffic.

If there is no need for router advertisement messages on a link, disable them with the command **ipv6 nd suppress-ra**.

Another option is to use Secure Neighbor Discovery (SeND). SeND is an extension to NDP that provides some additional security, such as

- Proof of address ownership to prevent address hijacking
- Message integrity and replay protection
- Router authorization for specific prefixes only

SeND requires a pair of public and private keys for all IPv6 nodes. Its deployment can become complex and its use can be CPU intensive.

DHCPv6

DHCP can also be used to assign IP addresses on an IPv6 network. DHCP for IPv6 is covered in Part I in the “DHCP” section of Chapter 9.

Renumbering

IPv6 supports easy network renumbering. Note in Example 2-3 that lifetimes are listed for the subnet address. When it is time to change the subnet, simply configure the router to advertise the old prefix with a short lifetime and the new prefix with a longer one. You can even configure the router to expire a prefix at a certain date and time. The router sends out an RA with both prefixes and their lifetimes. Hosts then update their addresses. Anyone who has had to renumber a large range of IPv4 addresses can testify to how useful this feature is!

IPv6 Routing

Routing with IPv6 will seem familiar to you. The same IGPs—RIP, EIGRP, and OSPF—are used as in IPv4; they have been adapted to carry IPv6 routes. BGP extensions enable it to perform IPv6 routing. The same rules for metric and administrative distance apply. The commands are similar, too. The main difference in commands is that you must specify that the command

pertains to IPv6 because IPv4 is the default. One big configuration difference is that IGP's no longer use the **network** command to initiate routing; it is enabled at each interface instead. BGP still uses the network command to designate which networks to advertise.

The specific commands and examples are given in the chapters for each routing protocol. The following is some general information that applies to all protocols.

Static Routing

Static routing with IPv6 works exactly like it does with version 4. Aside from understanding the address format, there are no differences. The syntax for the IPv6 static route command is

```
Router(config)# ipv6 route ipv6-prefix/prefix-length {ipv6-  
address | interface-type  
interface-number [ipv6-address]} [administrative-distance]  
[administrative-multicast-distance | unicast | multicast] [tag  
tag]
```

The following configuration shows the command in context as it might be applied. The first line shows a static route that lists a next-hop address. The second line shows a directly connected static default route that lists an outbound interface. The third line shows a fully specified static route, which lists both the next-hop address and the outbound interface:

```
Router(config)# ipv6 route 2001:db8:1:2::/64 2001:db8:1:1::1  
!  
Router(config)# ipv6 route ::/0 serial1/0/0  
!  
Router(config)# ipv6 route 2001:db8:1:2::/64 serial1/0/1  
2001:db8:1:1::1
```

Verify your configuration with the **show ipv6 route** command.

IPv6 Route Summarization

You must summarize IPv6 routes for the same reasons as you summarize IPv4 routes. There might even be a greater need because you can have so many more IPv6 routes! Originally, the idea was to use the 16-bit local subnet portion of the address for local routes, but you can use part of the 64 bits allocated for host addresses for subnetting, too. As in IPv4, this just gives you fewer possible hosts. After you understand the concept, you can apply it to any portion of the IPv6 address you need.

Recall that each number in an IPv6 address is 4 bits, which gives you 16 possible values: 0–F. Each of the quartets separated by colons is thus 16 bits. For instance, consider the following set of networks. How would you summarize them?

- 2001:db8:1:0::/64
- 2001:db8:1:1::/64
- 2001:db8:1:2::/64
- 2001:db8:1:3::/64

The fourth quartet is the one we are interested in. The binary equivalents for the fourth quartet of the preceding networks are

- 0000 0000 0000 0000
- 0000 0000 0000 0001
- 0000 0000 0000 0010
- 0000 0000 0000 0011

You can see that the last two bits are the ones you can summarize. It doesn't matter whether their value is 0 or 1. You need only to focus on the value of the first 14 bits in this quartet. The bit mask for the summary breaks down this way:

2001: db8: 1: 0::

16 bits + 16 bits + 16 bits + 14 bits = 62 bits

Thus, the summarized address is 2001:db8:1:0::/62.

Integrating IPv4 and IPv6

Several strategies exist for migrating from IPv4 to IPv6. Each strategy must be considered when organizations decide to make the move to IPv6 because each has positive points to aiding a smooth migration. It should also be said that there does not have to be a global decision on strategy—your organization might choose to run dual-stack in the United States, convert completely to IPv6 in Japan, and use tunneling in Europe. The transition mechanisms include

- **Dual stack:** Running IPv6 and IPv4 concurrently on the same interface.

- **Tunneling:** Routers that straddle the IPv4 and IPv6 worlds encapsulate IPv6 traffic inside IPv4 packets.
- **Translation:** Using an extension of NAT, NAT64, to translate between IPv4 and IPv6 addresses. This is covered in Chapter 9.

Tunneling IPv6 over IPv4

A tunnel serves as a virtual point-to-point link between IPv6 domains. It doesn't matter what the underlying IPv4 structure is if there is IP reachability between the tunnel endpoints.

There are several ways to tunnel IPv6 over IPv4:

- Manual tunnels
- GRE tunnels
- 6to4 tunnels
- Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)

Manual Tunnels

When you manually create the tunnel, the source and destination IP addresses are IPv4 addresses because IPv4 is the transport protocol. Use loopback addresses for increased stability. IPv6 addresses go on the tunnel interfaces because IPv6 is the passenger protocol. Because IPv6 considers the tunnel a point-to-point link, the address of each end of the tunnel is in the same subnet. Include the command **tunnel mode ipv6ip** in tunnel configuration mode to enable IPv6 over IP encapsulation.

To verify your configuration use the commands **debug tunnel** or **show interface tunnel *int-number***.

GRE Tunnels

GRE is the default tunnel mode for Cisco routers. It provides more flexibility because it is protocol-agnostic. It can carry multiple protocols, including IPv6 and routing protocols.

Configuring an IPv4 GRE tunnel to carry IPv6 traffic is the same as configuring a manual tunnel except you do not have to specify the tunnel mode because GRE is the default. You can allow a routing protocol on the tunnel interface, too. The process is the same as enabling it on a physical interface.

To configure a completely IPv6 GRE tunnel, use IPv6 interface addresses as the tunnel source and destination. Give the tunnel endpoints IPv6 addresses, too. You need a command to identify that the transport protocol is IPv6. That command, given in tunnel configuration mode, is **tunnel mode gre ipv6**.

6to4 Tunnels

This technique dynamically creates tunnels that IPv6 considers point-to-multipoint interfaces. Use the reserved prefix 2002::/16 in your IPv6 domain, and then add the IPv4 address of the dual-stack router on the other side of the IPv4 domain as the next 32 bits of the network address. This means you must translate that IP address into hexadecimal.

When IPv6 traffic arrives at an edge dual-stack router with a destination IPv6 prefix of 2002::/16, the router examines the first 48 bits, derives the embedded IPv4 address from them, and uses it to determine the packet destination. The router then encapsulates the IPv6 packet in an IPv4 packet with the extracted IPv4 address as the packet destination.

Configure a tunnel as before, using IPv4 addresses as the source, but do not manually specify a destination. Give the tunnel an IPv6 address as previously described, with the tunnel destination embedded in its prefix. The tunnel mode command is **tunnel mode ipv6ip 6to4**.

Each router needs a route to its peer on the other side of the IPv4 network. The only current options for this are static routes and BGP.

ISATAP Tunnels

ISATAP tunnels are similar to the other two tunnel techniques in that an IPv4 address is encoded into the IPv6 address. It is meant to be used within a site, between hosts and routers, although it can be used between sites.

The tunnel source address is an IPv4 address. Do not specify a tunnel destination. The IPv6 address of the tunnel itself combines the network prefix, 0000:5EFE, and the 32-bit IPv4 tunnel source address. The IPv4 address is encoded into the least significant 32 bits of the address. Use any network prefix. The tunnel interface link-local address still starts with FE80 and then uses 0000:5EFE plus the encoded IPv4 address.

For instance, the link-local address of a tunnel that uses 10.8.8.8 as its source is

```
FE80::5EFE:A08:808
```

The unicast IPv6 address of that same tunnel interface, assuming that prefix 2001:db8:1:3/64 was assigned to the interface, is

2001:db8:1:3:0:5EFE:A08:808

ISATAP tunnels do not support multicast. A route is needed to the tunnel destination if it is in a different subnet; this can be either a static route or a BGP route.

IPv6 Link Types

IPv6 recognizes three types of links:

- Point-to-point
- Point-to-multipoint
- Multiaccess

Point-to-Point Links

Recall that an IPv6 interface uses its MAC address to create its link-local address. A serial link has no MAC address associated with it, so it uses one from an Ethernet interface. You can manually configure the link-local address to make it more recognizable. Be sure to begin the IPv6 address with the link-local prefix FE80.

Point-to-point links do not necessarily need global unicast addresses. The routers can communicate with only link-local addresses, but you could not reach those interfaces from off the network because the link-local is not a routable address.

Point-to-Multipoint Links

For point-to-multipoint links, such as Frame Relay, you must map the destination IPv6 address to the correct DLCI, just as with IPv4. The difference is that with IPv6 you must also map the link-local address to the DLCI because it is used as the next hop for routing. So for each DLCI, you must have at least two mappings: the remote router's IPv6 global unicast address and the remote router's link-local address. The map command is

```
frame relay map ipv6 destination-address out dlci dlci-number  
broadcast
```

In a hub-and-spoke topology, the hub must be configured for IPv6 unicast routing for the spokes to communicate with each other.

Multiaccess Links

Devices on multiaccess links, such as Ethernet, build a table mapping destination Layer 3 addresses to Layer 2 addresses, whether you use IPv4 or IPv6. IPv4 uses a separate protocol, ARP, to accomplish this. In IPv6, the process is built in to the IPv6 protocol and uses the Neighbor Discovery Protocol. An IPv6 device sends a neighbor solicitation (NS) multicast with a prefix of FE02. The neighbor responds with a neighbor advertisement (NA) message listing its MAC address. As with ARP, these mappings have a set lifetime called the *reachable time*, so an NS can also be sent periodically to verify that a neighbor is still reachable.

To add a static entry to the Neighbor Discovery table, use the command **ipv6 neighbor** *ipv6-address interface-type interface-number hardware-address*. A static address does not age out of the table.

Display the mappings with the **show ipv6 neighbors** command.

CHAPTER 3

RIP

RIP Version 2

Routing Information Protocol (RIP) has been in existence since 1988. It is a basic distance vector protocol that uses hop count as its metric, and thus does not pick up any differences in bandwidth between different routes. RIPv2 is a classless protocol—it carries subnet mask information in its updates, enabling you to use various subnet masks in the network. Some other characteristics of RIPv2 include the following:

- Uses UDP port 520.
- All routes advertised every 30 seconds, along with triggered updates due to topology change.
- Administrative distance is 120.
- Updates sent as multicasts to IPv4 address 224.0.0.9.
- Maximum metric (hop count) is 15. A hop count of 16 is considered infinity, poisoning the route.
- Supports plain text and MD5 authentication.
- No neighbor relationship formation process exists—all interfaces participating in RIP send route updates whether or not another RIP router is out of that interface.
- Route summarization is performed at each interface.
- Supports variable-length subnet masks but does auto-summary by default.
- Load balances across up to four equal metric paths by default.

RIP implements *split horizon* to help prevent routing loops. This does not allow a router to advertise out an interface a route learned via that interface. Split horizon typically comes into play on multiaccess interfaces where advertisements from multiple neighbors are learned via the same interface.

RIPv2 Configuration

To configure RIP, enter the RIP routing process in global configuration mode, and specify the interfaces that will run RIP by using the **network** command. The router then multicasts its routing table out all interfaces with IP addresses within the networks specified by that command. The **passive-interface interface** command stops RIP from sending updates out an interface. Use the **neighbor ip-address** command to inform RIP to send updates as unicasts to the specified neighbor.

Example 3-1 shows the configuration that enables RIP on all interfaces with IP addresses in the 10.0.0.0 range. The version of RIP is set to version 2, interface e0/1 is passive for RIP, and neighbor 10.1.1.2 is on a nonbroadcast network, thus updates are sent as unicast.

Example 3-1 RIPv2 Configuration

```
Router(config)# router rip
Router(config-router)# version 2
Router(config-router)# no auto-summary
Router(config-router)# network 10.0.0.0
Router(config-router)# passive-interface e0/1
Router(config-router)# neighbor 10.1.1.2
```

RIPng for IPv6

RIP next generation (RIPng) is the IPv6 version of RIP and is defined in RFC 2080. Like RIPv2 for IPv4, RIPng is a distance vector routing protocol that uses a hop count for its metric and has a maximum hop count of 15. It uses UDP but on port 521 instead of 520, and still has an administrative distance of 120. RIPng also sends periodic multicast updates—every 30 seconds—to advertise routes. The multicast address is FF02::9. The source address of RIPng updates is the link-local address of the outbound interface.

Two important differences exist between the old RIP and the next-generation RIP. One is that RIPng supports multiple concurrent processes, each identified by a process name. Another is that RIPng is initialized in global configuration mode and then enabled on specific interfaces. There is no **network** command in RIPng.

RIPng Configuration

Example 3-2 shows the syntax used to apply RIPng to a configuration. Notice that the syntax is similar to traditional RIP. You must first enable IPv6 routing. The global command to enable RIPng is optional; the router creates it automatically when the first interface is enabled for RIPng. You might need the command for additional configuration, such as originating a default route, as shown in Example 3-2.

Example 3-2 RIPng Configuration

```
Router(config)# ipv6 router rip process-name
!
Router(config)# interface type number
Router(config-if)# ipv6 rip process enable
Router(config-if)# ipv6 rip process default-information originate
```

Like RIP for IPv4, troubleshoot RIPng by looking at the routing table (**show ipv6 route [rip]**), by reviewing the routing protocols (**show ipv6 protocols**), and by watching routing updates propagated between routers (**debug ipv6 rip**).

CHAPTER 4

EIGRP

EIGRP Overview

Enhanced Interior Gateway Routing Protocol (EIGRP) is a Cisco proprietary, advanced distance vector, classless routing protocol that uses a composite metric based on multiple factors. The following are some features of EIGRP:

- Fast convergence.
- Partial updates conserve network bandwidth.
- Runs directly over IP, using protocol number 88.
- Support for all Layer 2 (data link layer) protocols and topologies.
- Sophisticated metric that supports load-balancing across unequal-cost paths.
- Use of multicast (and unicast where appropriate) instead of broadcasts.
- Support for authentication.
- Manual summarization at any interface.
- Uses multicast address 224.0.0.10 for EIGRPv4 and FF02::A for EIGRPv6.

EIGRP's function is controlled by three key technologies:

- **Neighbor discovery and maintenance:** Periodic hello messages
- **The Reliable Transport Protocol (RTP):** Controls sending, tracking, and acknowledging EIGRP messages
- **Diffusing Update Algorithm (DUAL):** Determines the best loop-free route

EIGRP uses three tables:

- The **Neighbor table** is built from EIGRP hellos and is used for reliable delivery.

- The **Topology table** contains EIGRP routing information for best paths and loop-free alternatives.
- EIGRP places best routes from its topology table into the common **Routing table**.

EIGRP Neighbor Establishment

EIGRP uses various message types to initiate and maintain neighbor relationships and to maintain an accurate routing table. It is designed to conserve bandwidth and router resources by sending messages only when necessary and only to those neighbors that need to receive them.

Packet Types

EIGRP uses five packet types:

- **Hello:** Identifies neighbors and serves as a keepalive mechanism
- **Update:** Reliably sends route information
- **Query:** Reliably requests specific route information
- **Reply:** Reliably responds to a query
- **ACK:** Acknowledgment

EIGRP is reliable, but hellos and ACKs are not acknowledged. The acknowledgment to a query is a reply.

If a reliable packet is not acknowledged, EIGRP periodically retransmits the packet to the nonresponding neighbor as a unicast. EIGRP has a window size of one, so no other traffic is sent to this neighbor until it responds. After 16 unacknowledged retransmissions, the neighbor is removed from the neighbor table.

Neighbor Discovery and Route Exchange

When EIGRP first starts on an interface, it goes through a process of neighbor discovery and route exchange as shown in the following steps:

- Step 1.** Router A sends out a hello.
- Step 2.** Router B sends back a hello and an update containing routing information. EIGRP AS number and K values must match to form a neighbor relationship. Timers do not have to match.

Step 3. Router A acknowledges the update.

Step 4. Router A sends its update.

Step 5. Router B acknowledges.

When two routers are EIGRP neighbors, they use periodic hellos between them as keepalives. Additional route information is sent only if a route is lost or a new route is discovered. A neighbor is considered lost if no hello is received within three hello periods (called the *Hold time*). The default Hello/ Hold timers are as follows:

- 5 seconds/15 seconds for multipoint circuits with bandwidth greater than T1 and for point-to-point media
- 60 seconds/180 seconds for multipoint circuits with bandwidth less than or equal to T1

Unicast Neighbors

EIGRP neighbors can also be statically defined using the command **neighbor** *{ipv4-address | ipv6 address} interface-type interface-number*. Messages to statically defined neighbors are sent as unicasts rather than multicasts. The neighbor IP address must be in the same subnet as one of the router's own interfaces.

Troubleshooting and Verifying EIGRP Neighbors

You can view the neighbor exchange process using **debug ip eigrp** packets and see the update process in real time using **debug ip eigrp**. You can see the neighbor table with the command **show ip eigrp neighbors** for IPv4 and **show ipv6 eigrp neighbors** for IPv6.

EIGRP Route Selection

EIGRP, as an *advanced* distance vector protocol, has a metric and route selection process that is more complex than other IGPs. The metric has the ability to reflect the actual usability of links within the network and can scale to accommodate today's faster bandwidths. EIGRP can use multiple paths, even if they do not have an equal metric.

EIGRP Metric

The traditional EIGRP metric is composed of

- **Bandwidth:** Minimum path bandwidth, in kb per second

- **Delay:** Cumulative path delay, in 10s of microseconds
- **Load:** Range from 1–255, with 255 being fully loaded
- **Reliability:** Range from 1–255, with 255 being most reliable

These values are associated with each interface and can be seen with the **show interface** command.

MTU and hop count are passed along but not used in the metric calculation.

The EIGRP metric is a 32-bit number calculated using the formula shown in Figure 4-1.

Figure 4-1 EIGRP Traditional Metric Formula

$$\text{metric} = 256 \left(k_1 \times \frac{10^7}{BW_{\min}} + \frac{k_2 \times BW_{\min}}{256 - \text{load}} + k_3 \times \sum \text{delays} \right) \left(\frac{k_5}{\text{reliability} + k_4} \right)$$

The k values are constants with default values of k1 = 1, k2 = 0, k3 = 1, k4 = 0, and k5 = 0. If k5 = 0, the final part of the equation (k5 / [reliability + k4]) is ignored.

If default k values are used, this works out to be 256 * (BW + cumulative delay).

Bandwidth is the largest contributor to the metric. The delay value enables you to choose a more direct path when bandwidth is equivalent.

Wide Metrics

Traditional EIGRP metrics are scaled in a way that does not pick up bandwidth differences in links over 10G and delay differences in links over 1G. (Specifically, the bandwidth value used is actually 10⁷/Interface-BW and the delay value actually used is Delay/10.) To address this, EIGRP release 8.0 and later supports *wide metrics*. Wide metrics use the actual, unscaled interface bandwidth reported in kb/sec. Delay is converted to picoseconds and is termed *latency*. Two additional metric values are included: jitter and energy. *Jitter* is variable delay, and *energy* is a measure of battery life for mobile networks. These are controlled via a new constant, k6, which is typically zeroed out. The resulting metric formula is shown in Figure 4-2 and still defaults to bandwidth + delay.

Figure 4-2 EIGRP Wide Metric Formula

$$\left[\left[K_1 \cdot \text{Throughput} + \left\{ \frac{K_2 \cdot \text{Throughput}}{256 - \text{Load}} \right\} \right] + (K_3 \cdot \text{Latency}) + (K_6 \cdot \text{Ext Attrib}) \right] \cdot \frac{K_5}{K_4 + \text{Reliability}}$$

EIGRP neighbors exchange capability information, and EIGRP has mechanisms to allow both traditional and wide metrics to function in the same network. A mixed network could cause some suboptimal routing and may require some adjustment with redistribution. For details, see the Cisco website for datasheets on wide metrics.

Diffusing Update Algorithm

Diffusing update algorithm (DUAL) is the algorithm used by EIGRP to choose the best loop-free paths. An EIGRP router receives advertisements from each neighbor listing its own metric to a network, plus the path delay and minimum bandwidth. This is the *reported distance*, or RD. The EIGRP router then compares its interface bandwidth and adds its interface delay to calculate its own metric. This is called the *feasible distance*, or FD.

Note

The reported distance (RD) is sometimes called the advertised distance (AD).

The path with the lowest feasible distance is called the *successor* and is submitted to the routing table. Nonsuccessor paths with a lower RD than the FD of the successor path are guaranteed loop-free and called *feasible successors*. They are included in the topology table as backup paths. If the successor path is lost, the router can use a feasible successor immediately without recalculating or risk of loops.

After the router has chosen a path to a network, it is passive for that route. If a successor path is lost and no feasible successor is identified, the router goes active for that route. It sends out queries on all interfaces in an attempt to identify an alternative path. No successor can be chosen until the router receives a reply to all queries. If a neighbor does not reply within 3 minutes, the router becomes stuck in active (SIA). In that case, it resets the neighbor relationship with the neighbor that did not reply.

Three common causes for SIA routes are as follows:

- CPU or memory usage is so high on the neighbor that it cannot process the query or reply.
- The link between the routers drops packets. Enough packets get through to maintain the neighbor relationship, but some queries or replies are dropped.
- Unidirectional link, so the router never receives packets from its neighbor.

Newer versions of EIGRP use SIA-Queries and SIA-Replies to prevent the loss of a neighbor unnecessarily during SIA conditions. A router sends its neighbor an SIA-Query after no reply to a normal query. If the neighbor responds with an SIA-Reply, the router does not terminate the neighbor relationship after 3 minutes because it knows the neighbor is available, just not able to answer the query.

You can see the active or passive state of a route, its successors, and feasible successors with the command **show ip eigrp topology** or **show ipv6 eigrp topology**.

EIGRP for IPv4

This section examines configuration for EIGRP for IPv4, along with some advanced options and ways to optimize EIGRP.

Basic Configuration

EIGRP for IPv4, commonly called *EIGRPv4*, is configured by entering router configuration mode and using the **network** command to identify the interfaces on which it will run. By default, EIGRP looks for neighbors out any interface with an IP address falling within the networks specified by that command. When setting up EIGRP, an autonomous system number must be used (7 is used in the example). Autonomous system numbers must agree for two routers to form a neighbor relationship and to exchange routes.

```
Router(config)# router eigrp 7  
Router(config-router)# network 192.168.1.0
```

The wildcard mask option can be used with the **network** command to more precisely identify EIGRP interfaces. For example, suppose a router has two interfaces—e0/0 (192.168.1.1) and e0/1 (192.168.1.33)—but needs to run EIGRP only on e0/0. Using a **network** command of 192.168.1.0 enables EIGRP on both interfaces, an undesired result. In this case, the following command can be used:

```
Router(config-router)# network 192.168.1.0 0.0.0.1
```

In this instance, a wildcard mask of 0.0.0.1 matches only two IP addresses in network 192.168.1.0–192.168.1.0 and 192.168.1.1. Therefore, only interface e0/0 is included in EIGRP routing.

To include all interfaces into EIGRP routing use **network 0.0.0.0**, which matches all IP addresses.

To ensure that the correct metric is calculated, or to influence the metric, you might want to configure the bandwidth on the interface. Use the **interface** command:

```
R1(config)# interface s0/0/0
R1(config-if)# bandwidth kbps
```

Optimizing the EIGRP Configuration

EIGRP provides some ways to customize its operation, such as passive interface, unicast neighbors, route summarization, unequal-metric load balancing, and authentication. This section describes how to configure these.

Passive Interface

The **passive-interface** command prevents either routing updates or hello messages from being sent out an interface, depending on the protocol. RIP does not send updates when it is enabled; EIGRP and OSPF do not send hellos, and thus they don't discover neighbors or form an adjacency out that interface. To disable the protocol on one interface, use the routing protocol configuration command **passive-interface interface**. To turn off the protocol on all interfaces, use **passive-interface default**. Then use **no passive-interface interface** for the ones that should run the protocol, as shown here:

```
Router(config)# router eigrp 7
Router(config-router)# passive-interface default
Router(config-router)# no passive-interface e0/0
```

Advertising a Default Route

EIGRP is typically configured to advertise a default route in the following ways:

- Create a static default route and redistribute it into EIGRP.
- Summarize route advertisements to 0.0.0.0 at an interface. Only neighbors out the interface receive the default route advertisement. This is shown in Example 4-1 in the next section.

Summarization

Route summaries are configured manually on an interface. When a summary is produced, a matching route to null0 is automatically placed in the routing table as a loop prevention mechanism. Configure a summary route out a particular interface using the **ip summary-address eigrp autonomous_system** command. Example 4-1 advertises a default route out

GigabitEthernet0/1 and the summary route 172.16.104.0/22 out Serial0/0/0 for EIGRP AS 7.

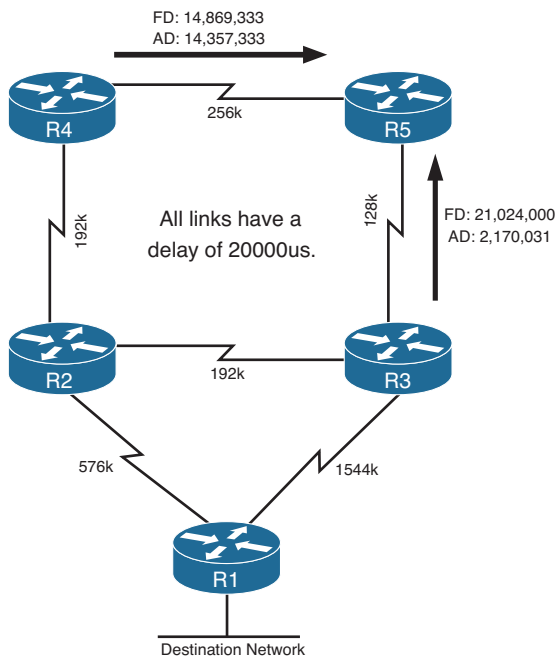
Example 4-1 Configuring EIGRP Summary Routes

```
Router(config)# int gi0/1
Router(config-if)# ip summary-address eigrp 7 0.0.0.0 0.0.0.0
!
Router(config)# int s0/0/0
Router(config-if)# ip summary-address eigrp 7 172.16.104.0
255.255.252.0
```

Load Balancing

Like most IP routing protocols, EIGRP automatically load balances over equal metric paths. What makes EIGRP unique is that you can configure it to proportionally load balance over *unequal* metric paths. The **variance** command is used to configure load balancing over up to six loop-free paths that have a metric lower than the product of the variance and the best metric. Figure 4-3 shows routers advertising a path to the network connected to R1.

Figure 4-3 EIGRP Unequal-Cost Load Balancing



By default, R5 uses the path through R4 because it offers the lowest metric (14,869,333). To set up unequal cost load balancing, assign a variance

of 2 under the EIGRP process on R5, which multiplies the best metric of 14,869,333 by 2 to get 29,738,666. R5 then uses all loop-free paths with a metric less than 29,738,666, which includes the path through R3. R5 then sends traffic along each path in proportion to its metric.

```
R5(config)# router eigrp 7
R5(config-router)# variance 2
```

EIGRP Authentication

By default, no authentication is used for any routing protocol. Some protocols, such as RIPv2, IS-IS, and OSPF, can be configured to do simple password authentication between neighboring routers. In this type of authentication, a clear-text password is used. EIGRP does not support simple authentication. However, it can be configured to authenticate each packet exchanged using an MD5 hash created from a preconfigured, shared password. This is more secure than clear text because only the message digest is exchanged, not the password. The password is called the *key*.

EIGRP authenticates each of its packets and verifies the source of each routing update by including the hash in each one. If the hash value does not match, the packet is silently dropped.

To configure the router for EIGRP authentication, follow these steps:

- Step 1.** Configure a key chain to group the keys.
- Step 2.** Configure one or more keys within that key chain. The router checks all inbound packets against the list of keys and uses the first valid one it finds.
- Step 3.** Configure the password or authentication string for that key. Repeat Steps 2 and 3 to add more keys if desired.
- Step 4.** Optionally configure a lifetime for the keys within that key chain. If you do this, be sure that the time is synchronized between the two routers.
- Step 5.** Enable authentication and assign a key chain to an interface.
- Step 6.** Designate MD5 as the type of authentication.

Example 4-2 shows a router configured with EIGRP authentication. It shows configuring a lifetime for packets sent using key 1 that starts at 10:15 and lasts for 300 seconds. It also shows configuring a lifetime for packets received using key 1 that starts at 10:00 and lasts until 10:05. Router clocks must be synchronized when using lifetimes, so use an NTP server.

Example 4-2 Configuring EIGRP Authentication

```
Router(config)# key chain RTR_Auth
Router(config-keychain)# key 1
Router(config-keychain-key)# key-string Password
Router(config-keychain-key)# send-lifetime 10:15:00 300
Router(config-keychain-key)# accept-lifetime 10:00:00 10:05:00
!
Router(config)# interface e0/0
Router(config-if)# ip authentication mode eigrp 10 md5
Router(config-if)# ip authentication key-chain eigrp 10 RTR_Auth
```

Verify your configuration with the **show key chain** command. **show ip eigrp neighbors** is also useful because no neighbor relationship is formed if authentication fails. Using the **debug eigrp packets** command shows packets containing authentication information sent and received, and it enables you to troubleshoot configuration issues. The debug output lists an authentication mismatch message if authentication does not succeed.

EIGRP Stub Routing

A stub router is connected to no more than two neighbors and should never be a transit router. Stub routing is one way to limit queries. When a router is configured as an EIGRP stub, it notifies its neighbors. The neighbors then do not query that router for a lost route. An EIGRP stub router still receives all routes from its neighbors—turning on stub routing does not cause its neighbor to advertise only a default route. You must manually configure that.

Under router configuration mode, use the command **eigrp stub [receive-only|connected|static|summary|redistributed]**. If you use only the command **eigrp stub**, then the router advertises only connected and summary routes. To change this, use one or more of the options listed in Table 4-1.

Table 4-1 eigrp stub Command Options

Command Option	Effect
receive-only	Prevents the router from advertising any networks, including its own. Cannot be combined with any other option. BE CAREFUL with this option—it will prevent other routers from learning the local routes.
Connected	Enables the router to advertise connected routes. These must either be included in a network statement or redistributed into EIGRP. Enabled by default.

Command Option	Effect
Static	Enables the router to advertise static routes. They must be redistributed into EIGRP before they will be advertised.
Summary	Enables the router to advertise summary routes, both those created manually and automatically. Enabled by default.
Redistributed	Enables the router to advertise routes redistributed into EIGRP from another protocol or AS.

Verify a neighbor router's stub status with the **show ip eigrp neighbor detail** or **show ipv6 eigrp neighbor detail** command.

EIGRP for IPv6

EIGRP has been expanded to support IPv6, although you must verify that your specific version of IOS is capable of doing this. EIGRP for IPv6 is based on the IPv4 version, and the two can run in tandem on the same router and on the same interfaces. EIGRPv6 is still an advanced distance vector routing protocol that uses a complex metric, has a reliable update mechanism, and uses DUAL to retain fallback paths. Like EIGRPv4, it sends multicast hellos every 5 seconds, but the multicast address is now FF02::A. Messages are exchanged using the interface link-local address as the source address.

The router ID must still be in the form of an IPv4 address. If there are no active IPv4 addresses on the router, you must statically assign an ID; it does not need to be a routable address. Like RIPng, there is no **network** command; EIGRP routing is enabled at each interface. After this is accomplished, the router looks for neighbors out that interface and advertises the interface networks by default. Example 4-3 shows how to enable basic IPv6 EIGRP.

Example 4-3 Basic EIGRP IPv6 Commands

```
Router(config)# ipv6 unicast-routing
!
Router(config)# ipv6 router eigrp AS
Router(config-rtr)# router-id ipv4-address
!
Router(config)# interface type number
Router(config-if)# ipv6 eigrp AS
```

IPv6 EIGRP can summarize routes at the interface, and the stub feature is also available, just as with the IPv4 version. Example 4-4 shows a sample

configuration for IPv6 EIGRP, with both summarization and stub routing enabled. Notice that the routing protocol is enabled under each interface.

Example 4-4 Configuring EIGRP for IPv6

```
Router(config)# ipv6 router eigrp 1
Router(config-rtr)# eigrp router-id 10.255.255.1
Router(config-rtr)# eigrp stub connected summary
!
Router(config)# interface gigabitethernet0/0
Router(config-if)# description Local LAN
Router(config-if)# ipv6 address 2001:DB8:0:10::1/64
Router(config-if)# ipv6 eigrp 1
!
Router(config-if)# interface serial 1/0/1
Router(config-if)# description point-to-point link to Internet
Router(config-if)# ipv6 address 2001:DB8:0:11::2/64
Router(config-if)# ipv6 eigrp 1
Router(config-if)# ipv6 summary address eigrp 1 2001:DB8:0:10::/63
```

Verifying and Troubleshooting EIGRP

The commands for verifying EIGRP differ slightly between IPv4 and IPv6 versions. In the following commands, the first one listed is the IPv4 command and the second one is the IPv6 command:

- **show ip route/show ipv6 route** displays the routing table.
- **show ip route eigrp/show ipv6 route eigrp** filters the routing table and displays only the routes learned from EIGRP.
- **show ip protocols/show ipv6 protocols** verifies autonomous system, timer values, identified networks, and EIGRP neighbors (routing information sources).
- **show ip eigrp topology/show ipv6 topology** shows the EIGRP topology table and identifies successors and feasible successors.
- **show ip eigrp neighbors/show ipv6 neighbors** verifies that the correct routers are neighbors.
- **show ip eigrp traffic/show ipv6 traffic** shows the amount and types of EIGRP messages.
- **show ip eigrp interfaces/show ipv6 eigrp interfaces** lists the interfaces participating in EIGRP and any neighbors located out those interfaces, along with some other statistics.

Like EIGRP for IPv4, troubleshoot EIGRPv6 by looking at the routing table (**show ipv6 route**), by reviewing the routing protocols (**show ipv6 protocols**), and by monitoring neighbors (**show ipv6 eigrp neighbors**).

EIGRP Named Mode

Basic EIGRP separates IPv4 and IPv6 configuration in different places. Group the configuration by using EIGRP in *named* mode. To accomplish this, create a named EIGRP instance, and then all other configuration is given under either an IPv4 or IPv6 address family. You can configure actions such as summary addresses, stub routing, variance, and redistribution. There are three general configuration modes:

- **Address-family:** Configures general settings such as router ID, stub, and network (for IPv4)
- **Address-family Interface:** Configures interface specific settings such as authentication, timers, passive interface, and summary address
- **Address-family Topology:** Configures settings that impact the topology table such as variance, redistribution, and metrics

Example 4-5 shows how named mode can be used to configure EIGRP for both IPv4 and IPv6.

Example 4-5 Configuring EIGRP Named Mode

```
Router(config)# router eigrp CCNP
Router(config-router)# address-family ipv4 autonomous-system 2
Router(config-router-af)# network 0.0.0.0
!
Router(config-router-af)# af-interface default
Router(config-router-af-interface)# passive-interface
!
Router(config-router-af)# af-interface e0/0
Router(config-router-af-interface)# summary-address 172.16.1.0/24
Router(config-router-af-interface)# no passive-interface
!
Router(config-router-af-interface)# topology base
Router(config-router-af-topology)# variance 2
!
Router(config-router)# address-family ipv6 unicast autonomous-
system 2
Router(config-router-af)# eigrp router-id 1.1.1.1
Router(config-router-af)# eigrp stub
!
```

Example 4-5 Continued

```

Router(config-router-af) # af-interface e0/0
Router(config-router-af-interface) # summary-address
    2001:db8:0:100::/62
!
Router(config-router-af) # topology base
Router(config-router-af-topology) # variance 2

```

EIGRP Over WAN Links

EIGRP can be used across many types of WAN links. This section examines how it operates over some of them.

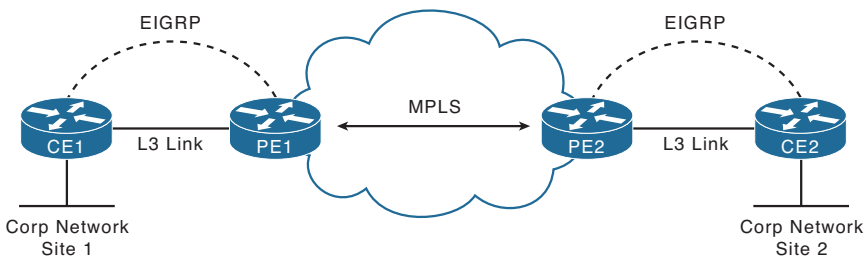
Layer 3 MPLS WAN

MPLS provides either a Layer 2 or a Layer 3 connection. In MPLS terminology, your WAN edge routers are called *CE* (customer edge) routers, and the ISP's WAN edge routers are called *PE* (provider edge) routers. Within the ISP's network are *P* (provider) routers, but they are not visible to the CE.

PE routers are involved in routing when using EIGRP over Layer 3 MPLS VPNs. Connected PE and CE routers form an EIGRP neighbor relationship. The PE router is just another neighbor to the CE router; it is not aware of the MPLS network or the ISP's P routers.

In Figure 4-4, CE1 creates an EIGRP neighbor relationship with PE1. CE1 sends routing updates about its networks to PE1, which installs the routes in the correct Virtual Routing and Forwarding (VRF) table and then transmits them across the WAN as MPLS packets to PE2. PE2 is an EIGRP neighbor to CE2, so it forwards the route advertisements as normal EIGRP updates.

Figure 4-4 Using EIGRP with Layer 3 MPLS WAN



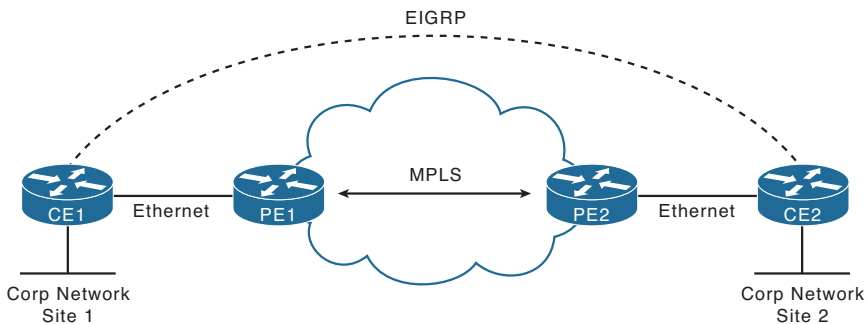
When using EIGRP over MPLS, the customer and the provider need to use the same basic EIGRP configuration such as AS number and authentication.

Layer 2 WAN

When you use a Layer 2 WAN such as Metro Ethernet or Layer 2 MPLS, each router forms a neighbor relationship with all other CE routers on the WAN link. In reality, each CE router has an Ethernet connection to its local PE router, but the WAN acts as a Layer 2 switch to the CE routers.

Figure 4-5 shows how this works. The PE1 router receives Ethernet frames from CE1; encapsulates them into an MPLS packet (in this case the WAN is MPLS); and then forwards them across the WAN to PE2, which is the local router connected to CE2. PE2 deencapsulates the packet, rebuilds the Ethernet frame, and sends it to the CE2.

Figure 4-5 Using EIGRP with Layer 2 WAN



It is important to understand that CE1 and CE2 build an EIGRP neighbor relationship with each other. The ISP routers are not involved in routing with the CE routers. Additionally, the PE routers do not learn any MAC addresses or participate in Spanning Tree.

Frame Relay

An issue with using EIGRP over Frame Relay is that one physical interface can support multiple logical connections, each identified by a Data Link Connection Identifier (DLCI). These are Layer 2 connections and must be mapped to a Layer 3 neighbor IP address. This mapping can be done either dynamically or statically. Multipoint interfaces are used in partial and full mesh topologies.

Dynamic mapping uses Inverse ARP. Routers form EIGRP neighbor adjacencies only with routers to which they connect via a Frame Relay virtual circuit (VC). Static mapping requires manual configuration under each interface but enables routers without VC connections to become neighbors. The static mapping command is given under interface configuration mode:

```
frame-relay map ip remote-ip-address local-dlci broadcast
```

The **broadcast** keyword is required because Frame Relay is, by default, a nonbroadcast medium. Static mapping can be used with both physical multipoint interfaces and subinterfaces. Note that a multipoint interface stays up if one DLCI is active, so a neighbor loss might not be detected until the Hold timer expires.

Frame Relay can emulate physical point-to-point links by using point-to-point subinterfaces, used in a hub-and-spoke topology. Neighbor loss is detected much more quickly on point-to-point links for two reasons:

- The default timers are shorter: 5 second Hold timer and 15 second Dead timer.
- The subinterface goes down when its associated DLCI goes down.

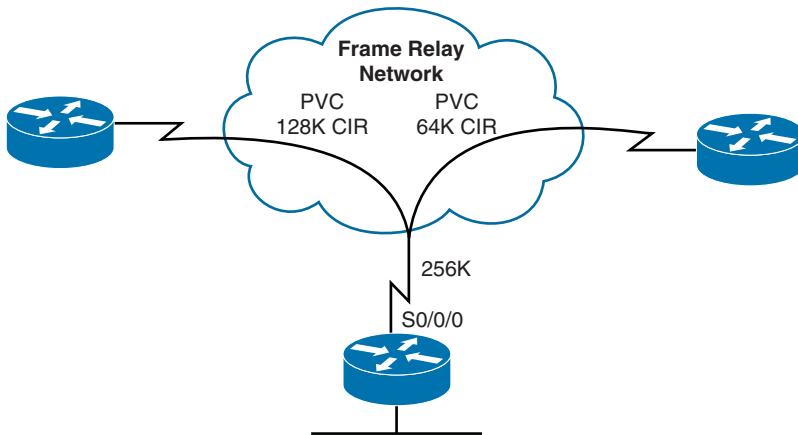
By default, EIGRP limits itself to bursting to half the link bandwidth. This limit is configurable per interface using the **ip bandwidth-percent** command. The following configuration assumes EIGRP AS 7 and limits EIGRP to one-quarter of the link bandwidth:

```
Router(config)# int s0/0/0
Router(config-if)# ip bandwidth-percent eigrp 7 25
```

The real issue with WAN links is that the router assumes that each link has 1544 kbps bandwidth. If interface Serial0/0/0 is attached to a 128 k fractional T1, EIGRP assumes it can burst to 768 k and could overwhelm the line. This is rectified by correctly identifying link bandwidth:

```
Router(config)# int serial 0/0/0
Router(config-if)# bandwidth 128
```

Figure 4-6 shows a situation in which these techniques can be combined: Frame Relay.

Figure 4-6 EIGRP with Frame Relay

In Figure 4-6, the hub router has a 256 kbps connection to the Frame Relay network and two permanent virtual circuits (PVC) with committed information rates (CIR) of 128 Kbps and 64 Kbps. EIGRP divides the interface bandwidth evenly between the neighbors on that interface. What value should be used for the interface bandwidth in this case? The usual suggestion is to use the CIR, but the two PVCs have different CIRs. You can use the **bandwidth-percent** command to allow SNMP reporting of the true bandwidth value, while adjusting the interface burst rate to 25 percent, or 64 kbps.

```
Router(config)# int serial 0/0/0
Router(config-if)# bandwidth 256
Router(config-if)# ip bandwidth-percent eigrp 7 25
```

A better solution is to use point-to-point subinterfaces and identify bandwidth separately. In Example 4-6, s0/0/0.1 bursts to 64 k, and s0/0/0.2 bursts to 32 k, using EIGRP's default value of half the bandwidth.

Example 4-6 Configuring Frame Relay Subinterfaces

```
Router(config)# int serial 0/0/0.1 point-to-point
Router(config-if)# bandwidth 128
Router(config-if)# frame-relay interface-dlci 100
!
Router(config)# int serial 0/0/0.2 point-to-point
Router(config-if)# bandwidth 64
Router(config-if)# frame-relay interface-dlci 101
```

In cases where the hub interface bandwidth is oversubscribed, it might be necessary to set bandwidth for each subinterface arbitrarily low and then specify an EIGRP bandwidth percent value over 100 to allow EIGRP to use half the PVC bandwidth.

EIGRP Over the Top

EIGRP Over the Top (OTP) is a technique to run EIGRP across a WAN that is transparent to the WAN provider. It does not require route redistribution between BGP and EIGRP on customer edge routers and can be used over any type of WAN link. Location ID Separation Protocol (LISP) encapsulates and routes the EIGRP traffic over the WAN. This feature was introduced in IOS version 15.4(1)T.

Basically, you configure unicast EIGRP neighbors on each CE router, but you must also specify LISP encapsulation for them.

```
Router(config) # router eigrp CCNP  
Router(config-router) # address-family ipv4 autonomous-system 2  
Router(config-router-af) # neighbor 172.16.1.2 e0/0 lisp-encap 1
```

In large networks with many CE routers, you might want to use EIGRP route reflectors to cut down on the number of EIGRP neighbor relationships that must be maintained. For more information on this, see the documentation at Cisco.com.

CHAPTER 5

OSPF

OSPF Overview

Open Shortest Path First (OSPF) is an open-standard link-state routing protocol that uses the Shortest Path First (SPF) algorithm to determine its best path to each network. OSPFv2 supports IPv4 routing and OSPFv3 supports both IPv4 and IPv6 routing.

Link-state routers create a database that reflects the structure of their area in the network. Link-state routing protocols learn more information on the structure of the network than other routing protocols and thus can make more informed routing decisions.

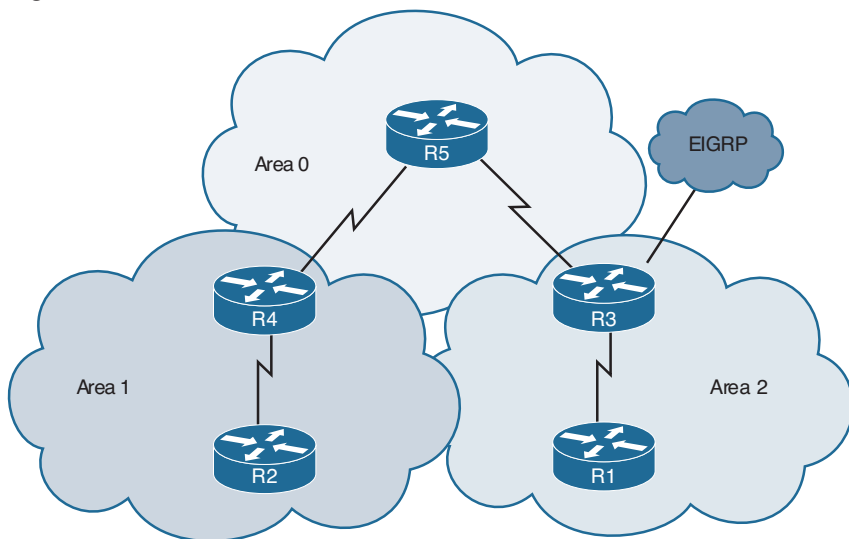
OSPF routers exchange hellos with each neighbor, learning router ID (RID) and cost. Neighbor information is kept in the Adjacency database. The router then constructs the appropriate link-state advertisements (LSA), which include information such as the RIDs of, and cost to, each neighbor. Each router keeps the complete set of LSAs in the link-state database (LSDB).

Each router runs the SPF algorithm to compute best paths. It then submits these paths for inclusion in the routing table or forwarding database.

OSPF Network Structure

OSPF routing domains are broken up into areas. An OSPF network must contain an Area 0 and might contain other areas. The SPF algorithm runs within an area, and interarea routes are passed between areas. OSPF has a two-level hierarchy: Area 0 is designed as a transit area, and other areas should be attached directly to Area 0 and only to Area 0. The link-state database must be identical for each router in an area. OSPF areas typically contain a maximum of 50 routers to 100 routers, depending on network volatility.

Figure 5-1 shows a network of five routers that has been divided into three areas: Area 0, Area 1, and Area 2.

Figure 5-1 OSPF Areas

Dividing an OSPF network into areas accomplishes the following:

- Minimizes the number of routing table entries
- Contains LSA flooding to a reasonable area
- Minimizes the impact of a topology change
- Enforces the concept of a hierarchical network design

The following are several types of areas:

- **Backbone area:** Area 0, which is attached to every other area.
- **Regular area:** Nonbackbone area; its database contains both internal and external routes.
- **Stub area:** Its database contains only internal routes, summary routes, and a default route.
- **Totally Stubby Area:** Cisco proprietary area designation. Its database contains routes only for its own area and a default route.
- **Not-so-stubby area (NSSA):** Its database contains internal routes, summary routes, routes redistributed from a connected routing process, and optionally a default route.
- **Totally NSSA:** Cisco proprietary area designation. Its database contains only routes for its own area, routes redistributed from a connected routing process, and a default route.

OSPF defines router roles as well. One router can have multiple roles, such as the following:

- An internal router has all interfaces in one area. In Figure 5-1, Routers R1, R2, and R5 are all internal area routers. They maintain a link-state database for their own area only.
- Backbone routers have at least one interface assigned to Area 0. R3, R4, and R5 are backbone routers.
- An area border router (ABR) has interfaces in two or more areas. In Figure 5-1, R3 and R4 are ABRs. ABRs separate LSA flooding areas, can summarize area routes, and can source default routes. They maintain a link-state database for each area to which they are connected.
- An autonomous system boundary router (ASBR) has interfaces inside and outside the OSPF routing domain. In Figure 5-1, R3 functions as an ASBR because it has an interface in an EIGRP routing domain.

OSPF Metric

OSPF uses a cost value as its metric; in Cisco routers, the cost is based on the interface bandwidth. Routers assign a cost to each interface that is equal to 100 Mbps divided by the interface bandwidth. The cost for each link is then accrued as the route advertisement about that link traverses the network.

This default formula results in a cost of 1 for interfaces with speeds of 100 Mbps and faster. It assigns the same cost to a Fast Ethernet interface and a 10Gigabit Ethernet interface, for example. The cost formula can be adjusted using the **auto-cost** command under the OSPF routing process. Values for bandwidth (in kbps) up to 4,294,967 are permitted (1 Gbps is shown in the following line):

```
Router(config-router)# auto-cost reference-bandwidth 1000
```

The cost can also be manually assigned under the interface configuration mode. The cost is a 16-bit number, so it can be any value from 1 to 65,535.

```
Router(config-if)# ip ospf cost 27
```

Link-State Advertisements

Each router maintains a database, called the *link-state database (LSDB)*, containing the latest received LSAs. A separate LSDB is maintained for each area connected to the router.

LSA Operation

Each LSA is assigned a sequence number, and a timer is run to age out old LSAs. The default timer is 30 minutes.

When a LSA is received, it's compared to the LSDB. If it is new, it is added to the database and the SPF algorithm is run. If it is from a router ID that is already in the database, the sequence number is compared and older LSAs are discarded. If it is a new LSA, it is incorporated in the database and the SPF algorithm is run. If it is an older LSA, the newer LSA in memory is sent back to whomever sent the old one.

OSPF sequence numbers are 32 bits. The first legal sequence number is 0x80000001. Larger numbers are more recent. The sequence number changes only under two conditions:

- The LSA changes because a route is added or deleted.
- The LSA ages out. (LSA updates are flooded within the area every half hour, even if nothing changes.)

The command **show ip ospf database** shows the age (in seconds) and sequence number for each router.

LSA Types

OSPF uses different types of LSAs to advertise different types of routes, such as internal area or external routing domain. Many of these are represented in the routing table with a distinctive prefix. OSPFv3, for IPv6, has added some LSA types and renamed some others. OSPFv3 LSA types are covered later in the section “OSPF for IPv6.”

Table 5-1 describes the LSA types for OSPFv2.

Table 5-1 OSPF LSA Types

Type	Description	Routing Table Symbol
1	Router LSA. Advertises intra-area routes. Generated by each OSPF router. Flooded only within the area.	O
2	Network LSA. Advertises routers on a multi-access link. Generated by a Designated Router (DR). Flooded only within the area.	O
3	Summary LSA. Advertises interarea routes. Generated by an ABR. Flooded to adjacent areas.	O IA

Type	Description	Routing Table Symbol
4	ASBR summary LSA. Advertises the route to an ASBR. Generated by an ABR. Flooded to adjacent areas.	O IA
5	External LSA. Advertises routes in another routing domain. Generated by an ASBR. Flooded to adjacent areas. E1—Metric increased by each router E2—Metric does not increase (the default)	O E1 O E2
6	Group membership LSA. Used in multicast OSPF operations. Not supported by Cisco.	
7	Not-so-stubby area (NSSA) LSA. Advertises routes in another routing domain. Generated by an ASBR within a not-so-stubby area. Flooded only within the NSSA. N1—Metric increased by each router N2—Metric does not increase (the default)	O N1 O N2
8	External attributes LSA. Used in OSPF and BGP interworking.	

OSPF Operation

OSPF uses several different message types to establish and maintain its neighbor relationships and to maintain correct routing information.

OSPF Packets

OSPF uses five packet types. It does not use UDP or TCP for transmitting its packets. Instead, it runs directly over IP (IP protocol 89) using an OSPF header. One field in this header identifies the type of packet being carried. The five OSPF packet types are as follows:

- **Hello:** Identifies neighbors and serves as a keepalive.
- **Link-state request (LSR):** Request for a link-state update (LSU). Contains the type of LSU requested and the ID of the router requesting it.
- **Database description (DBD):** A summary of the LSDB, including the RID and sequence number of each LSA in the LSDB.

- **Link-state update (LSU):** Contains a full LSA entry. An LSA includes topology information; for example, the RID of this router and the RID and cost to each neighbor. One LSU can contain multiple LSAs.
- **Link-state acknowledgment (LSAck):** Acknowledges all other OSPF packets (except hellos).

OSPF for IPv4 traffic is multicast to either of two addresses: 224.0.0.5 for all OSPF routers or 224.0.0.6 for all OSPF DRs. OSPF for IPv6 routing uses the multicast addresses FF02::5 for all OSPF routers and FF02::6 for all OSPF DRs.

OSPF Neighbor Relationships

OSPF routers send out periodic multicast packets to introduce themselves to other routers on a link. They become neighbors when they see their own router ID included in the Neighbor field of the hello from another router. This signals to each router that they have bidirectional communication. In addition, two routers must be on a common subnet for a neighbor relationship to be formed. (Virtual links are sometimes an exception to this rule.)

Certain parameters within the OSPF hellos must also match for two routers to become neighbors. They include

- Hello/dead timers
- Area ID
- Authentication type and password
- Stub area flag

DBD packets also contain the interface MTU information. This must match on both sides of the link or the neighbor relationship will be stuck in Exstart state.

OSPF routers can be neighbors without being adjacent. However, only adjacent neighbors exchange routing updates and synchronize their databases. On a point-to-point link, an adjacency is established between the two routers when they can communicate. On a multiaccess link, each router establishes an adjacency only with the DR and the backup DR (BDR).

Hellos also serve as keepalives. A neighbor is considered lost if no hello is received within four hello periods (called the dead time). The default hello/dead timers are as follows:

- 10 seconds/40 seconds for LAN and point-to-point interfaces
- 30 seconds/120 seconds for nonbroadcast multiaccess (NBMA) interfaces

These steps show the process of neighbor establishment and route exchange between two OSPF routers:

- Step 1. DOWN state:** OSPF process not yet started, so no hellos sent.
- Step 2. INIT state:** Router sends hello packets out all OSPF interfaces.
- Step 3. Two-way state:** Router receives a hello from another router that contains its own router ID in the neighbor list. All other required elements match, so routers can become neighbors.
- Step 4. Exstart state:** If routers become adjacent (exchange routes), they determine which one starts the exchange process.
- Step 5. Exchange state:** Routers exchange DBDs listing the LSAs in their LSD by RID and sequence number.
- Step 6. Loading state:** Each router compares the DBD received to the contents of its LS database. It then sends an LSR for missing or outdated LSAs. Each router responds to its neighbor's LSR with a link-state update. Each LSU is acknowledged.
- Step 7. Full state:** The LSDB has been synchronized with the adjacent neighbor.

Basic OSPFv2 Configuration

OSPF is configured by entering router configuration mode and identifying the range of interface addresses on which it should run and the areas they are in. When setting up OSPF, a process ID must be used (8 is used in the example), but the process ID does not need to agree on different OSPF devices for them to exchange information. The **network** statement uses a wildcard mask and can specify any range from a single address to all addresses. Unlike EIGRP, the wildcard mask is not optional. The following example shows a router configured as an ABR. Interfaces falling within the 192.168.1.0/24 network are placed in area 0, and interfaces falling within the 172.16.1.0/24 network are placed in area 1.

```
Router(config)# router ospf 8
Router(config-router)# network 192.168.1.0 0.0.0.255 area 0
Router(config-router)# network 172.16.1.0 0.0.0.255 area 1
```

Alternatively, you can enable OSPF directly on an interface rather than use a **network** statement. This is especially helpful on unnumbered interfaces and enables very granular control over which interfaces run OSPF.

```
Router(config)# int s0/0/0
Router(config-if)# ip ospf 8 area 0
```

The **ip ospf area** interface command takes precedence over a **network** command.

Router ID

The SPF algorithm maps the shortest path between a series of nodes. This causes an issue with IP because an IP router is not identified by a single IP address; its interfaces are. For this reason, a single IP address is designated as the “name” of the router: the router ID (RID).

By default, the RID is the highest loopback IP address. If no loopback addresses are configured, the RID is the highest IP address on an active interface when the OSPF process is started. The RID is selected when OSPF starts and—to maintain network stability—is not changed until the OSPF process restarts. The OSPF process can be restarted by rebooting the router or by using the command **clear ip ospf process**. Either choice affects routing in your network for a period of time and should be used only with caution.

A loopback interface is a virtual interface, so it is more stable than a physical interface for RID use. The loopback address does not need to be included in the OSPF routing process, but if you advertise it, you can ping or trace to it. This is helpful in troubleshooting.

A way to override the default RID selection is to statically assign it using the **ospf router-id** command. Router ID is typically statically assigned for predictability if a process is forced to unexpectedly restart. It does not have to be an actual IP address used by the router, but best practice is to use a loopback interface address.

```
Router(config)# router ospf 8
Router(config-router)# router-id 10.0.0.1
```

Verifying and Troubleshooting OSPF

You can view the neighbor initialization process using the **debug ip ospf adj** command. You can see the neighbor table with **show ip ospf neighbors**, which also identifies adjacency status and reveals the designated router and

backup designated router. Use the **debug ip ospf packet** command to view all OSPF packets in real time.

Often, the first place you notice OSPF issues is when inspecting the routing table: **show ip route**. To filter the routing table and show only the routes learned from OSPF, use **show ip route ospf**.

The command **show ip protocols** offers a wealth of information for any routing protocol issue. Use this command to verify parameters, timer values, identified networks, and OSPF neighbors (routing information sources).

Use **show ip ospf** to verify the RID, timers, and counters. Because wildcard masks sometimes incorrectly group interfaces to areas, another good place to check is **show ip ospf interface**. This shows the interfaces on which OSPF runs and their current correct assigned area.

OSPF Network Types

The SPF algorithm builds a directed graph—paths made up of a series of points connected by direct links. One of the consequences of this directed-graph approach is that the algorithm has no way to handle a multiaccess network, such as an Ethernet segment. The solution used by OSPF is to elect one router, called the *designated router* (DR), to represent the entire segment. Point-to-point links fit the SPF model perfectly and don't need any special modeling method. On a point-to-point link, no DR is elected and all traffic is multicast to 224.0.0.5.

OSPF supports five network types:

- **NBMA:** Default for multipoint serial interfaces and physical frame relay interfaces. RFC-compliant mode that uses DRs and requires manual neighbor configuration.
- **Point-to-multipoint (P2MP):** Doesn't use DRs so adjacencies increase logarithmically with routers. Resilient RFC-compliant mode that automatically discovers neighbors.
- **Point-to-multipoint nonbroadcast (P2MNB):** Proprietary mode that is used on Layer 2 facilities where dynamic neighbor discovery is not supported. Requires manual neighbor configuration.
- **Broadcast:** Default mode for LANs. Uses DRs and automatic neighbor discovery. Proprietary when used on WAN interface.
- **Point-to-point (P2P):** Proprietary mode that discovers neighbors and doesn't require a DR.

If the default interface type is unsatisfactory, you can statically configure it with the command **ip ospf network** under interface configuration mode:

```
Router(config-if)# ip ospf network network-type
```

When using the NBMA or P2MP nonbroadcast mode, neighbors must be manually defined under the routing process:

```
Router(config-router)# neighbor 172.16.0.1
```

The command **show ip ospf interface** displays the network type for each link.

Designated Routers

On a multiaccess link, one of the routers is elected as a DR and another as a backup DR (BDR). All other routers on that link become adjacent only to the DR and BDR, not to each other. (They stop at the two-way state.) The DR is responsible for creating and flooding a network LSA (type 2) advertising the multiaccess link. NonDR (DROTHER) routers communicate with DRs using the IP address 224.0.0.6. The DRs use IP address 224.0.0.5 to pass information to other routers.

The DR and BDR are elected as shown in the following steps:

- Step 1.** A router starting the OSPF process listens for OSPF hellos. If none are heard within the dead time, it declares itself the DR.
- Step 2.** If hellos from any other routers are heard, the router with the highest OSPF priority is elected DR and the election process starts again for BDR. A priority of zero removes a router from the election.
- Step 3.** If two or more routers have the same OSPF priority, the router with the highest RID is elected DR and the election process starts again for BDR.

After a DR is elected, elections do not take place again unless the DR or BDR are lost. Because of this, the DR is sometimes the first device that comes online with a nonzero priority.

The best way to control DR election is to set OSPF priority for the DR and BDR for other routers. The default priority is 1. A priority of 0 means that a router cannot act as DR or BDR; it can be a DROTHER only. Priority can be set with the **ip ospf priority** command in interface configuration mode.

```
Router(config)# int gi0/1
Router(config-if)# ip ospf priority 2
```

Verify the DR and BDR election with the **show ip ospf neighbor** command as shown in Example 5-1.

Example 5-1 The show ip ospf neighbor Command

Router# show ip ospf neighbor						
Neighbor ID	Pri	State	Dead Time	Interface ID	Interface	
4.4.4.4	1	FULL/DR	00:00:38	3	Ethernet0/0	
3.3.3.3	1	FULL/BDR	00:00:36	4	Ethernet0/1	

Nonbroadcast Multiaccess Networks

Routing protocols assume that multiaccess links support broadcast and have full-mesh connectivity from any device to any device. In terms of OSPF, this means the following:

- Frame Relay map statements should include the broadcast attribute.
- The DR and BDR should have full virtual circuit connectivity to all other devices.
- Hub-and-spoke environments should either configure the DR as the hub or use point-to-point subinterfaces, which require no DR. Use OSPF priority 0 to prevent spokes from becoming DR.
- Partial-mesh environments should be configured using point-to-point subinterfaces, especially when no single device has full connectivity to all other devices. If there is a subset of the topology with full connectivity, that subset can use a multipoint subinterface.
- Full-mesh environments can be configured using the physical interface, but often logical interfaces are used to take advantage of the other benefits of subinterfaces.
- Statically identifying neighbor IP addresses might be necessary.

OSPF over Layer 2 and Layer 3 MPLS

A Layer 2 Multiprotocol Label Switching (MPLS) connection provides logical mesh connectivity at Layer 2 and OSPF operates just as it would on an Ethernet network. It forms a neighbor relationship with the customer edge (CE) router across the WAN, and they elect a DR and BDR. The OSPF network type is multiaccess broadcast.

A Layer 3 MPLS virtual private network (VPN) requires that the CE routers form an OSPF neighbor relationship with their connected provider edge (PE) router. The PE router appears to the enterprise as just another router within their network. The OSPF network type is determined by the type of link between the CE and PE. Carefully consider your area design when using this type of WAN.

Advanced OSPF Configuration

OSPF provides many different ways to customize its operation to fit your network needs. This section discusses route summarization, passive interfaces, default routes, stub areas, and virtual links.

OSPF Summarization

Summarization helps all routing protocols scale to larger networks, but OSPF especially benefits because its processes tax the router memory and CPU resources. OSPF's multiple databases use more memory the larger they are. Summarization decreases the number of routes exchanged, and thus the size of the databases. Summarization localizes the impact of a topology change. It prevents topology changes from being passed outside an area and thus saves routers in other areas from having to run the SPF algorithm. One disadvantage is that summarization hides information, and any time you hide data you decrease the router's capability to make informed routing decisions.

Unlike EIGRP, OSPF can only summarize routes on specific routers—ABRs and ASBRs. There are two types of summarizations:

- Inter-area (LSA type 3) route summarizations are created on the ABR under the OSPF routing process using the **area range** command. A summary route is advertised as long as at least one subnet within the summary is active in the area. The summary route's metric is the lowest cost route within the summary range. The router automatically creates a static route for the summary, pointing to Null0.

The following command advertises 172.16.0.0/12 from Area 1:

```
Router(config-router)# area 1 range 172.16.0.0 255.240.0.0
```

- External (LSA type 5) route summarization is done on an ASBR using the **summary-address** command under the OSPF routing process. It can also be done on the ABR of a NSSA to summarize type 7 routes before advertising them as type 5. The router automatically creates a

static route for the summary, pointing to Null0. The following configuration summarizes a range of external routes to 192.168.0.0/16 and injects a single type 5 route into OSPF:

```
Router(config-router)#summary-address 192.168.0.0 255.255.0.0
```

Passive Interface

The **passive-interface** command prevents OSPF from sending hello messages out an interface. Thus, an OSPF router does not discover neighbors or form an adjacency out that interface. To disable the protocol on one interface, use the routing protocol configuration command **passive-interface interface**. To turn off the protocol on all interfaces, use **passive-interface default**. Then use **no passive-interface interface** for the ones that should run the protocol.

OSPF Default Routes

Default routes are a special type of summarization; they include all networks in one route announcement. By reducing routing information to a minimum, a default route provides such benefits as the following:

- Routers have a smaller routing table.
- Less use of router resources to advertise multiple routes.
- Routers do not need to keep information on external routes.

A default route is injected into OSPF as a *type 5 route*. There are several ways to use the router IOS to place a default route into OSPF. The best-known way is to use the **default-information** command under the OSPF routing process. This command, without the keyword **always**, advertises a default route learned from another source (such as a static route) into OSPF. If the **always** keyword is present, OSPF advertises a default even if that route does not already exist in the routing table. The **metric** keyword sets the starting metric for this route.

```
Router(config-router)# default-information originate [always]
[metric metric]
```

Alternatively, a default summary route can also be produced using the **summary-address** command or the **area range** command. These commands can cause the router to advertise a default route pointing to itself.

Stub and Not-So-Stubby Areas

Another way to reduce the route information advertised is to create a *stub* area. Configuring a stub area forces its ABR to drop all external (type 5) routes and replaces them with a default route. To limit routing information even more, an area can be made *Totally Stubby* using the **no-summary** keyword on the ABR only. In that case, all interarea and external routes are dropped by the ABR and replaced by a default route. The default route starts with a cost of 1; to change it, use the **area default-cost** command. The example that follows shows Area 2 configured as a Totally Stubby Area, and the default route injected with a cost of 5:

```
Router(config-router)# area 2 stub no-summary
Router(config-router)# area 2 default-cost 5
```

Stub areas are attractive because of their low overhead. They do have some limitations, including the following:

- Stub areas can't include a virtual link.
- Stub areas can't include an ASBR.
- Stubiness must be configured on all routers in the area.
- Area 0 cannot be a stub area.

Another kind of stub area is a *not-so-stubby area (NSSA)*. NSSA is like a stub or totally stub area but enables an ASBR within the area. External routes are advertised as type 7 routes by the ASBR. The ABR converts them to type 5 external routes when it advertises them into adjacent areas. NSSA is configured with the **area nssa** command under the OSPF routing process. The **no-summary** keyword on the ABR configures the area as *Totally NSSA*; this is a Cisco proprietary feature. By default, the ABR does not inject a default route back into an NSSA area. Use the **default-information-originate** keyword on the ABR or ASBR to create this route:

```
Router(config-router)# area 7 nssa [no-summary]
[default-information-originate]
```

Virtual Links

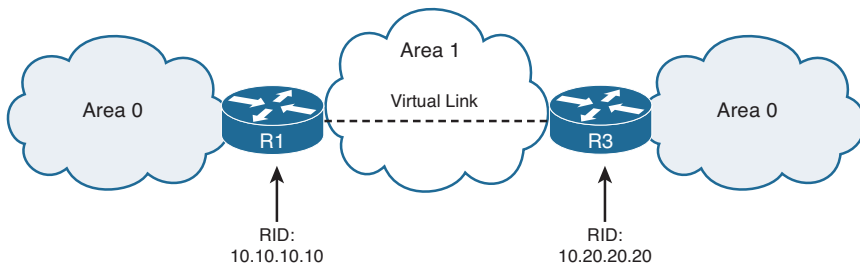
OSPF requires that all areas be connected to Area 0 and that Area 0 must be contiguous. When this is not possible, you can use a virtual link to bridge across an intermediate area. Virtual links provide the following:

- Connect areas that do not have a physical link to Area 0. (This should be a temporary solution.)

- Connect a discontinuous Area 0—when merging two company networks, for instance. (This should also be a temporary solution.)

Figure 5-2 shows a virtual link connecting two portions of the backbone Area 0.

Figure 5-2 OSPF Virtual Link



Area 1 is the transit area for the virtual link. Configure each end of a virtual link on the ABRs of the transit area with the command **area area-number virtual-link router-id**. Each end of the link is identified by its RID, *not* its interface IP address. The area listed in the command is the transit area, *not* the area being joined by the link. The configuration for router R1 is

```
R1(config)# router ospf 1
R1(config-router)# area 1 virtual-link 10.20.20.20
```

The configuration for router R3 is

```
R3(config)# router ospf 1
R3(config-router)# area 1 virtual-link 10.10.10.10
```

Verify whether the virtual link is up with the **show ip ospf virtual-links** command. Additionally, virtual interfaces are treated as actual interfaces by the OSPF process, and thus, their status can be verified with the **show ip ospf interface interface-id** command.

OSPF Authentication

By default, the router does no routing protocol authentication. For security purposes, you can configure OSPF to authenticate every OSPF packet and the source of every OSPF routing update. OSPF supports two types of authentication:

- Simple (plain text) authentication
- MD5 authentication

Example 5-2 shows a router configured for simple password authentication in OSPF Area 0, using a password (or *key*) of “simple.” Note that authentication commands are configurable under the OSPF process and the interface configuration. Use the OSPF process configuration mode to enable authentication for all interfaces within an area. You must still configure a password at the interface mode when you use this command. Use the interface configuration mode to enable authentication on a specific interface only. All OSPF neighbors reachable through an interface configured for authentication must use the same password. You can, however, use different passwords for different interfaces.

In Example 5-2, interface gi0/1, which is in Area 2, is also configured for OSPF authentication and a password of “simple.” This configuration is given at the interface mode.

Example 5-2 OSPF Simple Password Authentication

```
Router(config-if)# router ospf 1
Router(config-router)# area 0 authentication
!
Router(config)# int gi0/0
Router(config-if)# ip ospf authentication-key simple
!
Router(config)# int gi0/1
Router(config-if)# ip ospf authentication-key simple
Router(config-if)# ip ospf authentication
```

Example 5-3 shows the same router configured for OSPF MD5 authentication for Area 0, using a password of “secure.” Note that the commands are slightly different. The optional keyword **message-digest** is required in two of the commands, and a key number must be specified. Any neighbors reachable through the Gi0/0 interface must also be configured with the same key.

Interface gi0/1, which is in Area 2, is also configured for OSPF authentication and a password of “secure.” This configuration is given at the interface mode.

Example 5-3 OSPF MD5 Authentication

```
Router(config-if)# router ospf 1
Router(config-router)# area 0 authentication message-digest
!
Router(config-router)# int gi0/0
Router(config-if)# ip ospf message-digest-key 2 md5 secure
!
Router(config-router)# int gi0/1
```

```
Router(config-if)# ip ospf message-digest-key 2 md5 secure
Router(config-if)# ip ospf authentication message-digest
```

Use the following commands to verify and troubleshoot OSPF authentication:

- **debug ip ospf adj:** The debug shows an error message if there is a key mismatch.
- **show ip ospf neighbor:** If a neighbor relationship has been established, you can assume the authentication worked properly.
- **show ip route:** Verify whether route information is being exchanged between the two authenticating routers.

OSPF for IPv6

OSPFv3, which supports IPv6 routing, is a link-state routing protocol that uses the Dijkstra algorithm to select paths just like OSPFv2. Whereas OSPFv2 uses IPv4 as its transport, OSPFv3 uses IPv6. Neighbors peer using their link-local addresses.

OSPFv3 routers have the following characteristics:

- Use the same packet types as OSPFv2
- Form neighbors in the same way
- Flood and age LSAs identically
- Support the same topologies and techniques, such as NSSA and on-demand circuits

OSPFv3 differs from its predecessors principally in its new address format. OSPFv3 advertises using multicast addresses FF02::5 and FF02::6 but uses its link-local address as the source address of its advertisements. This means that OSPF can form adjacencies with neighbor routers that are not on the same global subnet. Multiple instances of OSPFv3 can run on each link. Authentication relies on the underlying IPsec capabilities of IPv6 but still requires configuration.

OSPFv3 configuration is similar to RIPng and EIGRP. The routing process is created and routing properties are assigned to it. As with EIGRP, the router ID must be in 32-bit dotted-decimal format, like an IPv4 address. If you do not manually set a router ID, then OSPF chooses one from IPv4 addresses currently active on the router. If there are no IPv4 addresses,

OSPF fails. Interfaces are associated with the OSPF process under interface configuration mode.

Interarea route summarizations still use the **area range** command. External route summarizations employ the command **summary-prefix**, a small change from OSPFv2.

OSPFv3 LSAs

OSPFv3 has renamed two of the LSAs and added two. Network Summary LSAs are now called *Interarea Prefix LSAs*. ASBR Summary LSAs are now called *Interarea Router LSAs*. A Link LSA and an Intra-area Prefix LSA have been added.

LSA type codes are now written in hexadecimal and include flooding scope information. If the first bit is 0, the LSA is only flooded on the local link—this is used by the new Link LSA. If it is 2, then it is flooded within an area—this is used by the majority of LSA types. The number 4 indicates that it is flooded domain-wide and is used by External LSAs. Table 5-2 shows the OSPFv3 LSAs.

Table 5-2 OSPFv3 LSAs

Type	Description
0x2001	Router LSA. Advertises the router's links. Generated by each OSPF router. Flooded only within the area.
0x2002	Network LSA. Advertises routers on a multiaccess link. Generated by the DR. Flooded only with the area.
0x2003	Interarea Prefix LSA. Advertises interarea routes. Generated by an ABR. Flooded to adjacent areas.
0x2004	Interarea Router LSA. Advertises the route to an ASBR. Generated by ABRs on behalf of an ASBR. Flooded to adjacent areas.
0x4005	AS-External LSA. Advertises routes in another routing domain. Generated by an ASBR. Flooded to adjacent areas. E1—Metric increased by each router E2—Metric does not increase (the default)
0x2006	Group Membership LSA.
0x2007	NSSA External LSA. Advertises routes in another routing domain. Generated by an ASBR in a not-so-stubby area. Flooded only within the NSSA area. N1—Metric increased by each router N2—Metric does not increase (the default)

Type	Description
0x0008	Link LSA. Advertises the link-local address and all prefixes associated with that link. Generated by each OSPF router. Flooded only on the local link.
0x2009	Intra-area Prefix LSA. Associates routes with a transit network link by referencing a Network LSA or associates routes with a router by referencing a Router LSA. Flooded only within the area.

Traditional OSPFv3 Configuration

There are two sets of commands for configuring OSPFv3. The traditional set includes the term “`ipv6`” as part of the command. If you are also using OSPFv2 for IPv4, that process runs separately from the IPv6 process and must be configured separately. It can run concurrently with OSPFv2 because each version maintains its own databases and runs a separate SPF calculation.

Assuming that **ipv6 unicast-routing** and interface IPv6 addresses are already in place, the commands to implement traditional OSPFv3 are shown in Example 5-4.

Example 5-4 Example Commands for OSPFv3

```
Router(config)# ipv6 router ospf process-id
Router(config-rtr)# router-id 32bit-address
!
Router(config-rtr)# interface type number
Router(config-if)# ipv6 ospf process-id area area
```

As illustrated in Example 5-5, route summarization is still configured under the OSPF routing process. Note that the router ID is still a 32-bit number, shown here in the form of an IPv4 address. Stub routing is also configured under the routing process, using the same commands as with OSPFv2. The default costs and interface priorities can be overridden at each interface. Example 5-5 shows how these commands might look on an actual router.

Example 5-5 Configuring Traditional IPv6 Routing

```
Router(config)# ipv6 unicast-routing
!
!Create the OSPFv3 process, assign router ID, create a summary
router, and configure Area 1 as a stub area
Router(config)# ipv6 router ospf 1
Router(config-rtr)# router-id 10.255.255.1
```

Example 5-5 Continued

```

Router(config-rtr)# area 1 range 2001:db8:1::/80
Router(config-rtr)# area 1 stub no summary
!
!Assign interface e0/0 to Area 1 with a cost of 10 and a DR priority
  of 20
Router(config-rtr)# interface e0/0
Router(config-if)# ipv6 address 2001:db8:1:1::2/64
Router(config-if)# ipv6 ospf 1 area 1
Router(config-if)# ipv6 ospf cost 10
Router(config-if)# ipv6 ospf priority 20
!
!Assign interface s1/0/0 to Area 0
Router(config-if)# interface serial 1/0/0
Router(config-if)# ipv6 address 2001:db8:1:5::1/64
Router(config-if)# ipv6 ospf 1 area 0

```

Troubleshooting OSPFv3 is similar to troubleshooting OSPFv2. Start by looking at **show ipv6 route** to verify whether routes have been advertised. Assuming the route is in the routing table, test reachability using **ping ipv6**. You can also look at the OSPF setup using **show ipv6 ospf process interface**, **show ipv6 ospf**, or **show ipv6 ospf database**.

New OSPFv3 Configuration

The newer method of OSPFv3 configuration runs both protocols in the same OSPF process and enables you to consolidate some of the configuration. Available starting in IOS release 15.1(3)S, it uses the term “ospfv3” in its configuration commands.

The following are some other features:

- A single OSPF process supports both IPv4 and IPv6 routing.
- This process uses a single OSPF database that has both IPv4 and IPv6 information.
- Routing is enabled under the interface for both IPv4 and IPv6—no more **network** commands.
- The **address-family** command is used for v4- or v6-specific configuration.
- Stub area configuration affects both v4 and v6.

Assuming that **ipv6 unicast-routing** and interface addresses are already in place, the commands to implement the new method of OSPFv3 are shown in Example 5-6. You do not actually have to explicitly create the routing process—it is created automatically when you enable OSPFv3 routing on an interface. However, it is useful when you need to add configuration under the process.

Example 5-6 Configuring IPv4 and IPv6 Routing Using OSPFv3

```
!Enable and configure OSPFv3 for both IPv4 and IPv6 routing:
Router(config)# router ospfv3 1
Router(config-router)# router-id 10.10.10.10
Router(config-router)# area 2 stub
!
Router(config-router)# address-family ipv4 unicast
Router(config-router-af)# passive-interface lo0
!
Router(config-router)# address-family ipv6 unicast
Router(config-router-af)# area 1 range 2001:db8:0:110::/60
!
!Enable OSPF for both IPv4 and IPv6 on the interfaces:
Router(config-if)# int e0/0
Router(config-if)# ospfv3 1 ipv4 area 0
Router(config-if)# ospfv3 1 ipv6 area 0
!
Router(config-if)# int e0/1
Router(config-if)# ospfv3 1 ipv4 area 2
Router(config-if)# ospfv3 1 ipv6 area 2
```

Verify many of the same commands as you would normally use with OSPF for IPv4 and IPv6, such as **show ip route** and **show ipv6 route**. However, to see both IPv4 and IPv6 neighbors, you must use **show ospfv3 neighbors**, and to see only IPv4 OSPF routes, use **show ip route ospfv3**. Use the command **show ospfv3 database** to see the combined v4 and v6 database. Example 5-7 shows the OSPFv3 neighbor database—note that it lists both IPv4 and IPv6 neighbors.

Example 5-7 The show ospfv3 neighbors Command

```
Router# show ospfv3 neighbor

OSPFv3 1 address-family ipv4 (router-id 10.10.10.10)
```

Example 5-7 Continued

Neighbor ID	Pri	State	Dead Time	Interface ID	Interface
4.4.4.4	1	FULL/DR	00:00:31	3	Ethernet0/0
3.3.3.3	1	FULL/BDR	00:00:31	4	Ethernet0/1

```
OSPFv3 1 address-family ipv6 (router-id 10.10.10.10)
```

Neighbor ID	Pri	State	Dead Time	Interface ID	Interface
4.4.4.4	1	FULL/DR	00:00:38	3	Ethernet0/0
3.3.3.3	1	FULL/BDR	00:00:39	4	Ethernet0/1

CHAPTER 6

Advanced Routing Techniques

There are times when you need to go beyond just turning on a routing protocol in your network. You might need to control exactly which routes are advertised or redistributed or which paths are chosen. You might also be required to use multiple routing protocols. Network performance can suffer when routing is not optimized. Excessive routing updates lead to extra CPU usage because of the amount of routing information and the frequency of updates. Running multiple protocols requires extra router resources and might result in suboptimal paths. Incorrectly configured route filters can lead to routing issues.

A good routing plan requires a sound design. Design IP addresses that can be summarized, and use techniques such as redistribution and passive interfaces appropriately.

Controlling Routing Updates

When a routing update arrives at a router's interface, the router checks to see whether a route filter is associated with that interface. If not, the update processes normally. If there is a filter, the router checks for an entry matching the update. If there is no matching entry, the update is dropped. If a matching entry exists, the router processes the update based on instructions in the filter.

Cisco IOS provides several ways to control routing updates:

- Route maps
- Prefix lists
- Distribute lists
- Passive interface

Route Maps

Route maps are a bit like programs that use an if/then/else decision-making capability. They match traffic against certain conditions and then set specified options for that traffic. Each statement has a sequence number, statements are read from the lowest number to highest, and the router stops reading when it gets a match. The sequence number can be used to insert or delete statements. Like an access list, there is an implicit “deny” at the end of each route map; any traffic not matched with a route map statement is denied. Some uses for route maps include:

- **Filtering redistributed routes:** Use the **route-map** keyword in the **redistribute** command.
- **Policy-based routing:** To specify which traffic should be policy routed, based on very granular controls.
- **Setting BGP policy:** To control routing updates and manipulate path attributes.

Route Map Syntax

Route maps are created with the global command:

```
Router(config)# route-map {tag} permit | deny [sequence_number]
```

Each statement in a route map begins this same way, with the same route map name but different sequence numbers, and with match and set conditions below it. *Permit* means that any traffic meeting the criteria of the match conditions is processed by the route map statement. *Deny* means that any traffic matching the match conditions is *not* processed by the route map statement; the router simply proceeds to the next route map statement.

Route Map Match and Set Conditions

Each route map statement can have zero to multiple **match** and **set** conditions. If no **match** condition exists, the statement matches anything, similar to a “permit any” in an access list. If there is no **set** condition, the matching traffic is either permitted or denied, with no other conditions being set.

Multiple match conditions on the same line use a logical OR. For example, the router interprets **match a b c** as “match a or b or c.” Multiple match conditions on different lines use a logical AND. For example, the router interprets the following route map statement as “match a and b and c”:

```
route-map Logical-AND permit 10
  match a
  match b
  match c
```

In route redistribution, some common conditions to match include

- **ip address:** Refers the router to an access list that permits or denies networks.
- **ip address prefix-list:** Refers the router to a prefix list that permits or denies IP prefixes.
- **ip next-hop:** Refers the router to an access list that permits or denies next-hop IP addresses.
- **ip route-source:** Refers the router to an access list that permits or denies advertising router IP addresses.
- **as-path:** Refers the router to an AS-path access list.
- **length:** Permits or denies packets based on their length in bytes.
- **metric:** Permits or denies routes with the specified metric from being redistributed.
- **route-type:** Permits or denies redistribution of the route type listed, such as internal or external.
- **tag:** Routes can be labeled (tagged) with a number, and route maps can look for that number.

In route redistribution, some common conditions to set include

- **metric:** Sets the metric for redistributed routes
- **metric-type:** Sets the type, such as E1 for OSPF
- **tag:** Tags a route with a number that can be matched later by other route maps

Controlling Route Redistribution Using Route Maps

Example 6-1 shows a route map named BGP-LP with three statements that control which routes to redistribute from OSPF into BGP. The router has already been configured with two access lists, numbered 23 and 103 (not shown). The first route map statement, with sequence number 10, is a *permit*

statement. The **match** condition orders it to use access list 23. Any traffic permitted by access list 23 matches this statement and is redistributed into BGP. Any traffic explicitly denied by access list 23 is not redistributed into BGP. The **set** condition tells it to set a BGP local preference of 200 for all traffic that matches statement 10. Traffic not matching access list 23 is checked against the second route map statement.

The second route map statement, sequence number 20, is a *deny* statement that matches access list 103. Any traffic permitted by access list 103 is denied by this statement and thus will not be redistributed. Any traffic explicitly denied by access list 103 is ignored by this statement and checked against the next route map statement. This route map statement has no **set** conditions. Traffic not matching route map statements 10 or 20 is checked against statement 30.

The third route map statement, sequence number 30, is a *permit* statement with no **match** or **set** conditions. This statement matches everything and sets nothing, thus permitting all other traffic without changing it. Without this statement, all other traffic is denied.

Finally, the route map is applied to the redistribution command to filter routes redistributed from OSPF into BGP.

Example 6-1 Controlling Redistribution Using a Route Map

```
Router(config)# route-map BGP-LP permit 10
Router(config-route-map)# match ip address 23
Router(config-route-map)# set local-preference 200
Router(config-route-map)# !
Router(config-route-map)# route-map BGP-LP deny 20
Router(config-route-map)# match ip address 103
Router(config-route-map)# !
Router(config-route-map)# route-map BGP-LP permit 30
!
Router(config)# router bgp 65001
Router(config-router)# redistribute ospf 1 route-map BGP-LP
```

Tagging Routes Using a Route Map

Another use for a route map is to tag routes as they are redistributed from one protocol to another. You can then deny tagged routes from being redistributed back into the original protocol. For example, suppose you are mutually redistributing routes between OSPF and EIGRP. You can tag EIGRP routes as you redistribute them into OSPF. Then when you redistribute OSPF

routes back into EIGRP, you can deny those tagged routes. Example 6-2 shows the commands to accomplish this.

Example 6-2 Tagging Redistributed Routes

```
Router(config)# route-map EIGRP2OSPF deny 5
Router(config-route-map)# match tag 1
Router(config-route-map)# route-map EIGRP2OSPF permit 10
Router(config-route-map)# set tag 2
!
Router(config)# route-map OSPF2EIGRP deny 5
Router(config-route-map)# match tag 2
!
Router(config-route-map)# route-map OSPF2EIGRP permit 10
Router(config-route-map)# set tag 1
!
Router(config)# router eigrp 1
Router(config-router)# redistribute ospf 2 route-map OSPF2EIGRP
metric 1 1 1 1 1500
!
Router(config-router)# router ospf 2
Router(config-router)# redistribute eigrp 1 route-map EIGRP2OSPF
subnets
```

Prefix Lists

A prefix list matches both the subnet, or *prefix*, and the number of bits in the subnet mask. Similar to an access list, it consists of one or more statements permitting or denying prefixes. Routers evaluate the prefix statements in order, stopping if they find a match. There is an implicit “deny all” at the end of the prefix list. The command syntax is as follows:

```
ip prefix-list {list-name [seq-number] {permit | deny} network/
length [ge ge-length] [le le-length]
```

The meaning of each command field is detailed in Table 6-1.

Table 6-1 ip prefix-list Command

Command Field	Meaning
<i>list-name</i>	Gives a name to the prefix list. Prefix lists are named, not numbered.
<i>seq-number</i>	[Optional] Assigns a sequence number to the prefix list statement. Statements are numbered in increments of 5 by default, enabling a statement to be inserted between two others by using the seq option.

Table 6-1 Continued

Command Field	Meaning
deny permit	Denies or permits the matching prefix.
<i>network/length</i>	Configures the prefix and number of bits that must be matched. If no ge or le option is given, the length also equals the length of the subnet mask.
ge <i>ge-length</i>	[Optional] Stands for “greater than or equal to.” Specifies the minimum number of bits a subnet mask must have to match the statement.
le <i>le-length</i>	[Optional] Stands for “lesser than or equal to.” Specifies the maximum number of bits a subnet mask can have to match the statement.

Some sample prefix lists include

- **ip prefix-list CCNP permit 0.0.0.0/0:** Permits only a default route.
- **ip prefix-list CCNP permit 0.0.0.0/0 le 32:** Permits all routes (equivalent to a “permit any” in an access list). The prefix 0.0.0.0/0 means that none of the prefix bits must be matched. “Le 32” means that the subnet mask must be less than or equal to 32. Thus, any network will match this statement.
- **ip prefix-list CCNP permit 0.0.0.0/0 ge 32:** Permits only host routes. The prefix 0.0.0.0/0 means that none of the prefix bits must be matched. “Ge 32” means that the subnet mask must be exactly 32 bits, thus this statement matches only host routes.
- **ip prefix-list CCNP permit 10.0.0.0/8 ge 24 le 24:** Permits any route whose first 8 bits equal 10, with a subnet mask of exactly 24 bits.

Prefix lists can also be used within a route map to control redistribution of networks. They can also be applied to a BGP neighbor to filter routing updates to that neighbor.

Distribute Lists

A distribute list enables you to filter both routing updates and routes being redistributed, through the use of an access list. Configure an access list that permits the routes to be advertised or redistributed, and then link that access list to the routing process with the **distribute-list** command, given under router configuration mode. This command has two options:

- **distribute-list *access-list* in:** Filters updates as they come into an interface. For OSPF, this controls routes placed in the routing table but not the database. For other protocols, this controls the routes the protocol knows about.
- **distribute-list *access-list* out:** Filters updates going out of an interface and updates being redistributed out of another routing protocol into this one.

Passive Interfaces

Using the **passive-interface** command is another way to control routing updates because it prevents any updates from sending out an interface that is marked as passive. OSPF and EIGRP do not send hello messages out a passive interface, and thus do not discover any neighbors. RIP does not send updates out a passive interface but listens for inbound updates. Chapter 4, “EIGRP,” and Chapter 5, “OSPF,” have more in-depth descriptions of this command.

Using Multiple Routing Protocols

There are several reasons you might need to run multiple routing protocols in your network:

- Migrating from one routing protocol to another, where both protocols will run in the network temporarily
- Applications that run under certain routing protocols but not others
- Areas of the network under different administrative control
- A multivendor environment where not all devices in the network support the desired protocol

Configuring Route Redistribution

Route redistribution is used when routing information must be exchanged among the different protocols or routing domains. Only routes that are in the routing table and learned via the specified protocol are redistributed. Each protocol has some unique characteristics when redistributing, as shown in Table 6-2.

Table 6-2 Route Redistribution Characteristics

Protocol	Redistribution Characteristics
RIP	Default metric is Infinity. Metric must be set, except when redistributing static or connected routes, which have a metric of 1.
OSPF	Default metric is 20. Can specify the metric type; the default is E2. Must use subnets keyword or only classful networks are redistributed.
EIGRP	Default metric is Infinity. Metric must be set, except when redistributing static or connected routes, which get their metric from the interface. Metric value is “bandwidth, delay, reliability, load, MTU.” Redistributed routes have a higher administrative distance than internal ones.
Static/Connected	To include local networks not running the routing protocol, you must redistribute connected interfaces. You can also redistribute static routes into a dynamic protocol.
BGP	Metric (MED) is set to IGP metric value.

You can redistribute only between protocols that use the same protocol stack, such as IP. To configure redistribution, issue this command under the routing process that is to receive the new routes:

```
Router(config-router)# redistribute {route-source} [metric metric]
[route-map tag]
```

The IPv6 commands to redistribute routes between protocols or between multiple instances of a protocol are just like the ones in IPv4. Under the routing protocol configuration mode, issue the command **redistribute route-source** and specify any options, such as a route map if desired.

Seed Metric

Redistribution involves configuring a routing protocol to advertise routes learned by another routing process. Normally, protocols base their metric on an interface value, such as bandwidth, but a redistributed route is not associated with an interface. Protocols use incompatible metrics, so the redistributed routes must be assigned a new metric compatible with the new protocol.

A route’s starting metric is called its *seed metric*. Set the seed metric for all redistributed routes with the **default-metric {metric}** command under the routing process. To set the metric for specific routes, either use the **metric** keyword within the redistribution command or use the **route-map** keyword to link a route map to the redistribution. After the seed metric is specified, it

increments normally as the route is advertised through the network (except for certain OSPF routes).

Administrative Distance

Recall that when a router receives routes to the same destination network from more than one routing process, it decides which to put in the routing table by looking at the administrative distance (AD) value assigned to the routing process. The route with the lowest AD is chosen.

The AD can be changed for all routes of a process or only for specific routes within a process. The command for all IGP's except EIGRP is

```
Router(config-router)# distance administrative_distance {address
wildcard-mask} [access-list-number | name]
```

Using the *address/mask* options in the command changes the AD of routes learned from the neighbor with that IP address. An entry of **0.0.0.0 255.255.255.255** changes the AD of all routes. Specifying an access list number or name changes the AD only on networks permitted in the ACL.

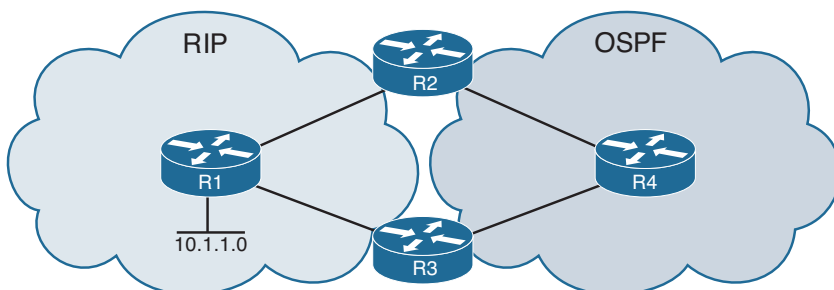
EIGRP and BGP have different AD values for internal and external routes, so you must list those separately when using the command with those protocols. BGP also enables you to change the AD for locally generated routes. For these protocols, the commands are

```
Router(config-router)# distance eigrp internal-distance external-
distance
```

```
Router(config-router)# distance bgp external-distance internal-
distance local-distance
```

Route redistribution can cause suboptimal routing; one way to correct this is to adjust the AD for redistributed routes. Figure 6-1 shows a network with two routing domains: RIP and OSPF.

Figure 6-1 Controlling Routing with AD



R2 redistributes its RIP routes into OSPF. These routes inherit OSPF's AD when they are advertised to R4, which then advertises them to R3 as OSPF routes.

R3 now recognizes the 10.1.1.0 network from two routing processes: RIP, with an AD of 120, and OSPF, with an AD of 110. The shortest path is the RIP route through R1. The OSPF path goes through R4 and R2, and then to R1—a much longer path. But, based on AD, R3 puts the OSPF path in its routing table.

To prevent this, increase the AD of the redistributed RIP routes when OSPF advertises them. Note that this doesn't change all OSPF routes, just the ones learned from RIP. The commands given on R2 (the router doing the initial redistribution) are shown in Example 6-3.

Example 6-3 Adjusting Administrative Distance

```
R2(config)# access-list 10 permit 10.1.1.0
!
R2(config)# router ospf 1
R2(config-router)# redistribute rip subnets
R2(config-router)# distance 125 0.0.0.0 255.255.255.255 10
```

The AD is increased to 125 for routes from all neighbors if they match the network permitted in access list 10. Now R3 hears about the 10.1.1.0 network from RIP with an AD of 120 and from OSPF with an AD of 125. The RIP route is put into the routing table based on its lower AD.

Routing protocols that assign a higher AD to external routes, EIGRP and BGP, accomplish a similar result automatically. OSPF can be configured to do so with the **distance ospf external** command.

Planning Route Redistribution

Plan carefully before redistributing routes between protocols. Different protocols have incompatible routing information and unique convergence times. First, decide which is the core or main protocol and which is the edge protocol. Decide whether you require one-way or two-way, and single-point or multipoint redistribution.

One-way redistribution involves redistributing routes from the edge routing protocol into the core protocol. Static or default routes are used in the edge protocol. Two-way redistribution involves redistributing routes mutually between both core and edge protocols. No static routes are needed because both protocols have information about all routes.

One-way and two-way redistribution at just one router within the network is considered safe because traffic between administrative domains has only one exit point, thus routing loops are not a problem. Redistribution at multiple routers within the network can cause routing loops and suboptimal routing.

With multipoint one-way redistribution, the following actions are suggested:

- Use a routing protocol that uses different ADs for external and internal routes (EIGRP, OSPF, and BGP).
- Ensure that the AD of the redistributed external routes is higher than the AD of the protocol where they originated.

Multipoint two-way redistribution adds the following considerations:

- Ensure that only internal routes are redistributed from each protocol. You can do this by tagging the routes and then filtering based on tags when redistributing.
- Adjust the metric of the redistributed routes.
- Consider using a default route to avoid multipoint two-way redistribution.

Redistribution Techniques

Try to design your route redistribution as safely as possible. The options include

- Redistribute all edge information into the core, but send only a default route into the edge.
- Redistribute all edge information into the core, but redistribute multiple static routes into the edge.
- Redistribute routes in both directions, but filter to prevent routes from being redistributed back into their original administrative domain.
- Redistribute all routes in both directions, but increase the AD for external routes.

There are several points to remember when configuring route redistribution:

- A router redistributes only routes learned by the source protocol. When redistributing routes into BGP, use the keyword **include-connected** to get the connected routes into BGP.

- When you redistribute routes between two OSPF processes, the routes are advertised into the new process as Type 2.
- You generally want to include the **subnets** keyword on routes distributed from another routing protocol into OSPF; otherwise, only routes that use their default classful subnet mask are redistributed.
- Be sure to specify a seed metric when redistributing routes into RIP; otherwise, the routes start with a metric of 16, which RIP interprets as “unreachable.”
- If you redistribute in multiple places, check the resulting traffic path because you might run into suboptimal routing. One solution is to tune the administrative distance for some of the routes.
- BGP does not redistribute routes learned via iBGP into an IGP by default. To change this behavior, use the router configuration command **bgp redistribute-internal**.

Path Control

In general, link redundancy is desirable, but it can also lead to some network problems. You might want to manually control the route taken by some or all of your traffic to provide a predictable and deterministic traffic flow. Path control can provide the following:

- Prevent suboptimal routing
- Ensure path availability
- Provide optimized performance for specific applications
- Provide load sharing among various paths

A good path control strategy understands that traffic is bidirectional and considers both inbound and outbound traffic. Asymmetric routing—where traffic exits via one link and enters via another—is not inherently bad. But you might need to minimize it when using stateful devices such as firewalls, or with sensitive applications such as voice.

Policy-Based Routing

Policy-based routing (PBR) overrides the normal routing process. Normal routing is performed based on the destination IP address. Policy-based routing is based on source IP address or interface or packet length. It uses a route-map to match traffic and then sets either a next-hop address or an exit interface. It can also mark the traffic that it policy routes. Any traffic not matched in the route map is routed normally. Policy-based routing can be applied both to traffic entering the router and to traffic originated by the router.

Some benefits of policy routing include

- The capability to route based on traffic source and other attributes
- The capability to set QoS markings
- The capability to force load sharing between unequal paths
- The capability to allocate traffic among multiple paths based on traffic attributes

Use the following steps to configure policy-based routing:

- Step 1.** Configure a route-map that matches the desired traffic and uses the **set** command to define the actions for that traffic.
- Step 2.** Apply the route map either to an incoming interface or to traffic generated by the router.
- Step 3.** Verify the configuration.

Some typical attributes to match in the route map include source or destination address with an access list and packet length. If no match criteria are specified, all packets are considered a match.

You can choose to set various options, but the most typical settings determine how the traffic leaves the router. The four most common ways to accomplish this are listed in Table 6-3.

Table 6-3 Policy-Based Routing Options

Command	Description
set ip next-hop <i>ip-address</i>	When this command is given, the router checks to see whether the next-hop address is reachable. If so, it forwards the traffic toward that address. If not, it uses the routing table.

Table 6-3 Continued

Command	Description
set interface <i>interface-type</i> <i>interface-number</i>	Multiple interfaces can be listed. When this command is given, the router checks that it has an explicit route for the destination network in its routing table before forwarding the traffic out the specified interface. If it does not, this command is ignored. A default route is not considered an explicit route. Listing multiple interfaces under the set command allows for redundancy if the first interface fails or goes down. The router uses the first active interface listed.
set ip default next-hop <i>ip-address</i> [verify-availability]	If the routing table contains an explicit route for the destination network, that route is used and this command is ignored. If no explicit route exists, this command is executed. A default route is not considered an explicit route. The verify-availability option checks the CDP database to ensure the next hop is available.
set default interface <i>interface-type</i> <i>interface-number</i>	If the routing table contains an explicit route for the destination network, that route is used and this command is ignored. If no explicit route exists, traffic is forwarded out the specified interface. A default route is not considered an explicit route.

To apply the PBR route-map to an interface, use the interface command **ip policy route-map** *name*. To cause PBR to act on packets originated by the router itself, use the global command **ip local policy route-map** *name*.

Example 6-4 shows a route map named LOCAL that matches the source addresses in access list 1. It assigns a next-hop IP address of 10.1.1.1 to this traffic. Because it is applied to the local router, it is used only for traffic generated by the router itself.

Example 6-4 Applying PBR to Local Router Traffic

```
Router(config)# route-map LOCAL
Router(config-route-map)# match ip address 1
Router(config-route-map)# set ip next-hop 10.1.1.1
!
Router(config)# ip local policy route-map LOCAL
```

Example 6-5 shows a PBR route map that matches traffic in access list 101 and policy routes it to a next hop of 10.1.1.1. The policy is applied to interface Ethernet0/0, so all traffic entering that interface is evaluated against the route-map. Packets not permitted by access list 101 are routed normally (destination routed.)

Example 6-5 Applying PBR to Interface Traffic

```
Router(config)# route-map Policy permit 10
Router(config-route-map)# match ip address 101
Router(config-route-map)# set ip next hop 10.1.1.1
!
Router(config)# interface Ethernet0/0
Router(config-if)# ip policy route-map Policy
```

PBR can also use IP SLA tracking, described in the following IP SLA section. Verify your configuration with the commands **show ip policy** and **show route-map {name}**. Use the **debug ip policy** command to see the policy routing in action.

Using IOS IP SLA

IP Service Level Agreement (IP SLA) is a feature that enables a Cisco router or switch to simulate specific types of traffic and send it either to an IP address or to a receiver, called a *responder*. IP SLA probes can simulate various types of traffic, such as

- HTTP
- FTP
- DHCP
- UDP jitter
- UDP echo
- HTTP
- TCP connect
- ICMP echo
- ICMP path echo
- ICMP path jitter
- DNS

The probes can also report statistics such as path jitter. IP SLA has highly granular application configuration options, such as TCP/UDP port numbers, TOS byte, and IP prefix bits. With IP SLA you can measure network performance and host reachability. This is useful for path control because it enables you to switch to a backup path if network performance on the primary path degrades, or if a link failure occurs someplace in the primary path.

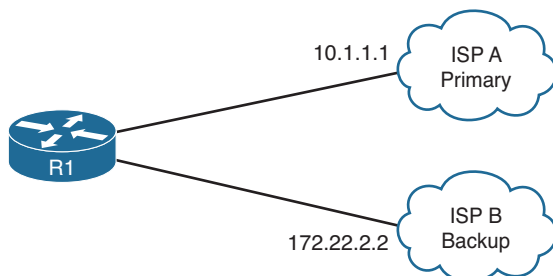
To use IP SLA for path control, you must follow these steps:

- Step 1.** Create a monitor session on the probe source device.
- Step 2.** Define the probe by specifying traffic type, destination IP address, and any other desired variables, such as Differentiated Services Code Point (DSCP) value.
- Step 3.** Schedule the probe beginning and ending times.
- Step 4.** Define a tracking object that is linked to the monitor session.
- Step 5.** Link the tracking object to a static route.

The destination can be any trusted device that responds to the traffic you send. To use IP SLA for bringing up a backup link, choose a destination that will reflect problems in the ISP's network. One benefit of using a Cisco device as the responder is that it can add time stamps to help measure latency and jitter. These time stamps take into account the device processing time so that the measurement reflects only network latency. The configuration of a Cisco responder is simple. Use the global command **ip sla responder**.

Figure 6-2 shows a network with two connections to the Internet, one primary and a backup. The edge router has a static default route pointing to each provider. Dynamic failover between the primary and backup path is desired.

Figure 6-2 Using IP SLA for Path Control



In Example 6-6, router R1 is configured to conditionally announce a default route based on the IP SLA probe response. Two IP SLA monitor sessions are configured to send a ping every 10 seconds to a DNS server within each ISP's network. Separate tracking objects verify reachability to each DNS server. The two default route statements notify the router to give the primary route an administrative distance of 2 if the tracked object is reachable. The backup route is assigned an administrative distance of 3 if its tracked object is reachable.

Example 6-6 Using IP SLA

```
R1(config)# ip sla 1
R1(config-ip-sla)# icmp-echo 10.1.1.50 source-interface e0/0
R1(config-ip-sla-echo)# frequency 10
!
R1(config)# ip sla 2
R1(config-ip-sla)# icmp-echo 171.22.2.52 source-interface e0/1
R1(config-ip-sla-echo)# frequency 10
!
R1(config)# ip sla schedule 1 life forever start-time now
R1(config)# ip sla schedule 2 life forever start-time now
!
R1(config)# track 1 ip sla 1 reachability
R1(config)# track 2 ip sla 2 reachability
!
!Primary route
R1(config)# ip route 0.0.0.0 0.0.0.0 10.1.1.1 2 track 1
!Backup route
R1(config)# ip route 0.0.0.0 0.0.0.0 172.22.2.2 3 track 2
```

Under normal circumstances, the default route with an AD of 2 is installed in the IP routing table instead of the one with an AD of 3. But when the primary DNS server is not reachable, the primary route is withdrawn and the backup route with an AD of 3 is installed in the routing table.

Additionally, you can combine IP SLA tracking with policy-based routing to provide redundancy for specific traffic types on a per-interface basis. For instance, after an access list permitting the interesting traffic is specified, a route-map such as the one shown in Example 6-7 can be configured. In this sample route-map, x.x.x.x and y.y.y.y represent different next hops for the traffic specified in ACL 101. The **track 1** and **track 2** keywords tie the configuration back to the SLA groups configured in Example 6-6. You then apply the policy map under the incoming interfaces, just as with normal policy routing.

Example 6-7 Using a Route Map with IP SLA

```

route-map REDUNDANT permit 10
  match ip address 101
  set ip next-hop verify-availability x.x.x.x 10 track 1
  set ip next hop verify-availability y.y.y.y 20 track 2
!

interface Ethernet0/1
  description LAN Interface
  ip policy route-map REDUNDANT

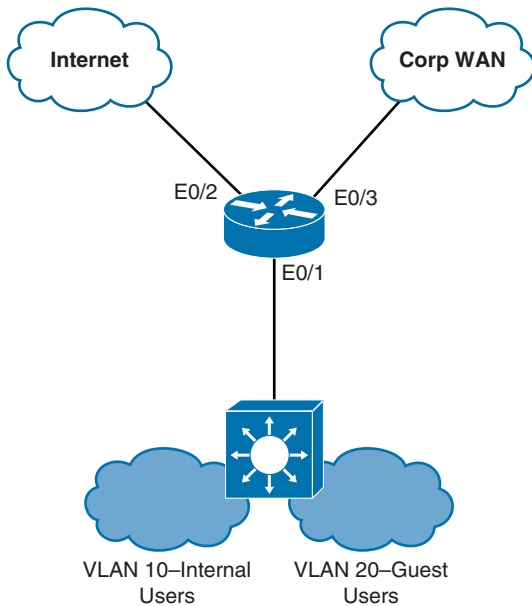
```

VRF-Lite

Virtual routing and forwarding (VRF) is a way to segment traffic and control the path it takes. You might think of it as a Layer 3 VLAN. Just as VLANs create virtual switches with segregated CAM tables, VRFs create virtual routers with segregated routing tables. This enables you to separate guest traffic from employee traffic, for example, and route it over a different path. We usually think of VRFs in association with MPLS, but VRF-Lite lets you create VRFs without using MPLS.

To configure a router for this design, create the necessary VRFs and apply each one to the interfaces to carry traffic for that particular VRF. For example, Figure 6-3 shows a simple network with a switch connected to a router. The router then has two external connections—one to the Internet and the other to the corporate WAN. Internal users' traffic must be separate from guest user traffic and routed to the corporate WAN. Guest users are routed to the Internet. The switch has two VLANs—one for internal users and another for guest users.

In Figure 6-3, two VRFs are created on the router to separate the two types of traffic: INT for internal users and GUEST for guest users. A static default route is created for each VRF. The link between the switch and the router is a trunk, with a subinterface for each of the VRFs. Each subinterface is associated with the appropriate VRF using the command **ip vrf forwarding vrf-name**, as are the interfaces to the WAN and to the Internet. Example 6-8 shows this configuration.

Figure 6-3 VRF-Lite**Example 6-8 VRF-Lite Configuration**

```

R1(config)# ip vrf INT
R1(config-vrf)# description Internal Traffic
!
R1(config-vrf)# ip vrf GUEST
R1(config-vrf)# description Guest Traffic
!
R1(config)# ip route vrf INT 0.0.0.0 0.0.0.0 209.165.201.2
R1(config)# ip route vrf GUEST 0.0.0.0 0.0.0.0 209.165.202.2
!
R1(config)# int e0/1
R1(config-if)# description To SW1
R1(config-if)# int e0/1.10
R1(config-subif)# encapsulation dot1Q 10
R1(config-subif)# ip vrf forwarding INT
R1(config-subif)# ip address 192.168.10.1 255.255.255.0
!
R1(config-subif)# int e0/1.20
R1(config-subif)# encapsulation dot1Q 20
R1(config-subif)# ip vrf forwarding GUEST
R1(config-subif)# ip address 192.168.20.1 255.255.255.0
!

```

Example 6-8 Continued

```

R1(config)# int e0/2
R1(config-if)# description To Internet
R1(config-if)# ip vrf forwarding GUEST
R1(config-if)# ip address 209.165.202.1 255.255.255.252
!
R1(config)# int e0/3
R1(config-if)# description To WAN
R1(config-if)# ip vrf forwarding INT
R1(config-if)# ip address 209.165.201.1 255.255.255.252

```

Note that the IP address is applied to the interface *after* it is associated with a VRF. If you configure the IP address prior to that, the router removes it when you apply the VRF configuration.

With this configuration, when traffic from VLAN 10 enters the router it is placed into VRF INT and routed to the WAN. The router places traffic from VLAN 20 into VRF GUEST and routes it to the Internet.

To verify your configuration, use the command **show vrf** to check the configured VRFs:

```

R1# show vrf

```

Name	Default RD	Protocols
Interfaces		
GUEST	<not set>	ipv4
Et0/1.20		
INT	<not set>	ipv4
Et0/1.10		

To view the routes for a particular VRF, append the VRF name to the **show ip route** command. Note that only the networks belonging to the “INT” VRF show up in Example 6-9.

Example 6-9 The show ip route vrf Command

```

R1# show ip route vrf INT

```

Routing Table: INT

Codes: L - local, 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, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is 209.165.201.2 to network 0.0.0.0

```
S* 0.0.0.0/0 [1/0] via 209.165.201.2
    192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
C    192.168.10.0/24 is directly connected, Ethernet0/1.10
L    192.168.10.1/32 is directly connected, Ethernet0/1.10
    209.165.201.0/24 is variably subnetted, 2 subnets, 2 masks
C    209.165.201.0/30 is directly connected, Ethernet0/3
L    209.165.201.1/32 is directly connected, Ethernet0/3
```

CHAPTER 7

BGP and Internet Connectivity

Planning an Internet Connection

Consider your company's needs when planning your Internet connection. What link type and bandwidth will support all the external connections plus your internal users? Is the company considering any hosted or cloud services that would drive additional bandwidth needs? Will you use static or dynamic routing? How much redundancy will you require, including both link and ISP redundancy?

Types of ISP Connections

A site with a single Internet service provider (ISP) connection is *single homed*, which is sufficient for a site that does not depend heavily on Internet or WAN connectivity. Either use static routes or advertise the site routes to the ISP and receive a default route from the ISP.

A *dual-homed* site has two connections to the same ISP, either from one or two routers. One link might be primary and the other backup, or the site might load balance over both links. Either static or dynamic routing works in this case.

Multihoming means connecting to more than one ISP at the same time. It is done for redundancy and backup in case one ISP fails, and for better performance if one ISP provides a better path to frequently used networks. This also gives you an ISP-independent solution. Border Gateway Protocol (BGP) is typically used with multihomed connections.

You can take multihoming a step further and be *dual-multihomed*, with two connections to multiple ISPs. This gives the most redundancy. BGP is used with dual-multihomed connections.

IP Addressing and AS Numbering

You can choose between provider-assigned and provider-independent IP addressing. If you are connecting to only one ISP, you can use IP addresses owned by that provider. You will either have one edge address or a pool, and use NAT for the inside private addresses. The provider is responsible for advertising your addresses to the Internet.

If your company is multihomed, you will likely acquire your own provider-independent IP address block. IP address allocation is coordinated by Regional Internet Registries (RIR) under the auspices of the Internet Assigned Numbers Authority (IANA). With provider-independent space, you are responsible for the way your routes are advertised to the Internet, giving you more control over routing policies.

To run BGP with an ISP, you need an autonomous system (AS) number. Autonomous systems are explained in the “BGP Overview” section of this chapter. If you are using provider-assigned IP addresses, you will likely use a provider-assigned private AS number. If you plan to use provider-independent addresses then you will need your own public AS number.

BGP Route Options

There are three ways to receive BGP routes from an ISP:

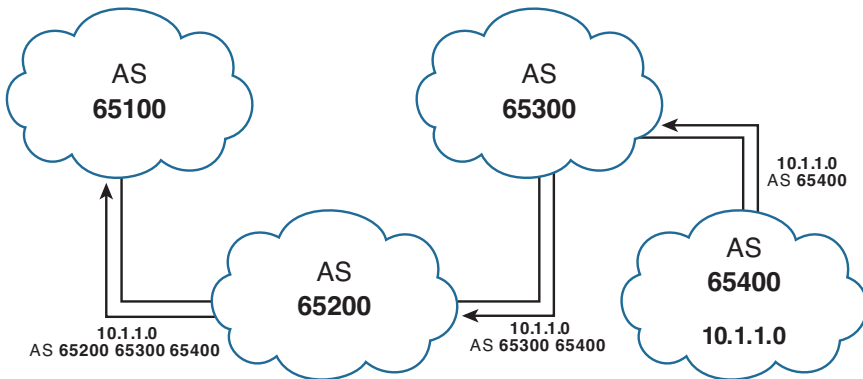
- **Default routes from each provider:** This is simple to configure and results in low use of bandwidth and router resources. The internal network’s IGP metric determines the exit router for all traffic bound outside the autonomous system. No BGP path manipulation is possible, so this can lead to suboptimal routing if you use more than one ISP.
- **Default routes plus some more specific routes:** This option results in medium use of bandwidth and router resources. It enables you to manipulate the exit path for specific routes using BGP so that traffic takes a shorter path to networks in each ISP. Thus, path selection is more predictable. The IGP metric chooses the exit path for default routes.
- **All routes from all providers:** This requires the highest use of bandwidth and router resources. It is typically used by large enterprises and ISPs. Path selection for all external routes can be controlled via BGP policy routing tools.

BGP Overview

BGP is an external gateway protocol, meant to be used between different network entities. It is the protocol used between ISPs and also can be used between an Enterprise and an ISP. BGP was built for reliability, scalability, and control, not speed. Because of this, it behaves differently from the protocols covered so far in this book. Some BGP characteristics are as follows:

- BGP uses the concept of autonomous systems. An *autonomous system* is a group of networks under a common administration. The IANA assigns AS numbers, which are traditionally 16 bits: 1 to 64,511 are public AS numbers and 64,512 to 65,535 are private AS numbers. Since 2007, a 32-bit format can be used: private AS numbers are 4,200,000,000–4,294,967,294 in this format.
- Autonomous systems run Interior Gateway Protocols (IGP) within the system. They run an Exterior Gateway Protocol (EGP) between them. BGP is the only EGP currently in use.
- BGP is a path-vector protocol. Its route to a network consists of a list of autonomous systems on the path to that network.
- Routing between autonomous systems is called *interdomain routing*.
- Routers running BGP are termed *BGP speakers*.
- BGP neighbors are called *peers* and must be statically configured.
- BGP routing with a peer in a different AS is called *external BGP* (*eBGP*). With a peer in the same AS it is called *internal BGP* (*iBGP*).
- The administrative distance for eBGP routes is 20. The administrative distance for iBGP routes is 200.
- BGP uses TCP port 179. BGP peers exchange incremental, triggered route updates and periodic keepalives.
- Routers can run only one BGP process at a time. Switches with Virtual Device Contexts (VDC) can run one BGP process per VDC. BGP's loop prevention mechanism is an autonomous system number. When an update regarding a network leaves an autonomous system, that AS's number is prepended to the list of autonomous systems that have handled that update. When a BGP router receives an update, it examines the AS list. If it finds its own autonomous system number in that list, the update is discarded.

In Figure 7-1, BGP routers in AS 65100 see network 10.1.1.0 as having an autonomous system path of 65200 65300 65400.

Figure 7-1 BGP AS-Path Advertisement

Use BGP when the AS is multihomed, when route path manipulation is needed, or when the AS is a transit AS. (Traffic flows through it to another AS, such as with an ISP.) Do not use BGP in a single-homed AS with a router that does not have sufficient resources to handle it or with a staff that does not have a good understanding of BGP path selection and manipulation.

BGP Databases

BGP uses three databases. The first two listed are BGP-specific; the third is shared by all routing processes on the router.

- **Neighbor database:** A list of all configured BGP neighbors. To view it, use the **show ip bgp summary** command.
- **BGP database:** A list of networks known by BGP, along with their paths and attributes. To view it, use the **show ip bgp** command.
- **Routing table:** A list of the paths to each network used by the router, and the next hop for each network. To view it, use the **show ip route** command.

BGP Message Types

BGP has four types of messages:

- **Open:** After a neighbor is configured, BGP sends an open message to try to establish peering with that neighbor. Includes information such as autonomous system number, router ID, and hold time.
- **Update:** Message used to transfer routing information between peers. Includes new routes, withdrawn routes, and path attributes.

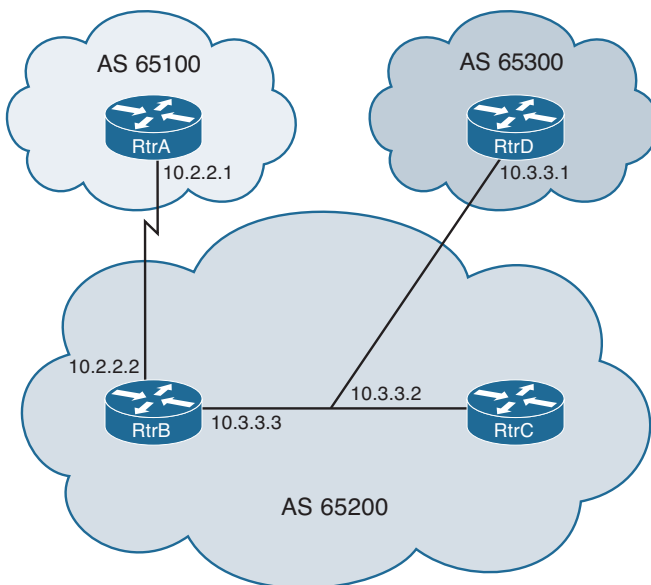
- **Keepalive:** BGP peers exchange keepalive messages every 60 seconds by default. These keep the peering session active.
- **Notification:** When a problem occurs that causes a router to end the BGP peering session, a notification message is sent to the BGP neighbor and the connection is closed.

BGP Next-Hop Selection

The next hop for a route received from an eBGP neighbor is the IP address of the neighbor that sent the update.

When a BGP router receives an update from an eBGP neighbor, it must pass that update to its iBGP neighbors without changing the next-hop attribute. The next-hop IP address is the IP address of an edge router belonging to the next-hop autonomous system. Therefore, iBGP routers must have a route to the network connecting their autonomous system to that edge router. For example, in Figure 7-2, RtrA sends an update to RtrB, listing a next hop of 10.2.2.1, its serial interface. When RtrB forwards that update to RtrC, the next-hop IP address remains 10.2.2.1. RtrC must have a route to the 10.2.2.0 network to have a valid next hop.

Figure 7-2 BGP Next-Hop Behavior



To change this behavior, use the **neighbor** *{ip address | peer-group-name}* **next-hop-self** command in BGP configuration mode. In Figure 7-2, this

configuration goes on RtrB. After you give this command, RtrB advertises its IP address to RtrC as the next hop for networks from AS 65100 rather than the address of RtrA. Thus, RtrC does not have to know about the external network between RtrA and RtrB (network 10.2.2.0).

BGP Next Hop on a Multiaccess Network

On a multiaccess network, BGP can adjust the next-hop attribute to avoid an extra hop. In Figure 7-2, RtrC and RtrD are eBGP peers, and RtrC is an iBGP peer with RtrB. When RtrC sends an update to RtrD about network 10.2.2.0, it normally gives its interface IP address as the next hop for RtrD to use. But because RtrB, RtrC, and RtrD are all on the same multiaccess network, it is inefficient for RtrD to send traffic to RtrC and RtrC to then send it on to RtrB. This process unnecessarily adds an extra hop to the path. So, by default, RtrC advertises a next hop of 10.3.3.3 (RtrB's interface) for the 10.2.2.0 network. This behavior can also be adjusted with the **neighbor next-hop-self** command.

BGP Synchronization Rule

The BGP synchronization rule requires that when a BGP router receives information regarding a network from an iBGP neighbor, it does not use that information until a matching route is learned via an IGP or static route. It also does not advertise that route to an eBGP neighbor unless a matching route is in the routing table. In Figure 7-2, if RtrB advertises a route to RtrC, then RtrC does not submit it to the routing table or advertise it to RtrD unless it also learns the route from some other IGP source.

Recent IOS versions have synchronization disabled by default. It is usually safe to turn off synchronization when all routers in the autonomous system run BGP. To turn it off in earlier IOS versions, use the command **no synchronization** under BGP router configuration mode.

Configuring Basic BGP

Before configuring BGP, gather the network requirements you need, which should include the following:

- Confirmation as to whether you need to run iBGP for internal connectivity
- External connectivity to the ISP

- Configuration parameters, such as neighbor IP addresses and their AS number, and which networks you will advertise via BGP

Table 7-1 lists the basic BGP configuration commands and their functions.

Table 7-1 Basic BGP Configuration Commands

Command	Description
router bgp <i>AS-number</i>	Starts the BGP routing process on the router.
neighbor <i>ip-address</i> remote-as <i>AS-number</i>	Sets up peering between BGP routers. IP address must match the source of routing updates.
neighbor <i>peer-group-name</i> peer-group	Creates a peer group to which you can then assign neighbors.
neighbor <i>ip-address</i> peer-group <i>peer-group-name</i>	Assigns a neighbor to a peer group.
neighbor { <i>ip-address</i> <i>peer-group-name</i> } next-hop-self	Configures a router to advertise its connected interface as the next hop for all routes to this neighbor or peer group.
neighbor { <i>ip-address</i> <i>peer-group-name</i> } update-source <i>interface-type number</i>	Configures a router to use the IP address of a specific interface as the source for its advertisements to this neighbor.
no synchronization	Turns off BGP synchronization.
network <i>prefix</i> [mask <i>subnet-mask</i>]	Initiates the advertisement of a network in BGP.

BGP Network Command

In most IGPs, the **network** command starts the routing process on an interface and causes the protocol to advertise the network associated with that interface. In BGP, the **network** command tells the router to originate an advertisement for that network. The network does not have to be connected to the router; it just has to be in the routing table. In theory, it can even be a network in a different autonomous system (not usually recommended).

When advertising a network, BGP assumes you are using the default classful subnet mask. If you want to advertise a subnet, you must use the optional keyword **mask** and specify the subnet mask to use. Note that this is a subnet mask, not the inverse mask used by OSPF and EIGRP network statements. The routing table must contain an exact match (prefix and subnet mask) to the network listed in the network statement before BGP advertises the route.

BGP Peering

Before any BGP speaker can peer with a neighbor router, that neighbor must be statically defined. A TCP session must be established, so the IP address used to peer with must be reachable. This enables different configuration and policies to be applied to different neighbors. Some BGP routers have many neighbors; because each must be configured separately, the configuration can get unwieldy. To alleviate this, neighbors with the same configuration parameters can be grouped into *peer groups*. This enables the configuration to be entered once, and then applied by joining the neighbor to the peer group. Peer group policies can be overridden by specific neighbor configuration.

BGP assumes that external neighbors are directly connected and that they are peering with the IP address of the directly connected interface of their neighbor. If not, you must tell BGP to look more than one hop away for its neighbor, with the **neighbor ip-address ebgp-multihop number-of-hops** command. You might use this command if you are peering with loopback interface IP addresses, for instance. BGP assumes that internal neighbors might not be directly connected, so this command is not needed with iBGP. If you do peer with loopback IP addresses, you must change the source of the BGP packets to match the loopback address with the **neighbor ip-address update-source interface** command.

To take down the peering session with a neighbor but keep the neighbor configuration, use the **neighbor ip-address shutdown** command.

BGP Peering States

The command **show ip bgp neighbors** shows a list of peers and the status of their peering session. This status can include the following states:

- **Idle:** No peering; router is looking for neighbor. Idle (admin) means that the neighbor relationship has been administratively shut down.
- **Connect:** A remote peer is trying to initiate a TCP session with the local BGP process.
- **Active:** The local BGP process is trying to establish a TCP session with its remote peer.
- **OpenSent:** TCP session is up, and an Open message has been sent to establish the peering. Router waits for an answering Open message from its peer.
- **OpenConfirm:** Router has sent a Keepalive message to its peer and is waiting to receive a Keepalive message in return.
- **Established:** Routers have a BGP peering session. This is the desired state.

Troubleshooting BGP

Troubleshoot session establishment with debug commands. Use **debug ip bgp events** or **debug ip bgp ipv4 unicast** to see where the process fails. Some common failure causes include the following:

- AS number misconfiguration
- Neighbor IP address misconfiguration
- A neighbor with no neighbor statement for your router
- A neighbor with no route to the source address of your router's BGP messages

BGP Path Selection

IGPs, such as EIGRP or OSPF, choose routes based on lowest metric. They attempt to find the shortest, fastest way to get traffic to its destination. BGP, however, has a different way of route selection. It assigns various attributes to each path; these attributes can be administratively manipulated to control the path that is selected. It then examines the value of these attributes in order, until it chooses just one path by default. It must be manually configured to choose multiple paths with the command **maximum-paths number-of-paths**. BGP can load balance over a maximum of six paths.

BGP Attributes

BGP chooses a route to a network based on attributes included in route advertisements for that network. There are four categories of attributes:

- **Well-known mandatory:** Must be recognized by all BGP routers, present in all BGP updates, and passed on to other BGP routers; for example, AS path, origin, and next hop.
- **Well-known discretionary:** Must be recognized by all BGP routers and passed on to other BGP routers but need not be present in an update; for example, local preference.
- **Optional transitive:** Might or might not be recognized by a BGP router but is passed on to other BGP routers. If not recognized, it is marked as partial; for example, aggregator or community.
- **Optional nontransitive:** Might or might not be recognized by a BGP router and is not passed on to other routers; for example, Multi-Exit Discriminator (MED), or originator ID.

Table 7-2 lists common BGP attributes, their meanings, and their category.

Table 7-2 BGP Attributes

Attribute	Meaning
AS path	An ordered list of all the autonomous systems through which this update has passed. Well-known, mandatory.
Origin	How BGP learned of this network. i = by network command, e = from EGP, ? = redistributed from other source. Well-known, mandatory.
Local Preference	A value telling iBGP peers which path to select for traffic leaving the AS. Default value is 100. Well-known, discretionary.
Multi-Exit Discriminator	Suggests to a neighboring autonomous system which of multiple paths to select for traffic bound into your autonomous system. Lowest MED is preferred. Optional, nontransitive.
Weight	Cisco proprietary, to tell a router which of multiple local paths to select for traffic leaving the AS. Highest weight is preferred. Has only local significance.

BGP Path Selection Criteria

BGP attempts to narrow its path selection down to one best path; it does not load balance by default. To do so, it examines the path attributes of any loop-free, synchronized (if synchronization is enabled) routes with a reachable next-hop in the following steps:

- Step 1.** Choose the route with the highest weight.
- Step 2.** If weight is not set, select the route with the highest local preference.
- Step 3.** Choose the route originated by this router.
- Step 4.** Select the path with the shortest autonomous system path.
- Step 5.** Choose the path with the lowest origin code (i is lowest, e is next, and ? is last).
- Step 6.** Select the route with the lowest MED, if the same autonomous system advertises the possible routes.
- Step 7.** Choose an eBGP route over an iBGP route.
- Step 8.** Select the route through the nearest IGP neighbor as determined by the lowest IGP metric.
- Step 9.** Choose the oldest route.

Step 10. Select a path through the neighbor with the lowest router ID.

Step 11. Choose a path through the neighbor with the lowest IP address.

To enable BGP to load balance over more than one path, you must enter the command **maximum-paths** *number-of-paths*. BGP can load balance over a maximum of six paths.

Influencing BGP Path Selection

BGP was not created to be a fast protocol; it was designed to enable as much administrative control over route path selection as possible. Path selection is controlled by manipulating BGP attributes, usually implementing route maps. You can set a default local preference by using the command **bgp default local-preference** and a default MED for redistributed routes with the **default-metric** command under the BGP routing process. But by using route maps, you can change attributes for certain neighbors only or for certain routes only. Example 6-1 in Chapter 6 shows a route map that sets a local preference of 200 for specific redistributed routes. This is higher than the default local preference of 120, so routers within the AS are more likely to prefer that path than others.

Route maps can also be applied to routes sent to or received from a neighbor. Example 7-1 shows a simple route map that sets an MED value and adds two more copies of its AS number to the AS path on all routes advertised out to an eBGP neighbor.

Example 7-1 Route Map to Set MED and Prepend

```
route-map MED permit 10
  set metric 50
  set as-path prepend 65001 65001
!
router bgp 65001
  neighbor 10.1.1.1 route-map MED out
```

When attributes are changed, you must apply the changes. Either clear the BGP session (**clear ip bgp ***) or do a soft reset (**clear ip bgp * soft in | out**). Routers using recent IOS versions do a route refresh when the session is cleared inbound.

Filtering BGP Routes

BGP provides several ways of controlling the routes advertised to, or received from, a BGP peer. This section discusses prefix lists and AS-path access lists, both of which can be either applied directly to a peer or used in a route map.

Prefix Lists

Prefix lists combined with route maps can be used to filter the routes advertised to or received from a BGP peer, control routes redistributed into BGP, and set BGP attributes for specific routes. Prefix lists also can be applied directly to a neighbor to filter route updates.

Create a prefix list to match the networks to be filtered. Permit the networks you want to allow to be advertised (all others are denied by default), and then apply the prefix list to the BGP neighbor, inbound or outbound.

Example 7-2 shows a prefix list that permits only summary routes in the 172.31.0.0 network. All other routes are denied by default. The prefix list is then applied to BGP neighbor 10.1.1.1 outbound, so only these routes are advertised to that peer.

Example 7-2 Configuring a Prefix List

```
ip prefix-list Summary permit 172.31.0.0/16 le 20
!
router bgp 65001
neighbor 10.1.1.1 prefix-list Summary out
```

To verify the results of your configuration use the command **show ip prefix-list**. To clear the counters shown in that command, use the **clear ip prefix-list** command.

You can combine a prefix list with a route map to set attributes on the routes allowed in the prefix list. In Example 7-3, prefix list Summary is used again. Route map CCNP sets the MED for those routes to 100 when they are advertised. It sets a MED of 200 for all other routes advertised. The route map is then applied to BGP neighbor 10.1.1.1 outbound.

Example 7-3 Using a Prefix List with a Route Map

```
route-map CCNP permit 10
match ip address prefix-list Summary
set metric 100
route-map CCNP permit 20
set metric 200
```

Example 7-3 Continued

```
!
router bgp 65001
neighbor 10.1.1.1 route-map CCNP out
```

AS Path Access List

An AS path access list looks at the AS path portion of route advertisements. The command syntax is **ip as-path access-list** *acl-#* { **permit** | **deny** } *regex*. It is applied directly to a neighbor using the filter list command **neighbor ip-address/peer-group filter-list** *acl-#* { **in** | **out** }.

AS path ACLs employ regular expressions (regex) to identify specific strings of numbers. Regular expressions consist of special characters, numbers, and letters. An explanation of regular expressions is beyond the scope of this book, but some frequently used BGP characters include

- **^** : This signifies the beginning of a string. For example, ^65001 would match an AS path that has AS 65001 as its first entry.
- **\$** : This signifies the end of a string. For example, 65001\$ would match an AS path that ended in AS 65001.

Local AS path information is added to a route advertisement when it is sent to an eBGP peer. Thus, a route originated within the local AS does not have AS path information attached to it in the BGP database. When the router looks to confirm whether it should be advertised, it sees an empty AS path set. Most companies do not want to ever become a transit AS, so they never want to advertise routes learned from another AS. They only want to advertise locally originated routes. The way this is typically accomplished is with the regular expression ^\$—this signifies an empty AS path list. Example 7-4 shows an AS path ACL used to limit route advertisements to local routes only.

Example 7-4 Configuring an AS Path Access List

```
ip as-path access-list 1 permit ^$
!
router bgp 65001
neighbor 10.1.1.1 filter-list 1 out
```

AS path ACLs can also be used in route maps to set attributes on the matching routes. Route map CCNP in Example 7-5 limits the advertised routes to local only and sets a MED of 100 when they are advertised to the BGP neighbor.

Example 7-5 Using an AS Path ACL with a Route Map

```

ip as-path access-list 1 permit ^$
!
route-map CCNP permit 10
match as-path 1
set metric 100
!
router bgp 65001
neighbor 10.1.1.1 route-map CCNP out

```

Order of Operations

As you have seen, multiple filters and policies can be applied to the same BGP neighbor. If you are configuring these, you must understand the order in which they are processed by the router. The order of BGP inbound and outbound policy application has changed over the years and varies based on IOS version. The order given here is based on IOS version 15.2(4)M3.

Note that a distribute list and a prefix list cannot both be applied to the same neighbor in the same direction.

Incoming route advertisements are handled in this order:

1. Filter list
2. Prefix list or distribute list
3. Route map

Outgoing route advertisements are processed in the following order:

1. Filter list
2. Route map or unsuppressed map
3. Advertise map
4. Prefix list or distribute list
5. Outbound route filter (ORF) prefix list

BGP Authentication

BGP supports MD5 authentication between neighbors, using a shared password. It is configured under BGP router configuration mode with the command **neighbor** {*ip-address* | *peer-group-name*} **password** *password*.

When authentication is configured, BGP authenticates every TCP segment from its peer and checks the source of each routing update. Most ISPs require authentication for their eBGP peers.

Peering succeeds only if both routers are configured for authentication and have the same password. If a router has a password configured for a neighbor, but the neighbor router does not, a message such as the following displays on the console while the routers attempt to establish a BGP session between them:

```
%TCP-6-BADAUTH: No MD5 digest from [peer's IP address]:11003 to
[local router's IP address]:179
```

Similarly, if the two routers have different passwords configured, a message such as the following will display on the screen:

```
%TCP-6-BADAUTH: Invalid MD5 digest from [peer's IP address]:11004
to [local router's IP address]:179
```

Verifying BGP

One of the best commands to verify and troubleshoot your BGP configuration is **show ip bgp** *[network]* to see the BGP topology database. **show ip bgp** is such an important command that it's worth looking at in depth. Example 7-6 shows the command output, which lists a table of all the networks BGP knows about, the next hop for each network, some of the attributes for each route, and the AS path for each route.

Example 7-6 Interpreting the BGP Topology Database

```
Router# show ip bgp
BGP table version is 5, local router ID is 192.168.11.1
Status codes: s suppressed, d damped, h history, * valid, > best,
i - internal,
                r RIB-failure, S Stale, m multipath, b backup-path,
f RT-Filter,
                x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
```

	Network	Next Hop	Metric	LocPrf	Weight	Path
*>	192.168.11.0	0.0.0.0	0		32768	i
*>i	192.168.22.0	192.168.2.2	0	100		0 i
* i	198.51.100.0	209.165.201.6	0	100		0 65200
	65300 i					
*>		209.165.201.2				0 65100
	65300 i					

```

* i 203.0.113.0/22    209.165.201.6          0    100    0 65200
65300 63000 i
*>                   209.165.201.2          0 65100
65300 i

```

Networks are listed in numerical order, smallest to largest. The first three columns list each route's status. An asterisk (*) in the first column means that the route has a valid next hop. Some other options for the first column include the following:

- **“s” for suppressed:** BGP knows about this network but is not advertising it, usually because it is part of a summarized route.
- **“d” for dampened:** BGP can stop advertising a network that flaps (goes up and down) too often until it is stable for a period of time.
- **“h” for history:** BGP knows about this network but does not currently have a valid route to it.
- **“r” for RIB failure:** The route was advertised to BGP but was not installed in the IP routing table. This might be because another protocol has the same route with a better administrative distance.
- **“S” for stale:** Used with nonstop forwarding to indicate that the route is stale and needs to be refreshed when the peer is reestablished.

The second column has a greater-than sign (>) beside the route that was selected as the best path to that network. In the example, there are two routes for network 198.51.100.0 and the router has selected the second one.

The routes learned from an iBGP neighbor have an “i” in the third column. The third column is blank for those routes learned from an eBGP neighbor. Recall that BGP prefers routes through an eBGP neighbor over an internal route.

The fourth column lists the networks. Those without a subnet mask shown use their classful mask. As shown in the example, when the router learns about the same network from multiple sources, it lists only the network once.

The fifth column lists the next-hop address for each route. As you learned in the section on BGP next hops, this might or might not be a directly connected router. A next-hop of 0.0.0.0 means that the local router originated the route.

If a MED value was received with the route, it is listed in the Metric column. This column can also have a value of 0—the default MED—in the Metric column.

The default Local Preference is 100. This may or may not be displayed in the table. Any nondefault Local Preference value is displayed in that column.

The Weight column always displays a value. The default value is 0 except for locally originated routes, which have a default weight of 32768.

The ninth column shows the AS path for each network. A blank AS path means that the route was originated in the local AS. Reading this field from left to right, the first AS number shown is the adjacent AS this router learned the route from. After that, the autonomous systems that this route traversed are shown in order. The last AS number listed is the originating AS. In the example, our router received an advertisement about network 198.51.100.0 from its neighbor AS 65200, which heard about it from AS 65300. Because AS 65300 is the last one listed, it is where the route advertisement originated.

Note

In the AS Path column, network 203.0.113.0 shows AS 65300 twice in its AS path list. Most likely, AS 65300 has prepended an extra copy of its AS number to make the path through 209.165.201.6 less attractive than the alternative path.

The last column shows how BGP originally learned about the route. All the networks in the example show an “i” for their origin codes. This means that the originating router had a network statement for that route. BGP considers a route with an origin code of “?” as an “incomplete” route. This typically means that the route was redistributed into BGP. You will likely never see the third possibility, an “e,” because that means BGP learned the route from the Exterior Gateway Protocol (EGP), which is no longer in use.

Some other useful commands for verifying and troubleshooting BGP include

- **show ip bgp rib-failure:** Displays routes that were not inserted into the IP routing table and the reason they were not used.
- **show ip bgp summary:** Displays the memory used by the various BGP databases, BGP activity statistics, and a list of BGP neighbors.
- **show ip bgp neighbors:** Displays details about each neighbor. Can be modified by adding the neighbor IP address.
- **show ip bgp neighbors *address* [received | routes | advertised]:** Lets you monitor the routes received from and advertised to a particular neighbor.

You can search for “public route servers” to find listings of BGP routers that enable public telnet access for viewing their BGP tables. Trying some of these commands on a public route server can help you become familiar with them.

Multiprotocol BGP

Multiprotocol BGP (MP-BGP) involves two extensions to BGP4 that enable BGP to carry reachability information for other protocols, such as IPv6, multicast IPv4, and MPLS. The extensions enable the NEXT_HOP attribute to carry IPv6 addresses and NLRI (network layer reachability information) to an IPv6 prefix. An address-family command is added to the BGP configuration to enable this.

The router ID must be manually configured in an all-IPv6 implementation and is a 32-bit dotted-decimal number. Unlike the IGP, configuration is done under the BGP router configuration mode, not at the interface. Neighbors are configured under the global BGP configuration mode but must be activated under the IPv6 address family mode. Any policies or networks relevant to this MP-BGP extension are also configured under the address family.

One interesting “feature” is that, because BGP runs over TCP, its advertisements can carry both IPv4 and IPv6 routes regardless of which protocol is used as the transport. You can use IPv4 to carry both IPv4 and IPv6 routes, you can use IPv6 to carry routes from both protocols, or you can use separate BGP sessions for IPv4 and IPv6.

In Example 7-7, an IPv4 BGP session is used to carry both IPv4 and IPv6 routes.

Example 7-7 Carrying IPv6 Routes over an IPv4 BGP Session

```
RouterA(config)# router bgp 65000
RouterA(config-rtr)# neighbor 10.1.1.2 remote-as 65001
!
RouterA(config-rtr)# address-family ipv4 unicast
RouterA(config-rtr-af)# neighbor 10.1.1.2 activate
RouterA(config-rtr-af)# network 10.1.0.0 mask 255.255.0.0
!
RouterA(config-rtr)# address-family ipv6 unicast
RouterA(config-rtr-af)# neighbor 10.1.1.2 activate
RouterA(config-rtr-af)# network 2001:db8:1::/48
```

Example 7-8 shows using BGP to advertise only IPv6 routes with the session established using IPv6 as the transport protocol. Note that a router ID has been manually configured.

Example 7-8 Using BGP with an IPv6 Session

```
RouterA(config)# router bgp 65000
RouterA(config-rtr)# router-id 10.255.255.1
RouterA(config-rtr)# neighbor 2001:db8:1:1:5::4 remote-as 65001
!
RouterA(config-rtr)# address-family ipv6 unicast
RouterA(config-rtr-af)# neighbor 2001:db8:1:1:5::4 activate
RouterA(config-rtr-af)# network 2001:db8:1::/48
```

Finally, Example 7-9 shows the BGP commands to run separate BGP sessions for IPv4 and IPv6. The IPv6 neighbor must be manually removed from the IPv4 address family because all neighbors are enabled under the IPv4 address family by default.

Example 7-9 Separate BGP Sessions for IPv4 and IPv6

```
RouterA(config)# router bgp 65000
RouterA(config-rtr)# router-id 10.255.255.1
RouterA(config-rtr)# neighbor 10.1.1.2 remote-as 65001
RouterA(config-rtr)# neighbor 2001:db8:1:1:5::4 remote-as 65001
!
RouterA(config-rtr)# address-family ipv4 unicast
RouterA(config-rtr-af)# no neighbor 2001:db8:1:1:5::4 activate
RouterA(config-rtr-af)# neighbor 10.1.1.2 activate
RouterA(config-rtr-af)# network 10.1.0.0 mask 255.255.0.0
!
RouterA(config-rtr)# address-family ipv6 unicast
RouterA(config-rtr-af)# neighbor 2001:db8:1:1:5::4 activate
RouterA(config-rtr-af)# network 2001:db8:1::/48
```

To verify your IPv6 BGP configuration, use the commands **show bgp ipv6 unicast summary** and **show ipv6 route bgp**.

CHAPTER 8

Infrastructure Security

Device Access Control

Because of their importance in packet forwarding, routers make attractive targets for network attacks. This section discusses some of the built-in protections you can enable to secure access to your device.

The most basic type of security is to require a password to access the router. Cisco devices use several types of passwords for access control:

- **Enable password:** Controls access to privileged EXEC mode if the enable secret password is not present. It is stored in clear text in the configuration by default. To configure, use the global command **enable password** *password*.
- **Enable secret:** Controls access to privileged EXEC mode. It is stored in a nonreversible one-way MD5 hash. If present in the configuration, it overrides the enable password. To configure, use the global command **enable secret** *password*. Recent IOS versions can use encryption type 4 (SHA-256). Configure this with the command **enable secret 4** *password*.
- **Line passwords:** Routers have VTY lines for Telnet and SSH access, the console port, the AUX port, and TTY lines that must all be protected. Access to these lines can be controlled either with AAA or with individual passwords applied to them. Line passwords are stored in clear text by default. To configure individual passwords on one of these lines, use the line-configuration mode commands **login** and **password** *password*.

You can encrypt the passwords that are stored in clear text by using the global command **service password-encryption**. However, this uses an encryption method that is easily broken, so do not rely on it to protect highly sensitive passwords.

SSH is more secure than Telnet; use it to access the device's VTY lines. Some Cisco devices have a feature called *Management Plane Protection* that enables you to designate specific interfaces as management interfaces, and

only management traffic such as SSH, SNMP, and TFTP is allowed to use these interfaces.

Disable any device services that are not being used. These can include the following, many of which are enabled by default:

- BOOTP server
- CDP
- HTTP/HTTPS
- ICMP Redirects
- IP source routing
- ICMP unreachable
- Proxy ARP

Cisco devices enable you to define up to 16 privilege levels with different command sets assigned to each. Normal user-level privileges are level 0. Standard privileged mode (access to all commands) is level 15. The following configuration shows how to assign the **traceroute** command to level 2, with a separate enable secret password of CCNP. After applying this configuration, users at level 0 can no longer execute traceroutes.

```
Router(config)# privilege exec level 2 traceroute  
Router(config)# enable secret level 2 CCNP
```

Router Security Features

In addition to securing access to the router itself, you must consider the security of the data that traverses your network. Cisco devices have some features that can help.

Access Control Lists

An *access control list (ACL)* is an ordered group of statements that permit or deny specified types of traffic. They are used extensively in router security configurations for things such as

- Controlling traffic through the router
- Permitting or denying access to services

- Mitigating address spoofing
- Mitigating attacks

The statements in an ACL are evaluated from the top down, in order. When the router finds a match, it takes the action in the matching statement and does not read the rest of the ACL.

IPv4 can use both standard and extended access lists. Standard ACLs enable filtering based on source IP address only. Extended ACLs permit filtering based on source or destination address as well as most other fields in the IP header. IPv4 access lists can be either numbered or named. Numbered standard ACLs range from 1 to 99 and 1300 to 1999. Numbered extended ACLs range from 100 to 199 and 2000 to 2699. IPv6 uses only named, extended access lists.

You can apply ACLs either inbound or outbound on an interface. Inbound ACLs affect traffic coming in on the interface; outbound ACLs affect traffic leaving out that interface.

All access lists have an implicit “deny any” at the end—any traffic not explicitly permitted by the ACL is denied. IPv6 access lists have two implicit permit statements before the final deny. Because IPv6 relies on the Neighbor Discovery protocol, blocking that would affect function. So IPv6 ACLs end with the following implicit rules:

```
permit icmp any any nd-na
permit icmp any any nd-ns
deny ipv6 any any
```

Configuring IPv4 ACLs

To configure a numbered standard IPv4 access control list, use the global command:

```
access-list number {permit | deny} source-address
source-wildcard-mask
```

To configure a basic numbered extended IPv4 access control list, use the global command:

```
access-list number {permit | deny} protocol source-address
source-wildcard-mask destination-address
destination-wildcard-mask
```

To configure a named ACL, use a global command to create it, and then configure each statement under the access-list configuration submode:

```
ip access-list {standard | extended} name
{permit | deny} protocol source-address source-wildcard-mask
destination-address destination-wildcard-mask
```

You can also configure an ACL statement that is in effect only during specified times. To do this, first create a time range that lists the times the ACL should be used. Then add the option **time-range** *time-range-name* to the end of your access list statement.

To apply an IPv4 access list to an interface, use the interface command:

```
ip access-group {number | name} {in | out}
```

To verify the ACL configuration and operation, use the command **show access-list**.

Configuring an IPv6 Access List

The commands to create an IPv6 ACL are similar to those for IPv4, except that you use the keyword **ipv6** in the command. Recall that IPv6 uses only named, extended ACLs.

```
ipv6 access-list name
{permit | deny} protocol source-address source-wildcard-mask
destination-address destination-wildcard-mask
```

The command to apply the access list to an interface is also slightly different. It uses the keyword **traffic-filter** rather than **access-group**:

```
ipv6 traffic-filter name {in | out}
```

To verify the ACL configuration and operation, use the command **show ipv6 access-list**.

Unicast Reverse Path Forwarding

Unicast Reverse Path Forwarding (uRPF) is a security feature that helps prevent traffic from spoofed addresses being sent through your network. It verifies whether the router can reach the source IP address of the packet. If the source address is not valid, the packet is dropped. There are two modes:

- **Loose mode:** A route for the source address must be in the routing table, but not as a default route or a route pointing to Null0. You can, however, configure the router to accept a default route as valid.

- **Strict mode:** A route for the source address must be in the routing table, and the packet must have come in on the interface associated with that route. In other words, the packet must have come in on the interface the router would use to send traffic out to that source address.

Use loose mode if you have asymmetric routing. uRPF relies on CEF, so make sure it is not disabled. Optionally, use an access list to control which traffic is checked.

uRPF is enabled at each interface with the following command:

```
ip verify unicast source reachable-via {any | rx} [allow-default]
  access-list
```

The **any** keyword enables loose mode, and the **rx** keyword enables strict mode.

To verify whether uRPF is configured on an interface, either check the router configuration or use the command **show cef interface type number**. To view uRPF operation, use the command **show ip traffic** to see how many packets have been dropped due to failing the uRPF check.

Tunneling Technologies

Many companies have sites or home offices that connect to the organization's network over the Internet. Because the Internet is inherently insecure, most use some form of a tunneling protocol and encryption to create a virtual private network (VPN). Tunneling protocols encapsulate the original packet in tunnel headers. Other routers along the path from source to destination look only at the tunnel headers and route based on them. The original headers and the data are left intact. The tunnel header is removed at the destination and the traffic is decrypted.

This section discusses three types of tunnels:

- GRE
- DMVPN
- Easy VPN

It also discusses carrying PPP over Ethernet.

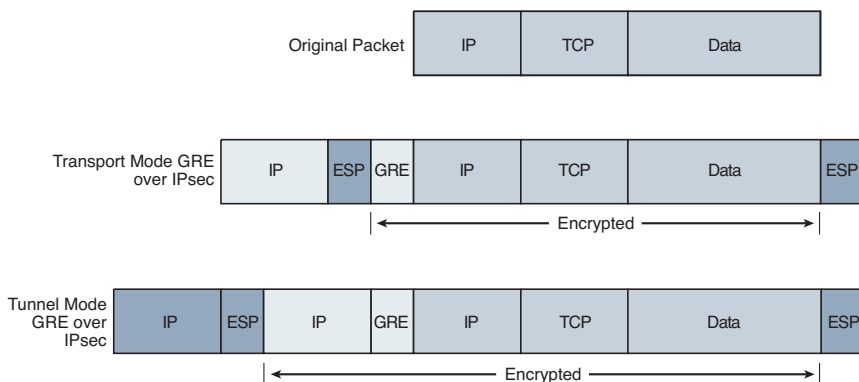
GRE Tunnels

Generic Routing Encapsulation (GRE) is a tunneling protocol that supports multiple Layer 3 protocols. It also enables the use of multicast routing protocols across its tunnel. It adds a 20-byte IP header and a 4-byte GRE header, hiding the existing packet headers. The GRE header contains a flag field and a protocol type field to identify the Layer 3 protocol being transported. It might optionally contain a tunnel checksum, tunnel key, and tunnel sequence number. GRE does not encrypt traffic or use any strong security measures to protect the traffic.

GRE can be used along with IPsec to provide data source authentication and data confidentiality and ensure data integrity. GRE over IPsec tunnels are typically configured in a hub-and-spoke topology over an untrusted WAN to minimize the number of tunnels that each router must maintain.

Figure 8-1 shows how the GRE and IPsec headers work together.

Figure 8-1 GRE over IPsec Headers



Configuring a GRE Tunnel

To configure GRE using IOS commands, you must first construct a logical tunnel interface. GRE commands are then given under that interface. Specify a source and destination for the tunnel; the source is a local outgoing interface or an interface address. You might also give the tunnel interface an IP address and indicate the tunnel mode. GRE is the default mode.

Example 8-1 shows the commands to configure a GRE tunnel interface.

Example 8-1 GRE Tunnel Configuration Commands

```
interface Tunnel number
  ip address address mask and/or
```

```

ipv6 address address/prefix
tunnel source {interface | address}
tunnel destination {address}
tunnel mode {gre | gre ipv6}

```

Be sure to configure the device on the other end of the tunnel as well. Adjust routing so that traffic uses the tunnel for destinations on the other side—you might want to use a static route for this. Routing protocols can be carried over the tunnel, but be sure that the router does not learn the tunnel destination address through the tunnel itself. Verify the tunnel configuration using **show interface**, or use **traceroute** to confirm whether traffic goes over the tunnel.

DMVPN

Configuring a full mesh of GRE or IPsec tunnels becomes unwieldy as the network grows in size. Dynamic Multipoint Virtual Private Network (DMVPN) dynamically creates IPsec tunnels on demand, thus easing the administrative burden. DMVPN leverages multipoint GRE (mGRE), IPsec, and Next Hop Resolution Protocol (NHRP). The hub site(s) creates an mGRE tunnel interface. Each of the remote sites (spokes) is configured with information about the hub and initiates a tunnel with the hub. The hub uses NHRP to map each spoke's tunnel IP address to its physical address.

When one spoke wants to send traffic to another, it uses an NHRP query to ask the hub for the physical next-hop address of the other spoke. It can then build a dynamic tunnel to that spoke.

DMVPN is a good choice when spoke-to-spoke connections are needed because traffic does not have to traverse the hub. It also supports multicast and routing protocols between the hub and the spokes.

Easy VPN

Cisco Easy VPN simplifies the deployment of services and policies to remote users. Easy VPN enables a server to push down VPN configuration to a client. It is a way to create site-to-site VPNs without manually configuring each remote router, so it is good for remote sites lacking technical support. It also can be used with software clients for remote users.

Cisco Easy VPN dynamically handles the following items:

- Negotiating VPN tunnel parameters
- Establishing the VPN tunnel based on those parameters

- NAT, PAT, or ACL configuration
- User authentication
- Managing encryption and decryption keys
- Authenticating, encrypting, and decrypting traffic

Cisco Easy VPN has two components: a server and a remote client. The Easy VPN Server can be a Cisco router, ASA Firewall, or Cisco VPN concentrator. It contains security policies and pushes those to remote clients. The Easy VPN Remote can be a Cisco router, ASA Firewall, a hardware client, or a software client. It contacts the server and receives policies from it to establish an IPsec tunnel between the remote site and the central site.

PPPoE

Although not technically a tunneling protocol, Point-to-Point Protocol over Ethernet (PPPoE) is a method used to create logical point-to-point links over a multipoint medium. PPP is a Layer 2 protocol that can be employed to transport many different protocols over point-to-point links. It was originally designed to carry TCP/IP over serial links but is now additionally being used with DSL, SONET, and Metro Ethernet. PPP provides authentication and some other controls that are lacking in these transport mediums.

There are two stages to establishing a PPPoE connection:

- **Active Discovery:** Because PPP was created to be used over a point-to-point connection, and Ethernet is multipoint, PPPoE uses a network-access server discovery process. After a server has been discovered, the client router associates with it and the PPPoE layer is established.
- **PPP Session Setup:** This includes the normal PPP session negotiations and authentication. If this stage succeeds, a virtual point-to-point link is established. Traffic is then sent across the link encapsulated in PPPoE and PPP headers.

Authentication is typically done via the Challenge Handshake Authentication Protocol (CHAP). CHAP uses a three-way handshake to verify the identity of its peer device.

Example 8-2 shows a PPPoE client that receives its outside IP address from the ISP. You can also use statically assigned IP addresses. Note that PPP requires a dialer interface even though nothing is actually being dialed. The IP address and authentication configuration is performed on the dialer interface.

Example 8-2 Configuring PPPoE

```
Router(config)# interface dialer 1
Router(config-if)# mtu 1492
Router(config-if)# encapsulation ppp
Router(config-if)# dialer pool 1
Router(config-if)# ip address negotiated
Router(config-if)# ppp authentication chap
Router(config-if)# ppp chap password CCNP
!
Router(config-if)# int e0/0
Router(config-if)# pppoe-client dial-pool-number 1
```

If you were configuring this router as an edge device for a small or home office, you might also want to add NAT and DHCP server commands to the configuration. Verify the configuration with the command **show pppoe session [all | packets]**. Confirm the PPP negotiations with **debug pppoe {data | errors | events | packets}**.

CHAPTER 9

Infrastructure Services

Cisco devices have several features and services that are useful for administering, managing, troubleshooting, and optimizing your network. This chapter covers the following subset of these:

- SMTP
- General logging
- Debug
- Syslog
- NTP
- DHCP
- NetFlow
- NAT

The available features and services vary by device, and the commands differ by operating system and version. This book uses commands from IOS version 15.2.

Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) collects information from SNMP agents residing on network devices, either through regular polling or by event-generated traps. The information is stored on the local device in a management information base (MIB). It can then be exported to an SNMP server for analysis. The information that can be collected varies by device but is extensive.

There are three versions of SNMP. Versions 1 and 2c send the community strings in clear text. They control access to the MIB by using SNMP community strings, which can be read-only (RO) or read-write (RW). Version 2c is the most widely accepted implementation of version 2. Neither of these can authenticate the source of a message nor encrypt a message; therefore, they should be used only for read-only access. SNMP version 3 adds three security models:

- **noAuthNoPriv:** Neither authenticates nor encrypts; specified by **no auth** during configuration
- **authNoPriv:** Authenticates the sender but does not encrypt the message; specified by **auth** during configuration
- **authPriv:** Both authenticates the sender and encrypts the message; specified by **priv** during configuration

SNMPv2c Configuration

Example 9-1 creates a standard access list that allows SNMPv2 access only to the SNMP manager at 10.1.1.1. Two community strings are created: **ccnp** for read-only access and **clsc0** for read-write access. Read-write access is permitted only from the host specified in access list 1. Next, the SNMP server address is given, along with the command to send traps messages to that server.

Example 9-1 SNMPv2 Configuration

```
Router(config)# access-list 1 permit 10.1.1.1
Router(config)# snmp-server community ccnp ro
Router(config)# snmp-server community clsc0 rw 1
Router(config)# snmp-server host 10.1.1.2 traps
```

SNMPv3 Configuration

SNMPv3 configuration can be more complex because you have the option to configure authentication and encryption. The basic items that you need to configure in SNMP version 3 are as follows:

- Create an access list for limiting SNMP access, if desired.
- Create an SNMPv3 group, assign it one of the three security models, and optionally a security level.
- Create an SNMPv3 user, link it with a group, and optionally configure authentication and encryption parameters.
- Specify an SNMP host or hosts.
- Enable SNMP traps.

In Example 9-2, an access list is created to restrict SNMP access to hosts on the management subnet of 192.168.1.0/24. An SNMPv3 group named CCNP

is configured to use security model `authNoPriv`, meaning that it authenticates but does not encrypt SNMP traffic. Access list 1 is linked to the group. An SNMPv3 user named `admin1` is created and linked to the CCNP group. User `admin1` will have an authentication password of `Secret` and use HMAC SHA for authentication. (Note the router's message after the user is configured.) The SNMP server address is added and linked to user `admin1`, and the collection of traps is enabled.

Example 9-2 Configuring SNMPv3

```
Router(config)# access-list 1 permit 192.168.1.0 0.0.0.255
Router(config)# snmp-server group CCNP v3 auth access 1
Router(config)# snmp-server user admin1 CCNP v3 auth sha Secret

*Jun 14 19:37:07.930: Configuring snmpv3 USM user, persisting
snmpEngineBoots. Please Wait...

Router(config)# snmp-server host 192.168.1.1 traps version 3 auth
admin1
Router(config)# snmp-server enable traps
!
Router# show snmp
<output omitted>
SNMP global trap: enabled
SNMP logging: enabled
Logging to 192.168.1.1.162, 0/10, 260 sent, 0 dropped.
```

Additional verification commands include **show snmp group**, **show snmp user**, and **show snmp host**.

Logging

Logging is an essential network management tool, providing insight into network traffic and device failures or critical events. Routers can gather information from various sources, such as debugs, Simple Network Management Protocol (SNMP) traps, system logging (syslog), and NetFlow. This information can be stored on the buffer of the local device, but this is not scalable and is erased if the device reloads. Sending it to a remote server or collection device is usually preferable.

By default, all logging is displayed on the device's console. You can stop this with the command **no logging console** or control it with the **logging console level** command. There are other options, such as

- **Terminal line:** Enable this with the command **terminal monitor**. Disable it with the command **terminal no monitor**.
- **Internal buffer:** Enable this with the command **logging buffered**.
- **Syslog server:** This is detailed in the “Syslog” section of this chapter.
- **SNMP server:** This is detailed in the “Simple Network Management Protocol” section of this chapter.

You will likely want to record the time the log message was sent. The command to enable this is **service timestamps {debug | log}**.

Debug

Debug is a useful troubleshooting tool, with the caveat that it can produce large amounts of traffic because it displays the device’s internal processes. The command to enable debugging is **debug action-to-be-displayed**. You can view many actions with debug, and they vary by device; use the **debug ?** command to see them all. To turn off debugging, use either the same command preceded by **no** or the command **undebug all** (the short version of this command is **u all**).

When you give the **debug** command, it applies to all traffic that matches; for example, debugging AAA looks at AAA traffic. You can limit it for some protocols to traffic matching certain conditions, such as an interface, by using *conditional debug*. The generic command to enable conditional debugging is **debug condition condition modifier**. For example, to restrict the debug output to a specific interface, use **debug condition interface interface**.

Syslog

Cisco devices produce system logging (or *syslog*) messages that can be output to the device console, VTY connection, system buffer, or remote syslog server. You are probably familiar with the following syslog message, for example:

```
%SYS-5-CONFIG_I: Configured from console by console
```

If sent to a syslog server, messages are sent on UDP port 514. A syslog message always starts with the percent sign and has the following format:

```
%FACILITY-SUBFACILITY-SEVERITY-MNEMONIC: message text
```

Each portion of a syslog message has a specific meaning:

- **FACILITY-SUBFACILITY:** This reports the protocol, module, or process that generated the message. Some examples are SYS for the operating system, OSPF, IF for an interface, and IP.
- **SEVERITY:** A number from 0 to 7 designating the importance of the action reported. Enabling logging of any of these levels also enables all levels above it. For instance, enabling level 4 also enables levels 0 through 3. The logging levels are
 - Emergency—0
 - Alert—1
 - Critical—2
 - Error—3
 - Warning—4
 - Notice—5
 - Informational—6
 - Debugging—7
- **MNEMONIC:** A code that identifies the action reported.
- A plain-text description of the event that triggered the syslog message.

Configuring Syslog

Configuring a Cisco device to send syslog messages to an external server is fairly simple. The server itself must, of course, also be configured to receive those messages. Example 9-3 shows the configuration to send syslog messages of level 3, Error, or worse to a server located at 192.168.1.1. Because there might be several ways to reach that server, the source interface of Lo0 is specified.

Example 9-3 Configuring Syslog

```
Router(config)# logging 192.168.1.1
Router(config)# logging trap 3
Router(config)# logging source-interface lo0
Router(config)# logging on
```

Network Time Protocol

The Network Time Protocol (NTP) is a client/server protocol that enables network devices to dynamically synchronize their internal clocks with a common external source. Clock synchronization is important for correlating syslog messages, authentication key expiration, certificate-based encryption, and other features. Cisco devices can act either as an NTP peer or an NTP server. In a peer association, the local system can either synchronize to another device or the other device can synchronize to it. In a server association, the local device can only synchronize to an NTP server.

NTP Version 3 supports IPv4, and clients can use broadcasts for clock updates. Version 4 adds IPv6 support, is backward compatible with v3, and messages are sent via multicast. NTP uses the concept of “stratum” to designate the reliability of a clock source, with 1 being the best. A stratum 1 clock is an authoritative source such as an atomic clock. Stratum 2 is one hop away from that, and so on.

Time is tracked internally using universal coordinated time (UTC). You can configure a time zone and daylight saving time so that time is displayed in local format.

Because time synchronization is a security-related feature, configuring authentication for NTP information is wise. This prevents an attacker from spoofing NTP packets to corrupt the system clock. For added security, you can add an access list to restrict the IP addresses with which the router can synchronize time.

Example 9-4 shows a router configured to get its time from an authenticated NTP version 3 server located at 10.1.1.1. The time zone is set to Eastern Standard Time and is adjusted for daylight saving time.

Example 9-4 NTP Configuration

```
Router(config)# ntp authenticate
Router(config)# ntp authentication-key 1 md5 my_ntp_key
Router(config)# ntp trusted-key 1
Router(config)# ntp server 10.1.1.1 key 1
Router(config)# clock timezone EST -5
Router(config)# clock summertime EDT recurring
```

The command for IPv6 routers to synchronize their time with an NTPv4 server is

```
Router(config)# ntp server ipv6-address version 4
```

Verify your NTP configuration with the commands **show clock**, **show ntp status**, and **show ntp associations**.

DHCP

DHCP automates the assignment of IP addresses to network hosts. DHCP addresses can be allocated in the following ways:

- **Manually:** A specific IP address is assigned to a MAC address.
- **Automatic:** An IP address is permanently assigned to a host.
- **Dynamic:** The IP address is assigned for a limited amount of time or until the client releases it.

DHCP for IPv4

The process of acquiring an IP address from a DHCP server has four steps:

- Step 1.** The host broadcasts a DHCPDISCOVER message.
- Step 2.** The server responds with a DHCPOFFER message containing the IP address and optionally other settings.
- Step 3.** The client broadcasts a DHCPREQUEST message, requesting the offered IP address.
- Step 4.** The server sends a DHCPACK confirming the address assignment.

Configuring DHCP for IPv4

Cisco routers can be DHCP clients, servers, or relay agents. To configure an IOS device as a DHCP client, use the **ip address dhcp** command on the interface that needs to obtain the DHCP address. To configure a router as a DHCP server, you must create an IP address pool and assign a network or subnet to that pool. Exclude any static IP addresses within the pool, such as the router's address.

You can optionally add information, such as default gateway, DNS server, domain name, or lease duration. The Option statement provides additional information to clients; some typical options include Option 150 to locate a TFTP server for Cisco IP phones and Option 43 to locate a Wireless LAN Controller for access points.

Example 9-5 shows a router configured as a DHCP server, giving addresses from the 10.6.3.0/24 network. The IP address range of 10.6.3.1–10.6.3.5 is excluded from the pool. The clients' default gateway is 10.6.3.1. Option 150 tells the phones the address of the TFTP server where they will download their configuration.

Example 9-5 Configuring a DHCP Server

```
Router(config)# ip dhcp excluded-address 10.6.3.1 10.6.3.5
!
Router(config)# ip dhcp pool PHONES
Router(dhcp-config)# network 10.6.3.0 /24
Router(dhcp-config)# default-router 10.6.3.1
Router(dhcp-config)# option 150 10.6.2.2
```

IPv4 DHCP Relay Agent

Hosts discover their DHCP server by sending broadcasts. If the DHCP server is on a different subnet, those broadcasts must be routed to the server as unicasts. You can configure a router to relay DHCP messages with the **ip helper-address** interface command. It is important to understand that this command must be given on the interface that receives the host broadcasts. A Cisco DHCP relay agent functions as follows:

- Step 1.** A client broadcasts a DHCP request, which is seen by the IOS device (a router, for instance).
- Step 2.** The router changes the destination address of the packet to the unicast address of the server. It optionally adds option 82 (relay agent option) information.
- Step 3.** The router sends the unicast packet to the server.
- Step 4.** The server responds with the IP address and other parameters, such as the default gateway assigned to the client.
- Step 5.** The router gets the packet from the server, removes any option 82 information, and forwards it to the client.

Verify your DHCP configuration with the commands **show ip dhcp database**, **show ip dhcp server statistics**, and **show ip dhcp binding**. Delete address assignments with the **clear ip dhcp binding {address | *}** command.

DHCP for IPv6

As described in Chapter 2, “IPv6 Overview,” clients can use stateless autoconfiguration to obtain their network prefix and then use that prefix plus their MAC address to create a unique IPv6 address. DHCPv6 can be used instead of or in addition to stateless autoconfiguration. Two types of DHCPv6 are

- **Stateful DHCP:** When you need to control the prefixes given to clients

- **Stateless DHCP:** Used with stateless autoconfiguration to provide more information, such as domain name, DNS server, NTP server, or other options

Note

One big difference between DHCPv4 and DHCPv6 is that DHCPv6 servers do not give out IP addresses. They give out prefixes. The client then uses its MAC address along with the prefix to create its IPv6 address, just as it would with stateless autoconfiguration.

DHCPv6 Process

IPv6 clients first listen for router advertisement (RA) to determine whether they should use DHCP. If there is no router on the link, or if the RA specifies the use of DHCP, clients begin this process:

- Step 1.** The client sends a SOLICIT message to the all-DHCP-agents multicast address.
- Step 2.** All DHCP servers receiving the multicast answer with a unicast ADVERTISE message. This contains prefix and any other configured information, such as DNS server.
- Step 3.** The client responds to one of the DHCP servers (if there are multiple) with a REQUEST message, confirming its configuration parameters.
- Step 4.** The DHCP server responds with a REPLY message.

As with DHCPv4, there is a time limit on the prefix lease, and clients must renew the lease periodically.

A faster, two-step process called Rapid-Commit does exist that uses only SOLICIT and REPLY messages. This is useful when there is only one DHCP server or users are highly mobile. Rapid-Commit must be enabled on both clients and servers before it is used.

Configuring Stateful DHCPv6

The steps to configure DHCPv6 are similar to those for configuring DHCPv4. You create a pool, and specify all the configuration parameters under that pool. However, you do not specify a range of address—you configure a network prefix. There is no “excluded address” command because actual addresses are not given out. Another difference is that you must associate the pool with an interface. In Example 9-6, a pool named CCNP is configured to provide prefix and DNS server information, and the

lease length is set to 2 days. The DHCP pool is applied to e0/0, which is the client-facing interface on which requests will come in.

Example 9-6 Configuring Stateful DHCPv6

```
Router(config)# ipv6 dhcp pool CCNP
Router(config-dhcpv6)# address prefix 2001:db8:1::/64
Router(config-dhcpv6)# dns-server 2001:db8:2::1
Router(config-dhcpv6)# information refresh 2
!
Router(config)# int e0/0
Router(config-if)# ipv6 dhcp server CCNP
```

Configuring Stateless DHCPv6

Stateless DHCPv6, or DHCPv6 Lite, does not distribute either IP address or prefixes—clients obtain those via another method, usually by stateless auto-configuration. Stateless DHCP is used to provide other information, such as DNS server address or options. You still must configure a DHCP pool and associate it with an interface. However, there is an additional line of interface configuration to let clients know to look to the pool for additional information. That command is **ipv6 nd other-config-flag**. Example 9-7 shows a DHCP pool that furnishes DNS server and domain name information. The pool is then linked to the incoming interface.

Example 9-7 Configuring Stateless DHCPv6

```
Router(config)# ipv6 dhcp pool STATELESS
Router(config-dhcpv6)# dns-server 2001:db8:2::1
Router(config-dhcpv6)# domain-name ccnp.com
!
Router(config)# int e0/0
Router(config-if)# ipv6 dhcp server STATELESS
Router(config-if)# ipv6 nd other-config-flag
```

IPv6 DHCP Relay Agent

Hosts discover their DHCPv6 server by sending link-scoped multicasts, not broadcasts. However, if the DHCP server is on a different subnet, those multicasts must still be routed to the server as unicasts. Configure a router to relay DHCPv6 messages with the interface command **ipv6 dhcp relay destination address** [*source interface*]. Source interface must be specified in the command if the destination address given is a link-local address. If it is a global address, then no source interface is needed. As with DHCPv4, this command must be configured on the interface that receives the host broadcasts.

The following configuration shows the router configured to forward DHCPv6 multicasts from interface e0/1 to a DHCPv6 server at 2001:db8:3::1:

```
Router(config)# int e0/1
Router(config-if)# ipv6 dhcp relay destination 2001:db8:3::1
```

NetFlow

NetFlow is an IOS feature that captures statistics on IP traffic flows through Cisco devices. It is beneficial in monitoring applications and users, which helps in such tasks as identifying security risks, network optimizing, troubleshooting, chargeback billing, and data mining. NetFlow collects data about unidirectional flows of traffic between sources and destinations. It uses the following seven fields to identify a unique flow:

- Source IP address
- Source port number
- Destination IP address
- Destination port number
- Layer 3 protocol type
- Type of service
- Logical input interface

NetFlow creates a flow record for each traffic flow. There have been several formats for flow records, but the current ones in use are version 5 and version 9. Version 5 added BGP support. Version 9, which supports IPv6, is the basis for an IETF standard for information export and is based on a template that makes it easily adaptable to new protocols. However, it is not backward compatible with any previous version.

Two main components of NetFlow are

- **NetFlow cache:** The flow record for each active flow is maintained in the cache.
- **NetFlow export:** Information about flows that are no longer active are grouped into export datagrams to be exported via UDP to an external NetFlow collector.

To begin collecting NetFlow data, enable it to capture ingress and/or egress traffic on at least one interface. Then configure a destination for exporting the data to, the UDP port number to use, and optionally set the version number. Example 9-8 shows a basic IPv4 NetFlow configuration.

Example 9-8 NetFlow Configuration

```
Router(config)# int e0/0
Router(config-if)# ip flow ingress
Router(config-if)# ip flow egress
!
Router(config)# ip flow-export destination 10.1.1.1 2000
Router(config)# ip flow-export version 9
```

Verify NetFlow configuration and action with the commands **show ip flow interface**, **show ip cache flow**, and **show ip flow export**.

Network Address Translation

Network Address Translation (NAT) replaces the original IP address in the IP header with a different one on outgoing traffic and then places the original IP address into the header on incoming traffic. This hides actual device IP addresses from the world outside the NAT device. It is commonly used to translate the private IP addresses used inside a network to global, public addresses for Internet access. It can also be used to translate internal network addresses when there is an overlapping IP address range in two portions of the same network. There is even a type of NAT, called NAT64, that can translate between IPv4 and IPv6 addresses.

NAT performs three types of translation:

- **Static NAT:** Each device has a manually configured one-to-one mapping between its inside and outside address.
- **Dynamic NAT:** Each device gets a unique outside address from a pool of many addresses.
- **PAT (Port Address Translation):** Each inside device gets translated to the same (or one of a few) outside address. PAT uses different port numbers to distinguish between the inside hosts.

NAT uses four terms to refer to host addresses:

- **Inside local:** The IP address assigned to a host on the internal network. This is the address that is typically translated.

- **Inside global:** The IP address of the internal host as it appears to the external network; what the rest of the world thinks is the host's address.
- **Outside local:** What an internal device thinks is the IP address of an external device; it might have been translated or it might actually be the device's address.
- **Outside global:** The IP address actually assigned to the external device.

Configuring Traditional NAT for IPv4

The first step in configuring traditional NAT is to identify which interfaces the traffic to be translated will come in on, and which interface it will exit. The ingress interfaces are designated as “inside” and the egress interface is designated as “outside.” You use the interface command **ip nat [inside | outside]** to configure this, as shown in Example 9-9. Interface Ethernet 0/0 is the ingress interface, where the traffic to be translated enters the router. Interface Ethernet 0/2 is the egress interface, where that traffic will exit the router.

Example 9-9 Configuring NAT Inside and Outside Interfaces

```
interface Ethernet0/0
  description LAN Interface
  ip address 192.168.10.1 255.255.255.0
  ip nat inside
!
interface Ethernet0/2
  description WAN Interface
  ip address 209.165.201.1 255.255.255.224
  ip nat outside
```

Static NAT

In static NAT, one inside local address is mapped to one inside global address. This might be used when you have a server with an inside address that must be reached from the outside. You always want it to have the same global IP address. Once you have the inside and outside interfaces configured, use the global command:

```
ip nat inside source static inside-local-addr inside-global addr
```

Dynamic NAT

Use Dynamic NAT when you have a pool of global addresses that will be used by local hosts. One global address is assigned to each local host, but it is done dynamically. As before, identify the inside and outside interfaces. Then create an access list that permits the inside local address range to be translated. Then create a pool that lists the starting and ending addresses to be used as inside global addresses. Link the two together with the **ip nat inside source** command.

The following example assumes that the inside and outside interfaces have been configured. It shows an access list that permits the range 192.168.10.0/24—these are the local addresses to be translated. The NAT pool named DYNAMIC specifies 11 addresses to be used as global addresses. They are assigned to the first 11 hosts that send traffic from interface e0/0 bound out of interface e0/2. The third command tells the router to translate traffic with source addresses matching access list 1, coming in the interfaces marked as “inside,” to an address in the pool DYNAMIC.

```
Router(config)# access-list 1 permit 192.168.10.0 0.0.0.255
Router(config)# ip nat pool DYNAMIC 209.165.201.10 209.165.201.20
prefix-length 26
Router(config)# ip nat inside source list 1 pool DYNAMIC
```

PAT

PAT is also called *NAT overload*. Instead of only translating IP addresses, PAT also translates the source port as necessary. Each translation is identified by both its IP address and its port number. Thus, one global address can represent multiple local hosts. The keyword **overload** tells the router to use PAT.

Start by identifying the inside and outside address as was done previously. You can create an address pool and translate multiple hosts to them, but typically you use the address of the Internet-facing interface instead.

The following example assumes that the inside and outside interfaces have been configured. It shows an access list that permits the range 192.168.10.0/24—these are the local addresses to be translated. No pool is created; all traffic uses the IP address of interface e0/2 as its source address.

```
Router(config)# access-list 1 permit 192.168.10.0 0.0.0.255
Router(config)# ip nat inside source list 1 int e0/2 overload
```

Verify your NAT and PAT configurations with the commands **show ip nat translations** and **show ip nat statistics**.

Configuring NAT Virtual Interface

In more recent IOS versions, you do not have to identify the inside and outside interfaces. Instead, just enable interfaces for NAT. The router creates a NAT Virtual Interface (NVI). When traffic comes in a NAT-enabled interface it is checked against the NAT translation table. If there is a match, then the traffic is routed to the NVI interface where it is translated. It is then routed to its exit interface.

Enable an interface to use NVI with the interface command **ip nat enable**. The actual translation commands are slightly different also, in that they omit the **inside** and **outside** keywords.

The NVI command equivalent to the static NAT command shown in the “Static NAT” section is

```
ip nat source static inside-local-addr inside-global-addr
```

The NVI command to use an access list and a NAT pool to do the dynamic NAT translation shown in the “Dynamic NAT section is

```
Router(config)# ip nat source list 1 pool DYNAMIC
```

Example 9-10 shows the configuration for NVI with PAT, using source addresses from access list 1 and translating to the IP address of interface e0/2. Note that the router has automatically created an NVI interface and given it the IP address of interface e0/0.

Example 9-10 Configuring NVI with PAT

```
Router(config)# access-list 1 permit 192.168.10.0 0.0.0.255
```

```
Router(config)# ip nat source list 1 int e0/2 overload
```

```
!
```

```
Router(config)# int e0/0
```

```
Router(config-if)# ip nat enable
```

```
Router(config)# int e0/2
```

```
Router(config-if)# ip nat enable
```

```
!
```

```
Router# show ip interface brief
```

Interface	IP-Address	OK?	Method	Status	Protocol
Ethernet0/0	192.168.10.1	YES	NVRAM	up	up
Ethernet0/1	unassigned	YES	NVRAM	down	down
Ethernet0/2	209.165.201.1	YES	NVRAM	up	up
NVI0	192.168.10.1	YES	unset	up	up

Verify your NVI configuration with the commands **show ip nat nvi translations** and **show ip nat nvi statistics** (notice the addition of the keyword **nvi**).

NAT64

Network Address Translation IPv6 to IPv4 (NAT64) enables IPv6 hosts to communicate with IPv4 hosts, and vice versa. This is accomplished by translating between IPv6 and IPv4 addresses, and also between the IPv4 and IPv6 header formats. It supports two types of translation:

- **Stateless NAT64:** Performs one-to-one IP address mapping of IPv6 addresses to IPv4 addresses
- **Stateful NAT64:** Performs one-to-many IP address mapping, similar to PAT in NAT for IPv4

NAT64 provides a migration path for enterprises building out IPv6 sections of their network. Hosts running either protocol can reach the same services without requiring them to be dual-stacked and without running IPv6 routing in the whole network .