CCIE Routing and Switching v4.0 Quick Reference

Brad Ellis
Jacob Uecker

Cisco Press

CCIE Routing and Switching v4.0

Quick Reference

Brad Ellis
Jacob Uecker
Steven Means

ciscopress.com

Table of Contents

Chapter 1

General Networking Theory

General Routing Concepts

Link-State and Distance Vector Protocols

Distance Vector

Examples: Routing Information Protocol Version 1 (RIPv1), RIPv2, Interior Gateway Routing Protocol (IGRP)

- Features periodic transmission of entire routing tables to directly connected neighbors
- Mathematically compares routes using some measurement of distance
- Features hop-count limitation

Link State

Examples: Open Shortest Path First (OSPF), Intermediate System-to-Intermediate System (IS-IS)

- Sends local connection information to all nodes in the internetwork.
- Forms adjacencies with neighboring routers that speak the same protocol; sends local link information to these devices.
- Although this floods of information to all nodes, the router sends only the portion of information that deals with the state of its own links.
- Each router constructs its own complete “picture” or “map” of the network from all the information received.

 Please see page 245 for more details.

Hybrid

- Example: Enhanced Interior Gateway Routing Protocol (EIGRP)
- Features properties of both distance vector and link-state routing protocols

Path Vector Protocol

Example: Border Gateway Protocol (BGP)

- Path vector protocols are a subset of distance vector protocols; BGP uses path vectors or a list of all the autonomous systems a prefix has crossed to make metric decisions and to ensure a loop-free environment.
- In addition to the autonomous system path list, an administrator can use many other factors to affect the forwarding or receipt of traffic using BGP.

Split Horizon

- Routing protocols use the Split horizon technique to help prevent routing loops. The split-horizon rule states that an interface will not send routing information out an interface from which the routing information was originally received. Split horizon can cause problems in some topologies, such as hub-and-spoke Frame Relay configurations.

Summarization

Summarization is the process in which the administrator collapses many routes with a long mask to form another route with a shorter mask. Route summarization reduces the size of routing tables and makes the routing function more efficient. Route summarization also helps to make networks more stable by reducing the number of updates sent when subnets change state. Route summarization makes classless interdomain routing (CIDR) possible. Variable-length subnet masking (VLSM) promotes the use of route summarization. Some dynamic routing protocols engage in route summarization automatically for changes in a major classful network, whereas others do not. For any routing protocol within the scope of the CCIE written exam, an administrator can disable any automatic summarization that might occur and configure manual summarization.

To engage in route summarization, find all the left-most bits that are in common and create a mask that encompasses them. An example follows.

 Please see page 245 for more details.

The following routes exist in the routing table—all routes use a 24-bit mask:

10.108.48.0 = 00001010 01101100 00110000 00000000
10.108.49.0 = 00001010 01101100 00110001 00000000
10.108.50.0 = 00001010 01101100 00110010 00000000
10.108.51.0 = 00001010 01101100 00110011 00000000
10.108.52.0 = 00001010 01101100 00110100 00000000
10.108.53.0 = 00001010 01101100 00110101 00000000
10.108.54.0 = 00001010 01101100 00110110 00000000
10.108.55.0 = 00001010 01101100 00110111 00000000

Notice that the first 21 bits of the subnetwork IDs are all common. These can be masked off. You can use the single route entry for all these subnetworks as follows:

10.108.48.0/21

Classful and Classless Routing Protocols

Classful routing protocols are considered legacy and do not include subnet mask information with routing updates. Examples of classful routing protocols are RIPv1 and IGRP. Because subnet mask information is not included in updates, consistency of the mask is assumed throughout the network. Classful routing protocols also feature automatic summarization of routing updates when sent across a major classful network boundary. For example, the 10.16.0.0/16 network would be advertised as 10.0.0.0/8 when sent into a 172.16.0.0 domain.

Although BGP and EIGRP are not classful routing protocols, both engage in automatic summarization behavior by default, and in that sense they act classful. The **no auto-summary** command is used to disable this behavior.

Classful routing protocols feature a fixed-length subnet mask (FLSM) because of their inherent limitations. The FLSM leads to inefficient use of addresses and limits the network's overall routing efficiency.

By default, classful routing protocols discard traffic bound for any unknown subnet of the major classful network. For example, if your classful routing protocol receives traffic destined for 10.16.0.0 and it knows of only the 10.8.0.0 and 10.4.0.0 subnets in its routing table, it discards the traffic—*even if a default route is present*! The **ip classless**

 Please see page 245 for more details.

command was introduced to change this behavior. The **ip classless** command enables the protocol to use the default route in this case. This command is on by default with Cisco IOS Release 12.0 and later routers.

As a classic example of a classless routing protocol, OSPF carries subnet mask information in updates. Wireless LAN Services Module (WLSM) is possible with such protocols.

Routing Decision Criteria

Routers must determine the best route to send traffic on toward its destination. This is accomplished as follows (note that the order of operations is critical and fixed):

1. **Valid next-hop IP address:** When updates are received, the router first verifies that the next-hop IP address to reach the potential destination is valid.
2. **Metric:** The router then examines the metrics for the various routes that might exist from a particular protocol. For example, if OSPF has several routes to the destination, the router tries to install the route with the best metric (in this case, cost) into the routing table.
3. **Administrative distance:** If multiple routing protocols run on the device, and multiple protocols all present routes to the destination with valid next hops, the router examines administrative distance. The route sourced from the lowest administrative distance protocol or mechanism is installed in the routing table.
4. **Prefix:** The router examines the route's prefix length. If no exact match exists in the routing table, the route is installed. This might cause the routing table to fill with the following entries: EIGRP 172.16.2.0/24 and RIP 172.16.2.0/19.

For the prefix length and the routing table, remember that when a router looks for a match in the IP routing table for the destination address, it always looks for the longest possible prefix match. For example, if the routing table contains entries of 10.0.0.0/8, 10.2.0.0/16, and 10.2.1.0/24, and your traffic is destined for 10.2.1.0/24, the longest match prefix is selected. This prefix length rule trumps administrative distance. So a /24 prefix learned via EIGRP would be preferred over a /16 added as a static route despite the static route having a superior administrative distance.

 Please see page 245 for more details.

Routing Information Base and Routing Protocol Interaction

Administrative Distance

If a router learns of a network from multiple sources (routing protocols or static configurations), it uses the administrative distance value to determine which route to install in the routing (forwarding) table. The default administrative distance values are listed here.

Source	Administrative Distance
Connected interface	0
Static route	1
EIGRP summary route	5
External BGP	20
Internal EIGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
Exterior Gateway Protocol	140
On-demand routing	160
External EIGRP	170
Internal BGP	200
Unknown	255

Administrators can create static routes that float. A floating static route means the administrator increases the administrative distance of the static route to be greater than the default of 1. For example, if you run EIGRP on your network, the AD of a static route could be increased to 95. This would mean the static route would be used only when a dynamic EIGRP route did not exist.

 Please see page 245 for more details.

Routing Table

The routing table has been the principal element of IP routing and the primary goal of routing protocols to build and maintain for most of modern internetworking. The main routing table model, the hop-by-hop routing paradigm, has the routing table list for each destination network of the next-hop address to reach that destination. If the routing tables are consistent and accurate, with no misinformation, this simple hop-by-hop paradigm works well enough to deliver data to anywhere from anywhere in the network. In recent practice, this simple hop-by-hop model is abandoned for new technologies such as Multiprotocol Label Switching (MPLS). These technologies enable a simple and efficient label lookup to dictate the next hop that data should follow to reach a specific destination. Although this determination can be based on the routing table information, it can easily be based on other parameters, such as quality of service (QoS) or other traffic engineering considerations. MPLS is explored in its own chapter of this Q.

Routing Information Base and Forwarding Information Base Interaction

The routing and forwarding architecture in Cisco routers and multilayer switches used to be a centralized, cache-based system that combined a control plane and a data plane. The control plane refers to the resources and technologies that create and maintain the routing table. The data plane refers to those resources and technologies needed to actually move data from the ingress port to the egress port on the device. This centralized architecture has migrated so that the two planes can separate to enhance scalability and availability in the routing environment.

The separation of routing and forwarding tasks has created the Routing Information Base (RIB) and the Forwarding Information Base (FIB). The RIB operates in software, and the control plane resources take the best routes from the RIB and place them in the FIB. The FIB resides in faster hardware resources. The Cisco implementation of this enhanced routing and forwarding architecture is called Cisco Express Forwarding (CEF).

Redistribution

Redistribution Between Routing Protocols

Route redistribution might be required in an internetwork because multiple routing protocols must coexist. Multiple routing protocols might be a necessity because of an interim period during conversion from one to another, application-specific protocol requirements, political reasons, or a lack of multivendor interoperability.

 Please see page 245 for more details.

A major issue with redistribution is the seed metric used when the routes enter the new routing protocol. Normally, the seed metric is generated from the originating interface. For example, EIGRP would use the bandwidth and delay of the originating interface to seed the metric. With redistributed routes, however, these routes are not connected to the router. Some routing protocols feature a default seed metric for redistribution, whereas others do not. Following is a list of the defaults for the various protocols. Infinity indicates a seed metric must be configured; otherwise, the receiving protocol will not use the route.

Protocol	Default Seed Metric
OSPF	20; except BGP, which is 1
IS-IS	0
RIP	Infinity
IGRP/EIGRP	Infinity

Redistribution Into RIP

Remember to set a default metric, using either the **redistribute** command or the **default-metric** command. Following is the command to redistribute routes into RIP:

```
redistribute protocol [process-id] [match route-type]
[metric metric-value] [route-map map-tag]
```

The **match** keyword enables you to match certain route types when redistributing OSPF. For example, you can specify internal, external 1, or external 2. The **route-map** keyword enables you to specify a route map for controlling or altering the routes that are redistributed.

Redistribution Into OSPF

The default seed metric is 20. The default metric type for redistributed routes is External Type 2 (E2), meaning the metric reflects only the cost from the redistributing router to the destination regardless of the path cost within the

 Please see page 245 for more details.

OSPF network. Type 1 (e1) can be optionally used, which means the metric will be based on the total path to the destination. Subnets are not
redistributed by default. Following is the command for redistribution into OSPF:

```
redistribute protocol [process-id] [metric metric-value] [metric-type type-value] [route-map map-tag]
\[subnets]

[tag tag-value]
```

The **subnets** keyword is critical in this command and specifies that subnets should indeed be redistributed. The **tag** value enables the administrator to configure an optional tag value that can be used later to easily identify these routes.

Redistribution into EIGRP

Remember that like RIP, you must set a default seed metric when redistributing into EIGRP. Following is the command for redistribution into EIGRP:

```
redistribute protocol [process-id] [match {internal | external 1 | external 2}]
[metric metric-value] [route-map map-tag]
```

Troubleshooting Routing Loops

You can perform one-way or two-way redistributions. You can also perform redistribution in multiple locations throughout the topology.

With one-way redistribution, you typically pass a default route into the edge protocol, and take all the edge protocol routes and redistribute them into the core protocol of the network.

With two-way redistribution, all routes from each routing protocol pass into each other. If two-way redistribution is

 Please see page 245 for more details.

performed in multiple areas in the network, an excellent chance exists for route feedback and routing loops. Routing loops are likely to
occur because routing information from one autonomous system can easily be passed back into that same autonomous system.

The safest way to eliminate the chance for a loop is to redistribute only in one direction (one-way redistribution). If this is not possible, and two-way redistribution is wanted, try these techniques to ensure a lack of loops:

- Redistribute from the core protocol into the edge with filtering to block routes native to the edge.
- Apply two-way redistribution on all routes, and manipulate administrative distance associated with the external routes so that they are not selected when multiple routes exist for the same destination.

An excellent technique to detect a routing loop during redistribution is to use the **debug ip routing** command. This command shows all routing table activity as it occurs and demonstrates a loop condition through routing table instability. In a stable
network, little to no output occurs.

 Please see page 245 for more details.

Chapter 2

Bridging and LAN Switching

Spanning Tree Protocol

802.1D

802.1D Spanning Tree Protocol (STP) is a Layer 2 loop-prevention mechanism. It is an IEEE standards-based protocol. Over the years, Cisco enhanced this protocol with new features to make much-needed improvements. This chapter discusses those improvements and new IEEE versions of the protocol that dramatically improve the technology. Layer 2 loops are terrible because of no Time To Live (TTL) value in frames. Loops can cause broadcast storms, MAC table corruption, and multiple-frame copies.

STP Process

The bridge ID (BID) is a critical element for the creation of the spanning-tree, loop-free topology. The bridge ID consists of a 2-byte bridge priority and a 6-byte MAC address. The default priority is 32,768. Newer switch operating systems break the priority field into two sections: the 4-bit priority and a 12-bit extended system ID. This extended system ID value is just the VLAN ID. This enables each VLAN to have a unique bridge ID while still using the same MAC address and priority value. Previously, multiple MAC addresses were needed for each VLAN to ensure uniqueness.

Path cost is the measure of distance from one bridge to another. Links are assigned a cost value by STP. This cost value is based on bandwidth. Higher-bandwidth links receive a lower-cost value, and STP deems a lower-cost path as preferred to a higher-cost path.

Initially with STP operations, a root bridge must be selected. This root bridge will have all its ports in the forwarding state (designated ports) and will be the central reference point for the creation of a loop-free Layer 2 topology. For

 Please see page 245 for more details.

the "election" of this device, configuration bridge protocol data units (BPDU) are sent between switches for each port and BIDs are compared. The switch with the lowest priority will be the root bridge. If a tie occurs, the switch with the lowest MAC address will be the root bridge.

After the root bridge for the network has been determined, this reference point can create the loop-free topology. This initial creation of the loop-free topology takes place in three steps:

Step 1. Elect a root bridge. The lowest BID wins.

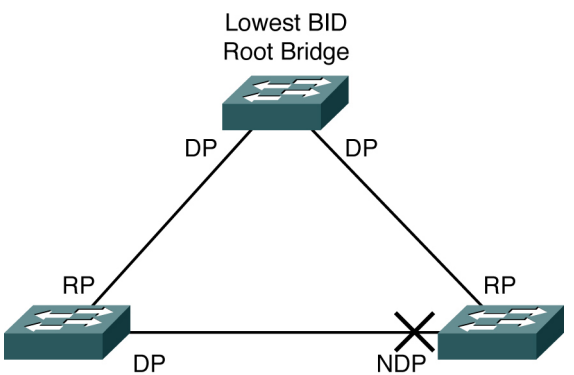
Step 2. Elect root ports. Every nonroot bridge selects one root port.

Step 3. Elect designated ports. Each segment has one designated port (the bridge with the designated port is the designated bridge for that segment); all active ports on the root bridge are designated (unless you connect two ports to each other).

When convergence occurs, BPDUs radiate out from the root bridge over loop-free paths. Figure 2-1 shows an example of STP in action.

Ports have a port state under 802.1D STP. Ports begin life on the switch as disabled and gradually transition to a forwarding state when STP deems it is safe to do so. The possible states are listed here along with the timers that control the transition times. The states are carefully ordered to demonstrate the order of transition:

FIGURE 2-1
Spanning-Tree Topology



 Please see page 245 for more details.

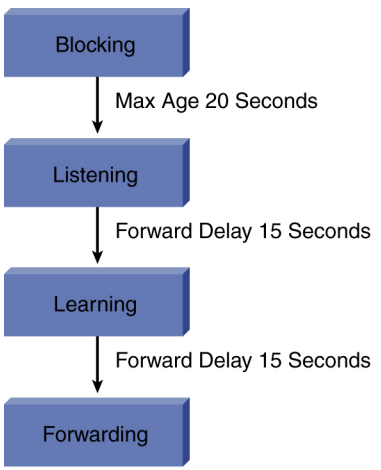
1. **Disabled:** Administratively down
2. **Blocking:** BPDUs received only (20 sec)
3. **Listening:** BPDUs sent and received (15 sec)
4. **Learning:** Bridging table is built (15 sec)
5. **Forwarding:** Sending/receiving data

STP timers control convergence in the process:

- **Hello:** 2 sec (time between each configuration BPDU)
- **Forward Delay:** 15 sec (controls durations of listening/learning states)
- **Max Age**: 20 sec (controls the duration of the blocking state)

Default convergence time is 30 to 50 seconds. Timer modification is possible from the root bridge. See Figure 2-2.

FIGURE 2-2
802.1D Timers



 Please see page 245 for more details.

Although the timers can be manipulated, Cisco does not recommend this. Instead, Cisco mechanisms can improve convergence times without direct manipulation of the timers by the administrator. Convergence time is a recognized issue with STP and the exact reason for IEEE's creation of new versions of the protocol.

Topology Changes

STP uses a Topology Change Notification (TCN) BPDU to alert the root bridge that a topology change to the spanning tree might need to occur. The Type field of the BPDU signifies the TCN BPDU: 0x80. TCN BPDUs improve convergence time when failures in the network occur—primarily because they help in a rapid updating of the MAC address tables.

The TCN process of 802.1D is as follows:

1. A bridge sends a TCN BPDU in two cases:
   a. It takes a port into forwarding and has at least one designated port (DP).
   b. A port goes from Forwarding/Learning to Blocking.
   c. TCNs are sent out the root port of nonroot devices; they are sent each hello interval until they are acknowledged by the upstream device.
2. Upstream bridges process TCN on DPs.
3. The upstream switch sets the Topology Change Acknowledgment (TCA) field of the next configuration BPDU received and sends this downstream. This causes the downstream switch to stop sending TCN BPDUs.
4. The upstream switch then sends the TCN further upstream.
5. This continues until the root bridge receives the TCN.
6. The root bridge then sets the TCA and Topology Change flags in the next configuration BPDU sent out downstream.
7. The root bridge sets the TC flag in all BPDUs sent for Forward Delay + Max Age. This instructs all switches to age MAC table address entries faster.

Note
The CCIE written exam focuses on the Cisco IOS-based command set. As a result, no CatOS commands are shown in any of the Quick Reference Sheets.

 Please see page 245 for more details.

Root Bridge Placement

You need to set the root bridge location in your network using the appropriate Cisco IOS command.

You should also select a secondary root if the primary root fails.

`spanning-tree vlan vlan_ID priority priority_value` enables you to modify the priority value and directly manipulate the root election. For example, spanning-tree vlan 100 priority 4096 sets the priority to 4096 for VLAN 100 on the local switch. If all switches are at the default priority value of 32,768, the bridge becomes the root. You can use the priority value of 8192 in this case on another switch to elect it as the secondary root bridge.

The command **`spanning-tree vlan vlan_ID root primary`** is actually a macro command that examines the priority of the existing root and sets the priority on the local switch to be 1 less. If the default is used on the root, the priority is set to 8192. To create a secondary root, you can use the following command:

```
spanning-tree vlan vlan_ID root secondary
```

This command sets the priority value to 16,384.

Remember, in a Cisco environment, by default all spanning-tree mechanisms occur on a VLAN-by-VLAN basis, which is Per-VLAN Spanning Tree (PVST+).

Fast STP Convergence with Cisco-Proprietary Enhancements to 802.1D

PortFast

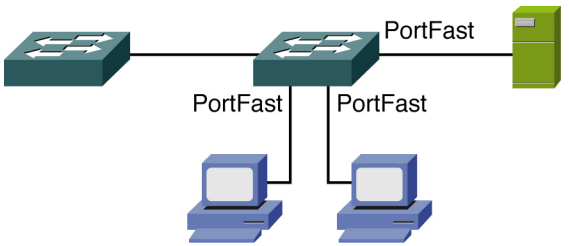
PortFast, as shown in Figure 2-3, is a Cisco enhancement to the 802.1D STP implementation. You apply the command to specific ports, and that application has two effects:

- Ports coming up are put directly into the forwarding STP mode.
- The switch does not generate a TCN when a port configured for PortFast is going up or down—for example, when a workstation power-cycles.

Therefore, consider enabling PortFast on ports connected to end-user workstations. Use caution with PortFast ports to ensure that hubs, switches, bridges, or any other device that might cause a loop do not connect to these ports.

 Please see page 245 for more details.

FIGURE 2-3
PortFast



UplinkFast

Configure UplinkFast on wiring closet switches, which detects a directly connected failure and enables a new root port to come up almost immediately.

When you configure UplinkFast, the local switch has a priority set to 49,152 and adds 3000 to the cost of all links. Finally, a mechanism is included that causes the manipulation of MAC address tables for other bridges.

BackboneFast

Configure BackboneFast on all switches to speed convergence when the failure occurs and is indirectly located, such as in the core of the backbone. It reduces convergence from approximately 50 seconds to approximately 30 seconds.

802.1w Rapid Spanning Tree Protocol

Rapid Spanning Tree Protocol (RSTP or IEEE 802.1w) improves on 802.1D. The protocol incorporates many new features to speed convergence, including incorporation of the ideas presented by Cisco in its enhancements to 802.1D. Although the new technology has many improvements, , the configuration remains almost identical—and the two technologies can coexist. Full benefits are not realized until all systems run RSTP, however.

RSTP requires full-duplex, point-to-point connections between adjacent switches to achieve fast convergence.

RSTP defines edge ports as those not participating in STP. Edge ports can be statically configured or will be recognized by the PortFast configuration command.

 Please see page 245 for more details.

RSTP Port States

RSTP port states are simplified from 802.1D and consist of the following:

- Discarding
- Learning
- Forwarding

Also, the port states are no longer tied directly to port roles. For example, a DP could be Discarding, even though it is destined to transition to the Forwarding state.

RSTP Port Roles

- **Root port:** This port role exists in 802.1D, too, and is the best path back to the root bridge; it must exist on all nonroot bridges.
- **Designated port:** This port role exists in 802.1D, too, and there must be a DP on all segments in the topology. By default, all ports on the root bridge are DPs.
- **Alternative port:** This port role is new to 802.1w and is a quickly converging backup port to the current DP on a segment.
- **Backup port:** This port role is new to 802.1w and is a quickly converging backup to the root port for a system.

RSTP BPDUs

All bridges now send BPDUs every hello time period (2 seconds by default). The BPDUs now act as a keepalive; protocol information is aged if no BPDUs are heard for three consecutive hello times.

RSTP proposal and agreement process/topology change mechanism

Convergence occurs on a link-by-link basis in 802.1w. No longer does a reliance on timers for convergence exist as in 802.1D. A proposal and agreement process replaces the timer methodology of STP and flows downstream from the root device.

 Please see page 245 for more details.

In RSTP, only nonedge ports moving to the Forwarding state cause a topology change (TC). The originator of a TC is now responsible for flooding it through the network.

Implementing RSTP

On most Cisco switches, configuring 802.1s (Multiple Spanning Tree, MST) automatically enables RSTP. Cisco did invent a mode of operation, PVST+ mode, that enables you to use RSTP without the implementation of MST. You can enable PVST+ mode on a switch with the following command:

```
spanning-tree mode rapid-pvst
```

802.1s Multiple Spanning Tree

MSTP (IEEE 802.1s) is an IEEE standard that enables several VLANs to be mapped to a reduced number of spanning-tree instances. This provides advantages over PVST+ because typical topologies need only a few spanning-tree topologies to be optimized.

You configure a set of switches with the same MISTP parameters, and this becomes an MST region. With MISTP, you have an internal spanning tree capable of representing the entire MST region as a common spanning tree for backward compatibility with earlier IEEE implementations.

Follow these steps to configure MISTP:

Step 1. Globally enable MISTP (MSTP) on your switches:

```
spanning-tree mode mst
```

Step 2. Enter MST configuration submode:

```
spanning-tree mst configuration
```

Step 3. Set the MST region name:

```
name name
```

Step 4. Set a configuration revision number:

 Please see page 245 for more details.

```
revision rev_num
```

Step 5. Map your VLANs to MST instances:

```
instance int vlan range
```

You can easily verify an MSTP configuration using the following commands:

```
show spanning-tree mst configuration
show spanning-tree mst vlan_id
```

Loop Guard

As its name implies, Loop Guard is a method for ensuring that STP loops never occur in a particular topology. Even though STP guards against such loops, they can still occur because of things such as unidirectional link failures or switch congestion issues.

Loop Guard prevents loops conservatively by preventing alternative or root ports from becoming DPs in the topology. If BPDUs are not received on a non-DP, and Loop Guard is enabled and that port moves into the STP loop-inconsistent Blocking state instead of the Listening/Learning/Forwarding state.

Loop Guard operates only on ports considered point-to-point by the spanning tree and cannot be run with Root Guard on an interface.

To enable Loop Guard, use the following global configuration mode command:

```
spanning-tree loopguard default
```

Unidirectional Link Detection

Unidirectional Link Detection (UDLD), as shown in Figure 2-4, detects and disables unidirectional links. A unidirectional link occurs when traffic transmitted from the local switch is received by the neighbor, but traffic sent from the neighbor is not. Unidirectional links can cause a variety of problems, including spanning-tree loops. UDLD performs tasks that autonegotiation cannot perform.

 Please see page 245 for more details.

FIGURE 2-4
UDLD

To perform UDLD, packets are sent to neighbor devices on interfaces with UDLD enabled. Therefore, both sides of the link must support UDLD. By default, UDLD is locally disabled on copper interfaces and is locally enabled on all Ethernet fiber-optic interfaces. Following is the Cisco IOS command to enable UDLD on an interface:

```
udld enable
```

Root Guard

Root Guard enables an administrator to enforce the root bridge placement in the network. Service providers that connect switches to customer networks are often interested in this technology because they want to ensure that no customer device inadvertently or otherwise becomes the root of the spanning tree. Root Guard ensures that the port on which Root Guard is enabled is the DP. If the switch receives superior STP BPDUs on a Root Guard-enabled port, the port is moved to a root-inconsistent STP state. This root-inconsistent state is effectively equal to the Listening port state. No traffic is forwarded across this port. This protects the current placement of the root bridge in the infrastructure.

You can enable this feature on a port with the following interface configuration command:

```
spanning-tree guard root
```

BPDU Guard

This Cisco STP feature protects the network from loops that might occur if BPDUs were received on a PortFast port. Because BPDUs should never arrive at these ports, their reception indicates a misconfiguration or a security breach. BPDU Guard causes the port to error-disable upon the reception of these frames.

You can configure BPDU Guard globally to have the feature enabled for all PortFast ports on the system. Following is the command to do this:

 Please see page 245 for more details.

```
spanning-tree portfast bpduguard
```

You can also enable the feature at the interface level. Use this command:

```
spanning-tree bpduguard enable
```

You can enable this feature at the interface level even if PortFast is not enabled on the port. Again, the receipt of a BPDU causes the port to error-disable.

Storm Control

The Storm Control feature protects a LAN from being affected by unicast, broadcast, or multicast storms that might develop. The switch implements storm control by counting the number of packets of a specified type received within the one-second time interval and compares the measurement with a predefined suppression-level threshold. Storm Control can typically enable the administrator to control traffic by a percentage of total bandwidth or the traffic rate at which packets are received. When the rate of multicast traffic exceeds a set threshold, all incoming traffic (broadcast, multicast, and unicast) is dropped until the level drops below the specified threshold level. Only spanning-tree packets are forwarded in this situation. When broadcast and unicast thresholds are exceeded, traffic is blocked for only the type of traffic that exceeded the threshold.

Storm Control is configured at the interface level with the following command:

```
storm-control {broadcast | multicast | unicast} level {level [level-low] | pps pps [pps-low]}
```

Unicast Flooding

If a destination MAC address is not in the MAC address table of the switch, the frame is flooded out all ports for that respective VLAN. Although some flooding is unavoidable and expected, excessive flooding might be caused by asymmetric routing, STP topology changes, or forwarding table overflow. Also, flooding can result from attacks on the network, especially if denial-of-service (DoS) attacks occur.

Switches can now implement a unicast flood-prevention feature. This is implemented through the following global configuration command:

 Please see page 245 for more details.

```
mac-address-table unicast-flood {limit kfps} {vlan vlan} {filter timeout | alert | shutdown}
```

An alternative configuration approach found on some Catalyst model devices (such as the 6500 series) is to use Unknown Unicast Flood Blocking (UUFB), which is configured with the following simple interface command:

```
switchport block unicast
```

LAN Switching

DTP

Dynamic Trunking Protocol (DTP) is a Cisco proprietary protocol that negotiates the trunking status of a switchport. Connected switches exchange DTP messages that indicate their desirability to create a trunk. The DTP port state dictates its capability to create a trunk. Following are the possible states:

Production: Formatted below as bulleted list; however, icon doesn't appear. San Dee

auto: Enables the switch to create a trunk if initiated from the other switch. A switch programmed with auto does not initiate a trunk but can form a trunk if the other side initiates. The trunk is formed with desirable and **on**.

desirable: Actively tries to create a trunk link with the peer. The trunk is formed with auto, **desirable**, and **on**.

on: DTP messages are sent, and a trunk will be formed unless the peer explicitly forbids it. The trunk is formed with **auto**, **desirable**, and **on**.

off: Trunking is not allowed on the switchport regardless of the DTP status of the peer.

nonegotiate: Disables DTP and will not form a trunk link with a peer which requires trunk negotiation. Trunk is formed with **on** and **nonegotiate.**

VLAN Trunking

802.1Q

The IEEE 802.1Q standard trunking protocol uses an extra tag in the MAC header to identify the VLAN membership

 Please see page 245 for more details.

of a frame across bridges. This tag is used for VLAN and quality of service (QoS) priority identification.

The VLAN ID (VID) associates a frame with a specific VLAN and provides the information that switches need to process the frame across the network. Notice that a tagged frame is 4 bytes longer than an untagged frame and contains 2 bytes of Tag Protocol Identifier (TPID) and 2 bytes of Tag Control Information (TCI). These components of an 802.1Q tagged frame are described in more detail here:

- **TPID:** The Tag Protocol Identifier has a defined value of 8100 in hex; with the EtherType set at 8100, this frame is identified as carrying the IEEE 802.1Q/802.1p tag.
- **Priority:** The first 3 bits of the Tag Control Information define user priority; notice the eight (23) possible priority levels; IEEE 802.1p defines the operation for these 3 user-priority bits.
- **CFI:** The Canonical Format Indicator is a single-bit flag, always set to 0 for Ethernet switches. CFI is used for compatibility reasons between Ethernet networks and the Token Ring.
- **VID:** VLAN ID identifies the VLAN; notice it enables the identification of 4096 (2^{12}) VLANs. Two of these identifications are reserved, permitting the creation of 4094 VLANs.

802.1Q trunks feature a concept called the native VLAN. The native VLAN is a VLAN for which frames are not tagged. Following are the aspects of the native VLAN:

- The VLAN a port is in when not trunking.
- The VLAN from which frames are sent untagged on an 802.1Q port.
- The VLAN to which frames are forwarded if received untagged on an 802.1Q port.

Cisco switches produce errors if the native VLAN does not match at each end of the link. The default native VLAN in Cisco devices is VLAN 1.

You can control the 802.1Q VLAN traffic sent over a trunk, which is possible for security purposes or load balancing.

The command that creates and controls trunks on Cisco IOS-based switches is the interface command:

 Please see page 245 for more details.

```
switchport trunk {allowed vlan vlan-list} | {encapsulation {dot1q | isl | negotiate}} | {native vlan
vlan-id} | {pruning vlan vlan-list}
```

VLAN Trunking Protocol (VTP) is a Cisco proprietary Layer 2 multicast messaging protocol that synchronizes VLAN information across all media types and tagging methods on your switches. To enjoy the benefits of VTP, your switches must meet the following requirements:

- You must configure the VTP domain name identically on each device; domain names are case-sensitive.
- The switches must be adjacent.
- The switches must be connected with trunk links.
- The same VTP password must be configured if used in the domain.

Generally, you find four items in all VTP messages:

- VTP protocol version (either 1,2 or 3)
- VTP message type
- Management domain name length
- Management domain name

VTP has four possible message types:

- Summary advertisements
- Subset advertisements
- Advertisement requests
- VTP Join messages (used for pruning)

The VTP configuration revision number is important. This value determines whether a switch has stale information about VLANs and ultimately controls whether the switch overwrites its VLAN database with new information. The revision number increments each time a change is made to the VLAN database on a Server mode VTP system. The number is from 0 to 4,294,967,295. When introducing new Server mode switches, ensure that you do not

 Please see page 245 for more details.

inadvertently overwrite the VLAN database because of a higher configuration revision number on the new switch. Introducing new switches in Transparent mode helps ensure that this problem never results.

You have three possible modes for your VTP servers:

- **Server:** Enables you to create, modify, and delete VLANs; these changes are advertised to VTP Client mode systems; Catalyst switches default to this mode.
- **Client:** Does not enable the creation, modification, or deletion of VLANs on the local device; VLAN configurations are synchronized from Server mode systems.
- **Transparent:** Permits the addition, deletion, and modification of VLAN information, but the information resides only locally on the Transparent device; these systems forward advertisements from servers but do not process them.

Following is a sample configuration of VTP for a Server mode system in Cisco IOS mode. Note that changing the VTP domain on this system resets the configuration revision number to 0:

```
Switch# configure terminal
Switch(config)# vtp mode server
Setting device to VTP SERVER mode.
Switch(config)# vtp domain Lab_Network
Setting VTP domain name to Lab_Network
Switch(config)# end
Switch#
```

VTP Pruning

VTP pruning enables you to limit the amount of traffic sent on trunk ports. It limits the distribution of flooded frames to only switches that have members of the particular VLAN. You can enable VTP pruning with this command:

```
vtp pruning
```

When you enable pruning on the switch, all VLANs are pruned by default (with the exception of VLAN 1). You

 Please see page 245 for more details.

need to configure pruning on only one VTP server, and the setting automatically propagates. You can change this behavior by making select VLANs you choose prune-ineligible. This is done with the following command:

```
switchport trunk pruning vlan {none | {{add |  except | remove} vlan[,vlan[,vlan[,...]]}}
```

Following is the Cisco IOS command:

```
vtp pruning
```

EtherChannel

EtherChannel enables you to bundle redundant links and treat them as a single link, thus achieving substantial bandwidth and redundancy benefits. It is often advisable to use an EtherChannel for key trunks in your campus design. Notice that EtherChannel affects STP because ordinarily one or more of the links would be disabled to prevent a loop.

Be aware of the following guidelines for EtherChannel:

- All Ethernet interfaces on all modules must support EtherChannel.
- You have a maximum of eight interfaces per EtherChannel.
- The ports do not need to be contiguous or on the same module.
- All ports in the EtherChannel must be set for the same speed and duplex.
- Enable all interfaces in the EtherChannel.
- An EtherChannel will not form if one of the ports is a Switched Port Analyzer (SPAN) destination.
- For Layer 3 EtherChannels, assign a Layer 3 address to the port-channel logical interface, not the physical interfaces.
- Assign all EtherChannel ports to the same VLAN or ensure they are all set to the same trunk encapsulation and trunk mode.

 Please see page 245 for more details.

- The same allowed range of VLANs must be configured on all ports in an EtherChannel.
- Interfaces with different STP port path costs can form an EtherChannel.
- After an EtherChannel has been configured, a configuration made to the physical interfaces affects the physical interfaces only.

EtherChannel load balancing can use MAC addresses, IP addresses, or Layer 4 port numbers—either source, destination, or both source and destination addresses.

Here is an example:

```
Router# configure terminal
Router(config)# interface range fastethernet 2/2 -8
Router(config-if)# channel-group 2 mode desirable
Router(config-if)# end
```

Ethernet

Ethernet refers to the family of LAN products covered by the IEEE 802.3 standard. This standard defines the carrier sense multiple access collision detect (CSMA/CD) protocol. Four data rates are currently defined for operation over optical fiber and twisted-pair cables:

- **10 Mbps:** 10BASE-T Ethernet
- **100 Mbps:** Fast Ethernet
- **1000 Mbps:** Gigabit Ethernet
- **10,000 Mbps:** 10 Gigabit Ethernet

Ethernet has replaced just about every other LAN technology because of the following reasons:

- Is easy to understand, implement, manage, and maintain

 Please see page 245 for more details.

- Has a relatively low cost
- Provides extensive topological flexibility
- Is a standards-compliant technology

802.3

802.3 defines the original shared media LAN technology. This early Ethernet specification runs at 10 Mbps.

Ethernet can run over various media such as twisted pair and coaxial. You often see 802.3 Ethernet referred to as different terms because of the differences in the underlying media. Here are examples:

- **10BASE-T:** Ethernet over Twisted-Pair Media
- **10BASE-F:** Ethernet over Fiber Media
- **10BASE2:** Ethernet over Thin Coaxial Media
- **10BASE5:** Ethernet over Thick Coaxial Media

802.3u (Fast Ethernet)

Fast Ethernet refers to any one of a number of 100-Mbps Ethernet specifications. As its name implies, Fast Ethernet offers speeds ten times that of the 10BASE-T Ethernet specification.

Although Fast Ethernet is a faster technology, it still preserves such qualities as frame format, MAC mechanisms, and maximum transmission unit (MTU). These similarities permit you to use existing 10BASE-T applications and network management tools on Fast Ethernet networks.

802.3z (Gigabit Ethernet)

This Ethernet technology builds on the foundations of the old but increases speeds tenfold over Fast Ethernet to 1000 Mbps, or 1 gigabit per second (Gbps).

 Please see page 245 for more details.

802.3ab (Gigabit Ethernet over Copper)

Gigabit Ethernet over Copper (also known as 1000BASE-T) is another extension of the existing Fast Ethernet standard. 802.3ab specifies Gigabit Ethernet operation over the Category 5e/6 cabling systems already installed. This reuse of the existing infrastructure helps make 802.3ab a cost-effective solution.

10 Gigabit Ethernet

The latest in Ethernet technologies, 10 Gigabit Ethernet provides the following features:

- High bandwidth
- Low cost of ownership
- Scalability from 10 Mbps to 10,000 Mbps

Long Reach Ethernet

The Cisco Long Reach Ethernet (LRE) networking solution delivers 5-Mbps to 15-Mbps speeds over existing Category 1/2/3 wiring. As the name conveys, this Ethernet-like performance extends 3500 to 5000 feet.

Gigabit Interface Converter

The Gigabit Interface Converter (GBIC) is a Cisco standards-based hot-swappable input/output device that plugs into a Gigabit Ethernet slot on a Cisco network device. This flexibility enables you to inexpensively adapt your network equipment to any changes in the physical media that might be introduced.

You can intermix GBICs in a Cisco device to support any combination of 802.3z-compliant 1000BASE-SX, 1000BASE-LX/LH, or 1000BASE-ZX interfaces. Upgrading to the latest interface technologies is simple because of these GBICs.

 Please see page 245 for more details.

Chapter 3

IP Addressing

IPv4 Addresses

IPv4 addresses consist of 32 bits, which are divided into four sections of 8 bits, each called an octet. Addresses are typically represented in dotted-decimal notation. For example

10.200.34.201

Subnet masks identify which portion of the address identifies a particular network and which portion identifies a host on the network.

The address classes defined for IP networks consist of the following subnet masks:

Class A 255.0.0.0 (8 bits)
Class B 255.255.0.0 (16 bits)
Class C 255.255.255.0 (24 bits)

Class A addresses begin with 0 and have a first octet in decimal of 1 to 127. Class B addresses begin with 10 and range from 128 to 191. Class C addresses begin with 110 and range from 192 to 223.

Class D and Class E addresses also are defined. The Class D address space has the first 4 bits set to 1110 and has a first octet of 224 to 247. These addresses are used for IP multicast.

Class E addresses have the first 4 bits set to 1111 and have a first octet of 248 to 255. These addresses are reserved for experimental use.

Of the entire IPv4 address space, several blocks of IPs have been reserved for a specific use. The private IP space, which should not be used outside of an administrative domain, has been allocated the following blocks:

 Please see page 245 for more details.

10.0.0.0 to 10.255.255.255
172.16.0.0 to 172.31.255.255
192.168.0.0 to 192.168.255.255

Other allocated ranges include the multicast ranges (224.0.0.0 to 239.255.255.255), the loopback range (127.0.0.0 to 127.255.255.255), and the link local range (169.254.0.0 to 169.254.255.255).

Subnetting

Subnetting enables for the creation of smaller, more-efficient networks. Overall network traffic is reduced, and security measures can be easily introduced in a subnetted network.

The IP address is 32 bits in length. It has a network ID portion and a host ID portion. The number of bits used for the host ID dictates the number of hosts possible on the network or subnetwork. One address is reserved for the network ID (all host bits set to 0), and one address is reserved for a subnet broadcast (all host bits set to 1). To calculate the number of hosts available on a subnet, use the formula 2 ^ *n* – 2, where *n* is the number of bits used for the host ID.

To identify subnets, bits are "borrowed" from the host portion. The number of subnets that can be created depends on the number of bits borrowed. The number of subnets available is calculated with 2 ^ *n*, where *n* is the number of bits "borrowed."

Here is an example of subnetting. Take the address 10.172.16.211 with a subnet mask of 255.255.192.0. First note that this mask uses 18 bits. Fourteen bits remain for host addressing. That means that on a subnet here 2 ^ 14 –2 addresses are available. That is, 16,382 host addresses are possible. A default Class A network uses 8 bits for the mask. Here 10 bits are "borrowed" from the host portion. That enables for the creation of 2 ^ 10 = 1024 subnets.

VLSM

One of the fundamental concepts in networking is subnetting, that is, breaking one subnet into smaller pieces. With Variable Length Subnet Masking (VLSM), a subnet can be broken up into variable length pieces. To illustrate, the following diagram shows that a /24 network can be broken up into two /25 networks, four /26 networks, or eight /27 networks.

 Please see page 245 for more details.

192.168.20.0/24

Two/25 Networks	*Four/26 Networks*	*Eight/27 Networks*
192.168.20.0/25	192.168.20.0/26	192.168.20.0/27
192.168.20.128/25	192.168.20.64/26	192.168.20.32/27
	192.168.20.128/26	192.168.20.64/27
	192.168.20.192/26	192.168.20.96/27
		192.168.20.128/27
		192.168.20.160/27
		192.168.20.192/27
		192.168.20.224/27

Before VLSM, only one of these options could be chosen. With VLSM, the same /24 network can be subnetted into one /25, one /26, and two /27s, as shown in the following diagram. That is, the new, smaller subnets can be of variable length; they don't need to be a single length (/25, /26, or /27).

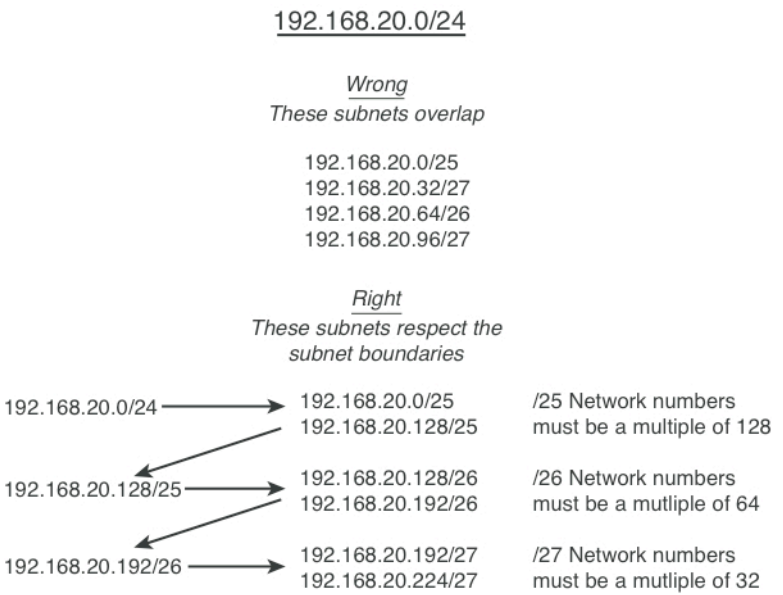
192.168.20.0/24

One/25 Network	*One/26 Network*	*Two/27 Networks*
192.168.20.0/25	192.168.20.128/26	192.168.20.192/27
		192.168.20.224/27

Before VLSM, to properly address a series of point-to-point networks, a /30 subnet would be required. Without variable length subnets, an entire network would need to be subnetted into /30 networks. If only a handful of /30s were required, many IPs would be wasted. VLSM enables a network administrator to choose subnetting boundaries based on the requirements of the network, rather than being forced to design around the constraints of IP addressing.

VLSM does not change other rules of IP addressing. Using the previous illustration, if a /24 network is subnetted into one /25, one /26, and two /27s, the organization must follow the standard "breaks" between subnets. In other words, the order of the subnets matter. The /24 cannot be broken into a /25, then one /27, and then a /26, followed by the second /27 as shown here:

 Please see page 245 for more details.



The subnetting must occur along natural breaks.

VLSM is often confused with classless networking and CIDR. They are related but refer to different IP addressing concepts. Classless networking refers to the delinking of Class A, B, C, and D networks from actual IP addresses. In a classless network, a subnet within the 10.x.x.x range doesn't need to be a /8. CIDR is a method in which subnets can be grouped together. It provides a way to refer a list of consecutive subnets without having to list each one individually. For example, the subnets of 192.168.0.0/24, 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24 can be aggregated together and referred to as 192.168.0.0/22. It is massively useful in large networks where large groups of IP address ranges can be aggregated together within a routing table or access lists.

Address Resolution Protocol

Address Resolution Protocol (ARP) can resolve IP addresses to MAC addresses in an Ethernet network. A host wanting to obtain a physical address broadcasts an ARP request onto the TCP/IP network. The host on the

 Please see page 245 for more details.

network with the IP address in the request then replies with its physical hardware address. When a MAC address is determined, the IP address association is stored in an ARP cache for rapid retrieval. Then the IP datagram is encapsulated in a link-layer frame and sent over the network. Encapsulation of IP datagrams and ARP requests and replies on IEEE 802 networks other than Ethernet is specified by the Subnetwork Access Protocol (SNAP).

Reverse Address Resolution Protocol (RARP) works the same way as ARP, except that the RARP request packet requests an IP address rather than a MAC address. Use of RARP requires an RARP server on the same network segment as the router interface. RARP often is used by diskless nodes that do not know their IP addresses when they boot. The Cisco IOS Software attempts to use RARP if it does not know the IP address of an interface at startup. Also, Cisco routers can act as RARP servers by responding to RARP requests that they can answer.

Enabling Proxy ARP

Cisco routers use proxy ARP to help hosts with no knowledge of routing determine the MAC addresses of hosts on other networks. If the router receives an ARP request for a host not on the same network as the ARP request sender, and if the router has all its routes to that host through other interfaces, it generates a proxy ARP reply packet, giving its own local MAC address. The host that sent the ARP request then sends its packets to the router, which forwards them to the intended host. Proxy ARP is enabled by default.

To enable proxy ARP if it has been disabled, use the following command:

```
Router(config-if)# ip proxy-arp
```

Defining Static ARP Cache Entries

To configure static mappings, use the following command:

```
Router(config)# arp ip-address hardware-address type
```

Use the following command to set the length of time an ARP cache entry stays in the cache:

```
Router(config-if)# arp timeout seconds
```

 Please see page 245 for more details.

Setting ARP Encapsulations

Cisco routers can actually use three forms of address resolution: ARP, proxy ARP, and Probe (similar to ARP). Probe is a protocol developed by Hewlett-Packard (HP) for use on IEEE 802.3 networks.

By default, standard Ethernet-style ARP encapsulation (represented by the **arpa** keyword) is enabled on the IP interface. You can change this encapsulation method to SNAP or HP Probe, as required by your network, to control the interface-specific handling of IP address resolution into 48-bit Ethernet hardware addresses.

To specify the ARP encapsulation type, use the following command:

```
Router(config-if)# arp {arpa | probe | snap}
```

Hot Standby Router Protocol

The Hot Standby Router Protocol (HSRP) provides high network availability by routing IP traffic from hosts without relying on the availability of any single router. HSRP is used in a group of routers to select an active router and a standby router. The active router is the router of choice for routing packets; a standby router is a router that takes over the routing duties when an active router fails or when other preset conditions are met.

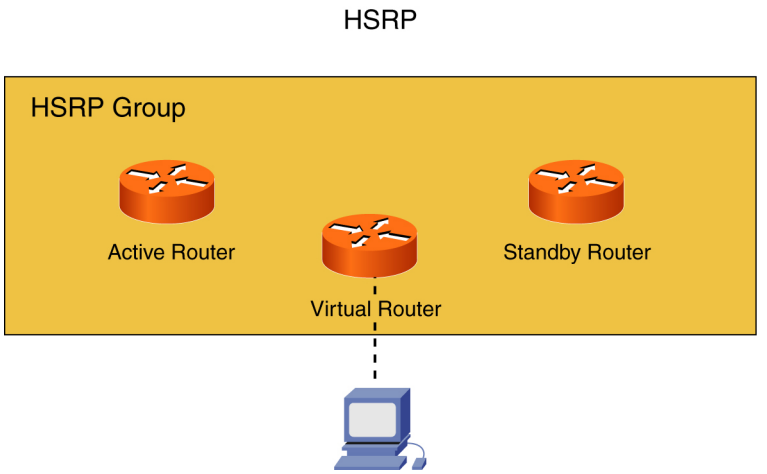
HSRP is useful for hosts that do not support a router discovery protocol (such as Internet Control Message Protocol [ICMP] Router Discovery Protocol [IRDP]) that cannot switch to a new router when their selected router reloads or loses power.

When the HSRP is configured on a network segment, it provides a virtual MAC address and an IP address shared among a group of routers running HSRP. The address of this HSRP group is the virtual IP address. One of these devices is selected by the protocol to be the active router.

HSRP detects when the designated active router fails, at which point a selected standby router assumes control of the MAC and IP addresses of the Hot Standby group. A new standby router is also selected at that time. Devices that run HSRP send and receive multicast User Datagram Protocol (UDP)-based hello packets to detect router failure and to designate active and standby routers. For an example of an HSRP topology, see Figure 3-1.

 Please see page 245 for more details.

FIGURE 3-1
HSRP topology



Devices that run HSRP send and receive multicast UDP-based hello packets to detect router failure and to designate active and standby routers.

You can configure multiple Hot Standby groups on an interface, thereby making fuller use of redundant routers and load sharing. To do so, specify a group number for each Hot Standby command you configure for the interface.

To enable the HSRP on an interface, use the following command:

```
Router(config-if)# standby [group-number] ip [ip-address [secondary]]
```

Whereas the preceding represents the only required HSRP configuration commands, you should be familiar with many others for configuring additional HSRP behaviors.

To configure the time between hello packets and the hold time before other routers declare the active router to be down, use the following command:

```
Router(config-if)# standby [group-number] timers [msec]
hellotime [msec] holdtime
```

 Please see page 245 for more details.

You can also set the Hot Standby priority used in choosing the active router. The priority value range is from 1 to 255, in which 1 denotes the lowest priority, and 255 denotes the highest priority:

```
Router(config-if)# standby [group-number] priority priority
```

You can also configure a router with higher priority to preempt the active router. In addition, you can configure a preemption delay after which the Hot Standby router preempts and becomes the active router:

```
Router(config-if)# standby [group-number] preempt [delay {minimum delay | reload delay | sync delay}]
```

You can also configure the interface to track other interfaces so that if one of the other interfaces goes down, the device's Hot Standby priority is lowered:

```
Router(config-if)# standby [group-number] track type number [interface-priority]
```

You can also specify a virtual MAC address for the virtual router:

```
Router(config-if)# standby [group-number] mac-address
macaddress
```

Finally, you can configure HSRP to use the burned-in address of an interface as its virtual MAC address rather than the preassigned MAC address (on Ethernet and FDDI) or the functional address (on Token Ring):

```
Router(config-if)# standby use-bia [scope interface]
```

Gateway Load Balancing Protocol

Gateway Load Balancing Protocol (GLBP) takes HSRP even further. Instead of just providing backup for a failed router, it can also handle the load balancing between multiple routers. GLBP provides this functionality using a single virtual IP address and multiple virtual MAC addresses. Workstations are configured with the same virtual IP address, and all routers in the virtual router group participate in forwarding packets. GLBP members communicate with each other using hello messages sent every 3 seconds to the multicast address 224.0.0.102.

 Please see page 245 for more details.

Members of a GLBP group elect one gateway to be the active virtual gateway (AVG) for that group. It is the job of other group members to back up for the AVG if that the AVG fails. The AVG assigns a virtual MAC address to each member of the GLBP group. The AVG is responsible for answering ARP requests for the virtual IP address. Load sharing is achieved by the AVG replying to the ARP requests with different virtual MAC addresses that the group members will respond to.

Although you can use many optional commands with GLBP, the primary command to enable GLBP follows:

```
glbp group ip [ip-address [secondary]]
```

Note how similar this command is to the HSRP configuration command.

Virtual Router Redundancy Protocol

Virtual Router Redundancy Protocol (VRRP) is so similar to HSRP that it can be basically thought of as the standards-based version of the protocol. Like HSRP, it lacks the inherent load-balancing capabilities that GLBP provides.

Although many customization commands exist, the command to enable the protocol is just like that of the other redundancy protocols in structure:

```
vrrp group ip ip-address [secondary]
```

Network Address Translation

Network Address Translation (NAT) enables an organization to use private IP address space inside the organization (or any other IP address it might require) and present this IP address differently to the outside networks. Organizations might use NAT for the following purposes:

- To connect private IP internetworks that use nonregistered IP addresses to the Internet, NAT translates the internal local addresses to globally unique IP addresses before sending packets to the outside network.
- Internal addresses must be changed, and this creates a large administrative burden. NAT is used instead to translate addresses.
- To do basic load sharing of TCP traffic. A single global IP address is mapped to many local IP addresses by using the TCP load distribution feature.

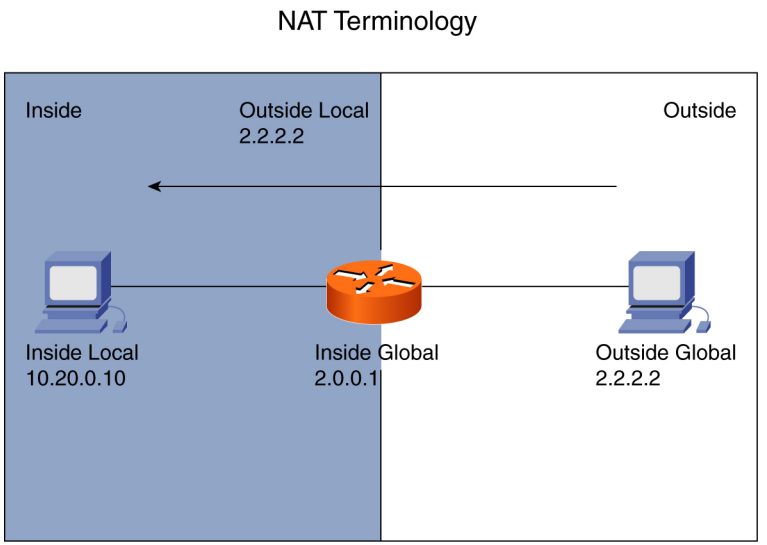
 Please see page 245 for more details.

NAT uses the following definitions:

- **Inside local address:** The IP address assigned to a host on the inside network. Often, this is a nonregistered IP address.
- **Inside global address:** A legitimate IP address that represents one or more inside local IP addresses to the outside world.
- **Outside local address:** The IP address of an outside host as it appears to the inside network.
- **Outside global address:** The IP address assigned to a host on the outside network by the owner of the host.

For a depiction of this NAT terminology, see Figure 3-2.

FIGURE 3-2
NAT terminology



 Please see page 245 for more details.

Translating Inside Source Addresses

You can configure static or dynamic inside source translation:

- Static translation establishes a one-to-one mapping between your inside local address and an inside global address. Static translation is useful when a host on the inside must be accessible by a fixed address from the outside.
- Dynamic translation establishes a mapping between an inside local address and a pool of global addresses.

Configuring Static Translations

To establish a static translation between an inside local address and an inside global address, use the following global configuration command:

```
Router(config)# ip nat inside source static local-ip global-ip
```

To mark the appropriate interface as connected to the inside, use the following interface configuration command:

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

To mark the appropriate interface as connected to the outside, use the following interface configuration command:

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

Configuring Dynamic Translations

To define a pool of global addresses to be allocated as needed, use the following global configuration command:

```
Router(config)# ip nat pool name start-ip end-ip {netmask netmask | prefix-length prefix-length}
```

To define a standard access list permitting those addresses to be translated, use the following global configuration command:

```
Router(config)# access-list access-list-number permit source [source-wildcard]
```

Next, a establish dynamic source translation, specifying the access list defined in the prior step, using the following global configuration command:

```
Router(config)# ip nat inside source list access-list-number pool name
```

 Please see page 245 for more details.

To mark the appropriate interface as connected to the inside, use the following interface configuration command:

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

To mark the appropriate interface as connected to the outside, use the following interface configuration command:

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

Overloading an Inside Global Address

You can conserve addresses in the inside global address pool by allowing the router to use one global address for many local addresses. When multiple local addresses map to one global address, the TCP or UDP port numbers of each inside host distinguish between the local addresses.

To permit this behavior, use the dynamic translations configuration from the previous section and include the **overload** keyword as follows:

```
Router(config)# ip nat inside source list access-list-number pool name overload
```

Translating Overlapping Addresses

You can use NAT to translate inside addresses that overlap with outside addresses. Use this feature if your IP addresses in the stub network are legitimate IP addresses belonging to another network and you want to communicate with those hosts or routers.

You can configure the translations using static or dynamic means. To do so, use the same commands from the "Translating Inside Source Addresses" section, but use the **ip nat outside** source syntax.

TCP Load Distribution

If your organization has multiple hosts that must communicate with a heavily used host, you can establish a virtual host on the inside network that coordinates load sharing among real hosts. Destination addresses that match an access list are replaced with addresses from a rotary pool. Allocation is done on a round-robin basis and only when a new connection is opened from the outside to the inside.

First, define a pool of addresses containing the addresses of the real hosts in global configuration mode:

 Please see page 245 for more details.

```
Router(config)# ip nat pool name start-ip end-ip {netmask netmask |  prefix-length prefix-length} type
rotary
```

Next, define an access list permitting the address of the virtual host in global configuration mode:

```
Router(config)# access-list access-list-number permit source [source-wildcard]
```

Next, establish dynamic inside destination translation, specifying the access list defined in the prior step:

```
Router(config)# ip nat inside destination list access-list-number pool name
```

To mark the appropriate interface as connected to the inside, use the following interface configuration command:

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

To mark the appropriate interface as connected to the outside, use the following interface configuration command:

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

Monitoring and Maintaining NAT

To clear all dynamic address translation entries from the NAT translation table, use the following command:

```
Router# clear ip nat translation *
```

To clear a simple dynamic translation entry containing an inside translation, or both inside and outside translation, use the following command:

```
Router# clear ip nat translation inside global-ip local-ip [outside local-ip global-ip]
```

To clear a simple dynamic translation entry containing an outside translation, use the following command:

```
Router# clear ip nat translation outside local-ip global-ip
```

To clear an extended dynamic translation entry, use the following command:

```
Router# clear ip nat translation protocol inside global-ip global-port local-ip local-port [outside
local-ip local-port global-ip global-port]
```

To display active translations, use the following command:

 Please see page 245 for more details.

```
Router# show ip nat translations [verbose]
```

To display translation statistics, use the following command:

```
Router# show ip nat statistics
```

Internet Control Message Protocol

Internet Control Message Protocol (ICMP) assists the operation of the IP network by delivering messages about the network's functionality—or lack thereof. ICMP includes functions for the following:

- **Communicating network errors:** Such as host or network unreachable.
- **Announcing network congestion:** An example is the ICMP Source Quench messages used to cause a sender to slow down transmission because of a router buffering too many packets.
- **Provide troubleshooting tools:** The Echo function is used by the ping utility to test connectivity between two systems.
- **Communicate timeouts in the network:** If a packet's TTL reaches 0, an ICMP message can be sent announcing this fact.

ICMP Protocol Unreachable Messages

If the Cisco device receives a nonbroadcast packet destined for itself that uses an unknown protocol, it sends an ICMP protocol unreachable message back to the source. Similarly, if the device receives a packet that it cannot deliver to the ultimate destination because it knows of no route to the destination address, it sends an ICMP host unreachable message to the source. This feature is enabled by default. To enable it if it's disabled, use the following command:

```
Router(config-if)# ip unreachables
```

ICMP Redirects

If the router resends a packet through the same interface on which it was received, the Cisco IOS Software sends an ICMP redirect message to the originator of the packet, telling the originator that the router is on a subnet directly connected to the receiving device and that it must forward the packet to another system on the same subnet.

 Please see page 245 for more details.

To enable the sending of ICMP redirect messages if this feature was disabled, use the following command:

```
Router(config-if)# ip redirects
```

Services

Network Time Protocol

There are many reasons that an administrator will want to keep the time accurate on all systems in the infrastructure. Network Time Protocol (NTP) assists the administrator in this goal by automatically synchronizing the time between network devices.

Devices in the network running NTP can receive the correct time from an authoritative time source, such as a Cisco router, a radio clock, or an atomic clock attached to a timeserver.

To configure a router to receive the time from an authoritative time source on the network, use the following command:

```
ntp server {{[vrf vrf-name] ip-address |  hostname} [version number] [key key-id] [source interface]
[prefer]}
```

Some platforms have a battery-powered hardware clock, referred to as the calendar, in addition to the software-based system clock. The hardware clock runs continuously, even if the router is powered off or rebooted. It is a good practice to periodically update the hardware clock with the time learned from NTP. To do this, use this command:

```
ntp update-calendar
```

To have the router provide the correct time for the network, you can use this command:

```
ntp master [stratum]
```

The *stratum* value is an indicator of how close a device is to the master time source. Consider it like a hop count. If you set the stratum to 1 on the router, you indicate that it is the authoritative time source.

You can also have the router synchronize the clock of a peer router, or be synchronized from that peer. The command to configure this is as follows:

 Please see page 245 for more details.

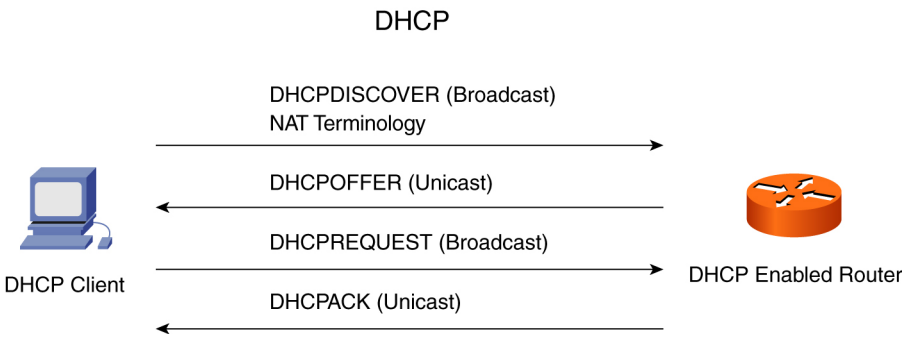
```
ntp peer {{[vrf vrf-name] ip-address |   hostname}[normal-sync][version number] [key key-id] [source
interface] [prefer]}
```

You should also note that NTP messages can be authenticated to ensure that accurate time is being sent to all devices.

DHCP

Cisco devices can function as DHCP servers and can be configured to forward requests to secondary servers if the Cisco device cannot satisfy the request. Figure 3-3 shows the four-step process that the router participates in to provide DHCP services.

FIGURE 3-3
DHCP



Configuring a Cisco Device as a DHCP Server

To configure the DHCP address pool name and enter DHCP pool configuration mode, use the following command:

```
Router(config)# ip dhcp pool name
```

The DHCP server assumes that all IP addresses in a DHCP address pool subnet are available for assigning to DHCP clients. You must specify the IP address that the DHCP server should not assign to clients. To do so, use the following command:

```
Router(config)# ip dhcp excluded-address low-address [high-address]
```

To configure a subnet and mask for the DHCP address pool, use the following command in DHCP pool configuration mode:

```
Router(config-dhcp)# network network-number [mask | /prefix-length]
```

 Please see page 245 for more details.

Additional DHCP pool configuration mode commands enable you to configure additional parameters for the scope, including default gateway, domain name, DNS server addresses, Windows Internet Naming Service (WINS) server addresses, and so on.

Web Cache Communication Protocol

Web Cache Communication Protocol (WCCP) enables an administrator to forward web traffic to a Cisco cache engine. The Cisco cache engine reduces transmission costs and downloading time for clients. When users request web pages, the WCCP-capable router sends the requests to a cache engine. If the cache engine has a copy of the requested page in storage, the cache engine sends the user that page. If there is no cached copy, the cache engine retrieves the requested page from the web server, stores a copy, and forwards the page to the user. The routers and the cache engine operate transparently from the perspective of end users. End users do not know that the page came from the cache engine rather than the web server.

The global configuration command used on the router to enable the protocol follows:

```
ip wccp {web-cache | service-number}
[group-address groupaddress] [redirect-list access-list]
[group-list access-list] [password [0-7] password]
```

To actually redirect traffic on an interface to a cache engine, use the following interface configuration command:

```
ip wccp {web-cache | service-number}
redirect out
```

Domain Name System

Cisco routers can participate in the Domain Name System (DNS). For example, you can specify a default domain name that the Cisco IOS Software uses to complete domain name requests. You can specify either a single domain name or a list of domain names. Any IP hostname that does not contain a domain name has the domain name you specify appended to it before being added to the host table. To specify this domain name, use the following command:

```
Router(config)# ip domain name name
```

 Please see page 245 for more details.

To define a list of default domain names to complete unqualified host names, use the following command:

```
Router(config)# ip domain list name
```

You can also specify DNS name servers for the router or switch to call on for name resolution. To do so, use the following command:

```
Router(config)# ip name-server server-address1 [server-address2...server-address6]
```

If you do not want to enable your router to use DNS for name resolution, you can use the following command to disable this default behavior:

```
Router(config)# no ip domain-lookup
```

Network Management

Logging and Syslog

Cisco devices communicate with an administrator through system messages. These system messages are typically sent to a logging process, so they are most often called syslog messages. Syslog is also the name of the UNIX-based service that handles system messages from UNIX systems (and also Cisco devices if configured to do so).

Logging is enabled by default. The **no logging on** command actually forces system messages to the console. This can impede the performance of the Cisco device because processes must wait for messages to be written to the console before the processes can continue their operations. It is recommended that the administrator leave the logging process enabled (the default behavior); that way logging messages can be written to the console more efficiently.

Because there is no way to stop the sending of system messages to the console, administrators should use the **logging synchronous** command inline configuration mode. This command prevents these messages from "interrupting" typing at the console.

To have the Cisco device store syslog messages in an internal buffer, administrators should ensure the logging process is in its default-enabled state (**logging console** command) and then use the command **logging buffered**\,

 Please see page 245 for more details.

which uses a default size of 4096 bytes. This can be changed by specifying an optional size at the end of the **logging buffered** command. To view the contents of the buffer, use the **show logging** command. The oldest messages display first. When the buffer fills to capacity, new messages overwrite the oldest messages. You can clear the buffer anytime with the **clear logging** command.

You can store syslog messages on a server (UNIX- or Windows-based) in the network. CiscoWorks LAN Management Suite (LMS) features a built-in syslog server application that stores these messages in a searchable database. It enables the filtering of messages, reporting on messages, and even action filters that enable automated responses to certain messages, including pages and emails.

To send system messages to a UNIX or CiscoWorks syslog server, ensure the logging process is enabled and then issue the command **logging x.x.x.x**, in which **x.x.x.x** is the IP address of the syslog server. The command can be entered multiple times to configure multiple destinations for the messages. To limit the sending of all messages, use the **logging trap level** command, in which **level** is the number or the name of the severity level. For example, **logging trap notifications** restricts the messages sent to only those of level 0 through 5. This keeps debugging and informational messages from being sent to the server. UDP port 514 is used for syslog messages, so be sure that your firewalls permit this port if you need the messages to pass through such devices.

UNIX syslog servers use a facility code to identify the source of syslog messages. They use this code to create different logs for the different sources of messages. Sample facilities include **lpr** for the Line Printer System and **mail** for the email system. UNIX syslog servers reserve the facility codes **local0** through **local7** for log messages received from remote servers and network devices. To have switches use one log file on the server and routers use another, change the facility code for switches using the **logging facility local6** command. By default, Cisco devices use **local7** for their messages so that your router messages will be in a different log. CiscoWorks requires the use of **local7**.

Some devices enable logging of system messages to a file in flash memory. The command to do this is simply **logging file flash:myname.txt**. This command can also set size limits on the file and control the types of messages sent to flash.

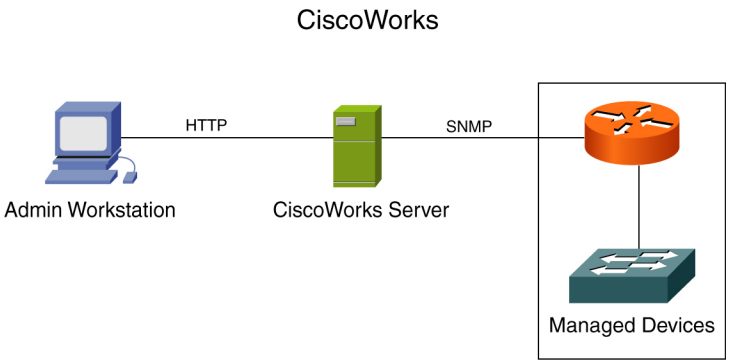
 Please see page 245 for more details.

Administrators should stamp syslog messages with the date and time that they were generated. This is accomplished with the **service timestamps log datetime** command.

Simple Network Management Protocol

Simple Network Management Protocol (SNMP) is a part of the TCP/IP suite of protocols and has powerful monitoring capabilities. CiscoWorks relies on SNMP and various other protocols to configure and monitor Cisco equipment. For an example, see Figure 3-4.

FIGURE 3-4
CiscoWorks



SNMP Version 2c

At a minimum, to configure a Cisco device for SNMP, you need to assign passwords—known as community strings in SNMP. Here are typical Cisco IOS global configuration commands for setting strings that permit configuration and monitoring, respectively:

```
snmp-server community [string] rw
snmp-server community [string] ro
```

Typically, you view information obtained by SNMP using a graphical user interface, like that provided by CiscoWorks.

 Please see page 245 for more details.

Following are some examples of **show** commands for monitoring SNMP activities on the equipment:

- **no snmp-server:** Disables SNMP agent operation
- **show snmp engineid:** Displays the identification of the local SNMP engine and all remote engines configured on the router
- **show management event:** Displays the SNMP event values configured on your routing device through the use of the event Management Information Base (MIB)
- **show snmp:** Checks the status of SNMP communications
- **show snmp group:** Displays the names of groups on the router and the security model, the status of the different views, and the storage type of each group
- **show snmp pending:** Displays the current set of pending SNMP requests
- **show snmp sessions:** Displays the current SNMP sessions
- **show snmp user:** Displays information on each SNMP username in the group username table

SNMP Version 3

SNMP Version 3 dramatically improves upon the security model for the management protocol. Whereas previous versions used clear-text passwords, SNMP Version 3 provides for authentication and encryption of network management information.

With SNMP Version 3, you create a view that defines what MIB variables a particular user or group of users can access. Here is the syntax to create a view. All the commands that follow are global configuration mode commands:

```
snmp-server view view-name oid-tree {included | excluded}
```

Notice how you provide the view with a name, and then you specify the portion of the MIB tree that the user can access. The example here adds the Internet portion of the tree and everything below it to the view name SAMPLEVIEW. This is basically the entire MIB structure:

 Please see page 245 for more details.

```
snmp-server view SAMPLEVIEW internet included
```

If you want a user or group of users to be able to access this view of the MIB that you defined, use the following syntax:

```
snmp-server group [groupname {v1 | v2c | v3 [auth | noauth | priv]}][read readview] [write writeview]
[notify notifyview] [access access-list]
```

Here is an example of the creation of a group to use the view:

```
snmp-server group MYSAMPLEGROUP v3 auth read SAMPLEVIEW
```

Adding a user account to this group is a simple matter. Use the syntax shown here:

```
snmp-server user username groupname [remote ip-address [udp-port port]] {v1 | v2c | v3
[encrypted] [auth {md5 | sha} auth-password ]} [access access-list]
```

Here is sample syntax using the group we just created:

```
snmp-server user jsmith MYSAMPLEGROUP v3 auth md5 secret
```

Switched Port Analyzer and Remote SPAN

Network analyses in a switched Cisco environment is handled using Switched Port Analyzer (SPAN). Traffic is mirrored from source ports to a destination port on the switch; a network analyzer should be located at the destination switch.

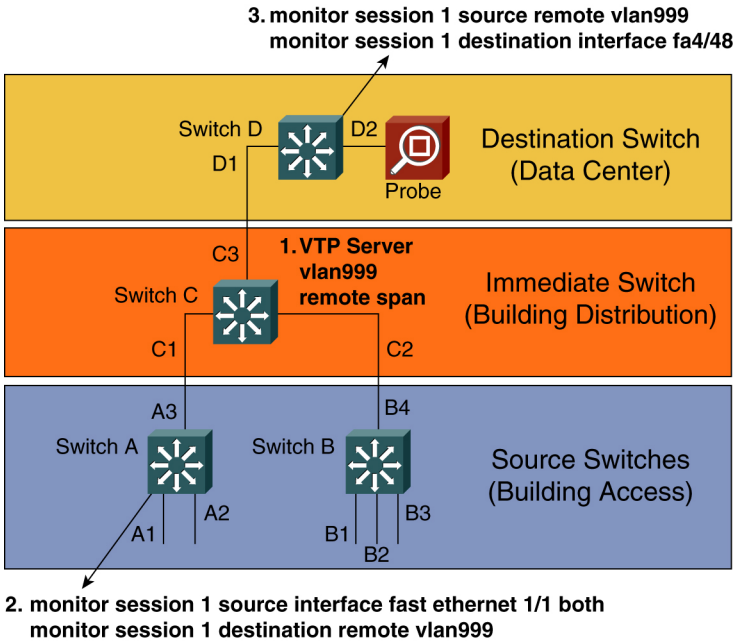
SPAN is available in several forms:

- **Local SPAN:** SPAN source ports and the destination port are located on the same device.
- **VLAN-based SPAN (VSPAN):** The source is a VLAN as opposed to one or more ports.
- **Remote SPAN (RSPAN):** The SPAN source and destination ports are located on different switches; a special-purpose VLAN carries the mirrored frames to the destination port in the network.

Figure 3-5 shows a sample RSPAN configuration.

 Please see page 245 for more details.

FIGURE 3-5
RSPAN



You should be aware of important guidelines for SPAN:

- You can configure destination ports as trunks to capture tagged traffic.
- A port specified as a destination port in one SPAN session cannot be a destination port for another SPAN session.
- A port channel interface (an EtherChannel) cannot be a destination.
- If you specify multiple ingress source ports, the ports can belong to different VLANs.
- Destination ports never participate in any spanning-tree instance.

A method of transporting source SPAN data to a remote destination. This is used when the SPAN source and the SPAN destination are located on two different switches. The data must be transported over a special purpose VLAN, which is configured on all switches in the transit path. The VLAN must be configured as a **remote-span** in VLAN configuration mode:

 Please see page 245 for more details.

```
Router(config-vlan)# remote-span
```

On the switch in which the data is originated, the SPAN destination is the RSPAN VLAN. On the switch in which the data is ultimately destined, the SPAN source is specified as the RSPAN VLAN, and the destination is a physical port. All transit switches simply trunk the RSPAN between switches.

Implementing IPv4 Tunneling and GRE

Typical network traffic is consistent with the TCP/IP model, meaning it has a number of distinct headers. For example, a packet can have a layer 2 header, such as Ethernet; a layer 3 header, such as IP; and a layer 4 header, such as TCP. When tunneling is implemented, a header with data is encapsulated within a header at the same layer. In IPv4 tunneling, an IP packet typically contains another IP packet, as shown in the following diagram:

```
Serial0/0 is up, line protocol is UP
  Hardware is GT96K Serial
  Internet address is 192.168.20.1/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 2000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  CRC checking enabled
<OUTPUT OMITTED>
```

Tunneling is used for a number of reasons including connecting two disjointed networks that might not have IP communication between them. A number of IPv4 tunneling protocols existing including IPsec and Generic Router Encapsulation (GRE).

GRE carries an arbitrary payload (like IPv4, IPv6, or IPsec) using IP packets of protocol 47. It does not, however, encrypt any tunneled data. GRE tunnels can be used with OSPF to extend the backbone to a disconnected area. Care must be taken to ensure that the route to the destination address provided during the GRE configuration is not learned via OSPF. This can lead to a recursive route, causing the GRE tunnel to bounce.

To create a GRE tunnel, it is necessary to create the numbered tunnel interface:

 Please see page 245 for more details.

```
Router(config)# int tunnel <#>
Configure an IP address on the interface:
Router(config-int)# ip address <IP> <MASK>
Specify the source interface or IP address:
Router(config-int)# tunnel source <int or IP>
Specify the destination address:
Router(config-int)# tunnel destination <IP>
```

 Please see page 245 for more details.

Chapter 4

IP Routing

Open Shortest Path First

Open Shortest Path First (OSPF) link-state routing protocol is designed to be more scalable and efficient than Routing Information Protocol (RIP). Consider the following OSPF features:

- Runs on IP and uses protocol 89.
- **Classless with variable:** Length subnet mask (VLSM) support.
- **Uses multicasts (224.0.0.5–all shortest path first [SPF] routers; 224.0.0.6:** Designated Router [DR]/ Backup Designated Router [BDR]) for hellos and updates.
- Plain text and Message Digest Algorithm 5 (MD5) authentication available. Null authentication is the default.
- Dijkstra's algorithm is used to produce a shortest-path tree for each destination. Link-state advertisements are used to build a database of the topology.

OSPF Packet Types

- **Type 1, Hello:** Builds adjacencies
- **Type 2, Database Description (DBD):** Checks for database synchronization between routers
- **Type 3, Link-State Request (LSR):** Requests link-state specifics from the router

 Please see page 245 for more details.

- **Type 4, Link-State Update (LSU):** Sends requested link-state records
- **Type 5, Link-State Acknowledgment (LSA):** Acknowledges the other packet types

OSPF Adjacencies

- Occurs through the exchange of hello packets.
- After adjacency is established, link-state databases (LSDB) are synched.
- Two OSPF neighbors on a point-to-point link form full adjacency with each other.
- In LANs, all routers form an adjacency with the DR and BDR; updates need to be sent only to the DR, which updates all other routers; and all other routers on the LAN are called DROTHERS and maintain a partial neighbor relationship with each other.

After adjacencies have been established, LSAs are exchanged through a reliable mechanism. LSAs are flooded to ensure topological awareness. LSAs have a sequence number and a lifetime value. LSAs convey the cost of links used for the SPF calculation. The cost metric is based on interface bandwidth. The LSA aging timer is a 30-minute default.

Hello packets are sent periodically and contain the following fields:

- **Router ID:** Identifies the router; highest IP chosen; loopback overrides all interfaces, however; can also be set with the **router-id** command; this ID is used to break ties for DR election.
- **Hello/Dead intervals:** Frequency at which hellos are sent and the amount of time that can elapse before router is declared dead; default is 10 seconds, and the default dead interval is 4 times that for an Ethernet-type network; these defaults vary based on network type.
- **Neighbors:** List of the adjacent routers.
- **Area ID:** Area identifier (always 0 for backbone).

 Please see page 245 for more details.

- **Router priority:** Priority value used for DR and BDR election.
- **DR/BDR addresses:** IP addresses of the DR and BDR if known.
- **Authentication password:** This password must match on routers configured for authentication.
- **Stub area flag:** All routers in the area must agree on this setting to form a stub area.

Here are the details of the exchange process between two routers on a LAN (Router 1 and Router 2) and the OSPF adjacency states involved:

1. Router 1 begins in the down state because it is not exchanging OSPF information with any other router. It sends hello packets via multicast address 224.0.0.5 (all SPF).
2. Router 2 receives the OSPF hello and adds Router 1 in its list of neighbors. This is the beginning of the Init State.
3. Router 2 sends a unicast hello packet response to Router 1.
4. Router 1 receives the hello and notes that it is listed in the packet. It adds Router 2 to its list of neighbors. Router 1 knows that it has bidirectional communication with Router 2. This is known as the two-way state.
5. In a LAN environment, the DR and BDR are elected.
6. In a LAN environment, the hello packets function as a keepalive mechanism every 10 seconds.

After the DR and BDR are established, the routers are in Exstart State, and they are ready to exchange database information. The exchange protocol functions as follows:

1. In the Exstart State, the DR and BDR establish an adjacency with each router in the network; a master-slave relationship is formed with the router ID indicating the master in the relationship.
2. The master and slave routers exchange DBD packets; this is the Exchange State. The LSAs in the DBD include sequence numbers used to indicate freshness.
3. When a DBD is received, the router acknowledges the receipt and compares the information with its current database. If more recent information is described in the DBD, the router sends an LSR to request the information, which is the Loading State. The router receiving the LSR responds with an LSU; this LSU is also acknowledged by the receiver.

 Please see page 245 for more details.

4. The router adds the new information to its LSDB.
5. When the exchange completes, the routers are in Full State.

Router information is later maintained using the following process:

1. The router notices the change and multicasts an LSU to the OSPF DR and BDR multicast address of 224.0.0.6.
2. The DR acknowledges the LSU and floods to all using multicast 224.0.0.5. This process involves acknowledgments, too.
3. The DR also sends the LSU to any other networks to which it is attached.
4. Routers update their LSDB with the new information in the LSU.

Summaries are sent every 30 minutes to ensure synchronization, and link state entries have a Max Age of 60 minutes.

Point-to-Point Links

Typically, a point-to-point link is a serial link, but it might also be a subinterface in a Frame Relay or ATM network. No DR or BDR election exists in the point-to-point environment. Packets are multicast to 224.0.0.5.

Nonbroadcast Multiaccess Modes of Operation

RFC: compliant modes:

- Nonbroadcast multiaccess (NBMA).
- One IP subnet required.
- Must manually configure neighbors—**neighbor address [priority number] [poll-interval number].**
- DR/BDR election.
- DR/BDR need full connectivity with all routers.

 Please see page 245 for more details.

- Sometimes used in partial mesh.
- Frame Relay and ATM networks default to this type.
- Point-to-multipoint.
- One IP subnet required.
- Hello packets used to discover neighbors.
- DR/BDR not required.
- Sometimes used in partial mesh.

Modes from Cisco:

- Point-to-multipoint nonbroadcast.
- Used if interface does not support multicast capabilities.
- Neighbors must be manually configured.
- DR/BDR election is not required.
- Broadcast
- Makes WAN appear as LAN.
- One IP subnet required.
- Hellos discover neighbors.
- DR/BDR elected.
- Requires full mesh.
- Point-to-point.
- One IP subnet required.

 Please see page 245 for more details.

- No DR/BDR election.
- Interfaces can be LAN or WAN.

You can use the following command to define the OSPF network type:

```
Router(config-if)# ip ospf network [{broadcast | nonbroadcast | point-to-multipoint | point-to-multipoint
nonbroadcast}]
```

Here is an example of statically defining adjacencies in a nonbroadcast multiaccess environment:

```
RouterA(config)# router ospf 1
RouterA(config-router)# network 172.16.0.0 0.0.255.255 area 0
RouterA(config-router)# neighbor 172.16.0.5 priority 0
RouterA(config-router)# neighbor 172.16.0.10 priority 0
```

Priorities are set to 0 for the neighboring routers to ensure that RouterA becomes the DR. This is the only router with full connectivity. Note that you can also set a router's priority locally using the **ip ospf priority** interface configuration command.

Troubleshooting Neighbor Relationships

OSPF neighbor list is empty:

- OSPF is not enabled properly on appropriate interfaces.
- Layer 1 or 2 is not functional.
- Passive interface is configured.
- Access list(s) blocking OSPF packets in multiple directions.
- Error in IP address or subnet mask configuration.
- Hello or dead interval mismatch.
- Authentication configuration error.
- Area ID mismatch.

 Please see page 245 for more details.

- Stub flag mismatch.
- OSPF adjacency exists with secondary IP addressing or asynchronous interface.
- Incorrect configuration type for NBMA environment.

OSPF neighbor stuck in Attempt State:

- Misconfigured **neighbor** statement
- Unicast nonfunctional in NBMA environment

OSPF neighbor stuck in Init State:

- Access list or Layer 2 problem blocking hellos in one direction
- Multicast nonfunctional on one side
- Authentication configured on only one side
- Broadcast keyword missing from the **map** command

OSPF neighbor stuck in Two-Way State:

- Priority 0 configured on all routers
- OSPF neighbor stuck in Exstart/Exchange
- Mismatched interface maximum transmission unit (MTU)
- Duplicate router IDs on routers
- Broken unicast connectivity
- Network type of point-to-point between Primary Rate Interface (PRI) and Basic Rate Interface (BRI)/dialer

 Please see page 245 for more details.

OSPF neighbor stuck in Loading State:

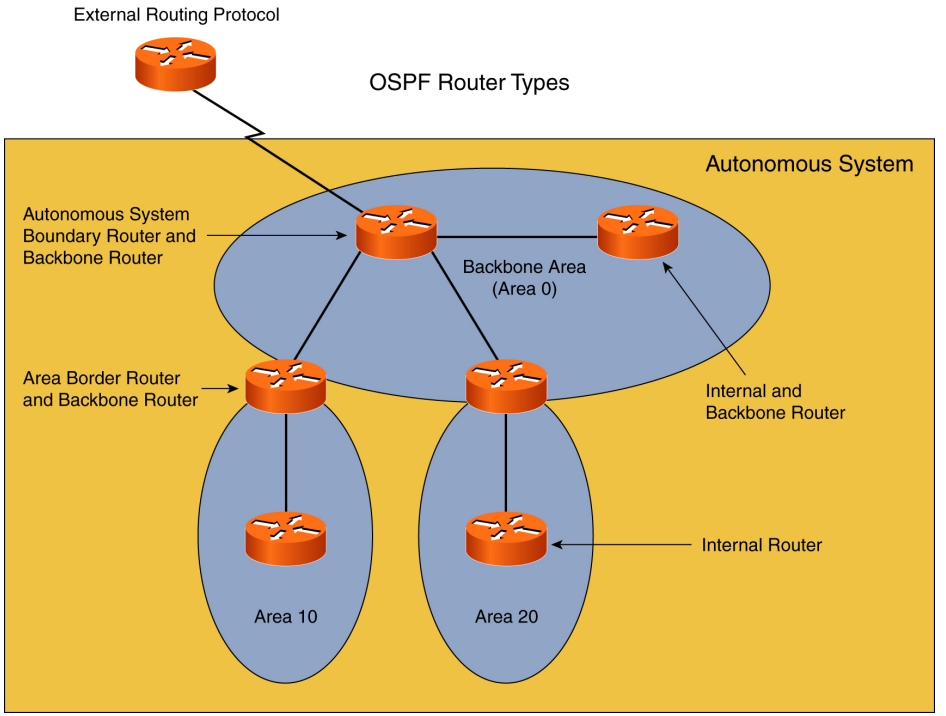
- Mismatched MTU
- Corrupted link-state request packet

Router Types

- **Internal routers:** All interfaces belong within the same area; these routers have a single link-state database.
- **Area Border Routers (ABR):** Connect one or more areas to the backbone; act as gateway for interarea traffic; separate link-state database for each connected area.
- **Backbone routers:** At least one interface in the backbone area.
- **Autonomous System Boundary Router (ASBR):** Inject routes into the OSPF network learned from another protocol; this router might be located anywhere. (It might also be backbone, internal, or ABR.)

 Please see page 245 for more details.

FIGURE 4-1
OSPF Router Types



LSA Types

OSPF uses various types of LSAs in its operation. You should be familiar with the types in the following table for the CCIE written exam.

Type	Description
1	Router
2	Network
3	Network Summary
4	ASBR Summary
5	AS External
7	NSSA External

 Please see page 245 for more details.

- **Router LSA (Type 1):** Lists all of a router's links and their state. These LSAs are flooded within the area they originated.
- **Network LSA (Type 2):** Produced by the DR on every multiaccess network. These LSAs list all attached routers, including the DR; they are flooded within the originating area.
- **Network Summary (Type 3):** Originated ABRs; sent into an area to advertise destinations outside the area; flooded throughout the autonomous system(AS).
- **ASBR Summary (Type 4):** Also originated by ABRs; the destination advertised is an ASBR; flooded throughout the AS.
- **AS External (Type 5):** Originated by ASBRs and advertises an external destination or a default route to an external destination; flooded throughout the AS.
- **NSSA External (Type 7):** Originated by ASBRs in not-so-stubby areas.

Types of Routes

OSPF uses routing designators in the routing table to distinguish between types of routes. Here are the designators used and their meaning. Remember, these can be seen using the **show ip route** command:

- **O:** OSPF intra-area (router LSA)—Networks from within the same area as the router; Type 1 LSAs are used to advertise.
- **O IA:** OSPF interarea (summary LSA)—These are networks outside of the area of the router, but within the AS; Type 3 LSAs are used to advertise.
- **O E1:** Type 1 external routes—Networks outside of the AS; advertised by Type 5 LSAs; calculate cost by adding the external cost to the internal cost of each link that the packet crosses; used when multiple ASBRs are advertising the external route.
- **O E2:** Type 2 external routes—Networks outside of the AS; advertised by Type 5 LSAs; cost is always the external cost only. This is the default type on Cisco routers.

 Please see page 245 for more details.

Areas

Standard OSPF Areas

An area that enables the transmittal of all OSPF LSA types. Any nonzero area must connect to area 0 through an area border router (ABR) or virtual link. An ABR that connects to a standard area advertises network summary (type 3), ASBR summary(type 4), and external summary(type 5) LSAs into the area. Autonomous System Border Routers (ASBR) may be present within a standard area.

To configure an OSPF router to be in a standard area, simply specify the area in the required **network** statement.

```
Router(config)# router ospf <process-id>
Router(config-router)# network <IP> <WILDCARD MASK> area <#>
```

or configure the area within the interface

```
Router(config-int)# ip ospf <process-id> area <#>
```

A router configured to be part of area 0 and another area will become an ABR. Remember that each area must be connected to area 0.

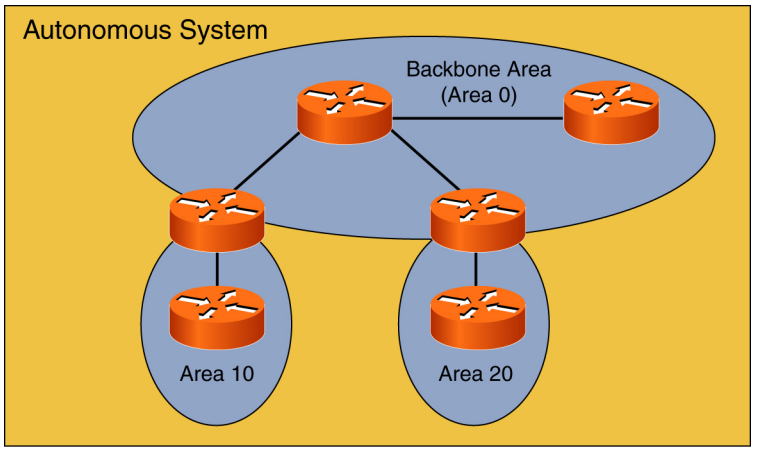
A router that advertises external networks into OSPF becomes an ASBR.

To view the current configuration of areas within the router

```
Router# show ip ospf
```

 Please see page 245 for more details.

FIGURE 4-2
OSPF Areas



- **Stub area:** A stub area is an area that does not permit the advertisement of type 5 (external) LSAs. Instead, these LSAs are replaced with a default route advertisement. Type 3 and 4 advertisements are sent into the area from the ABR. Stub areas are used when all traffic destined to an external network would travel through an ABR. A default route accomplishes this while saving resources.
  - For an OSPF adjacency to form, routers must agree on the area type. This means that all routers within a stub area must be configured as a stub:

```
Router(config)# router ospf <process-id>
Router(config-router)# area <#> stub
```

  - Because stub areas do not enable the propagation of type 5 LSAs, an ASBR cannot be part of a stub area. A Not-so-stubby-area (NSSA) was created for this purpose. Additionally, a virtual link cannot transverse a stub area.
  - **Totally stubby area:** A totally stubby area is a Cisco proprietary feature that extends the concepts of a stub area one step further. In addition to the Type 5 (external summary) LSAs being replaced by the ABR, Type 3 (network summary) and Type 4 (ASBR summary) LSAs are replaced with a default route as well.

To configure an area as totally stubby

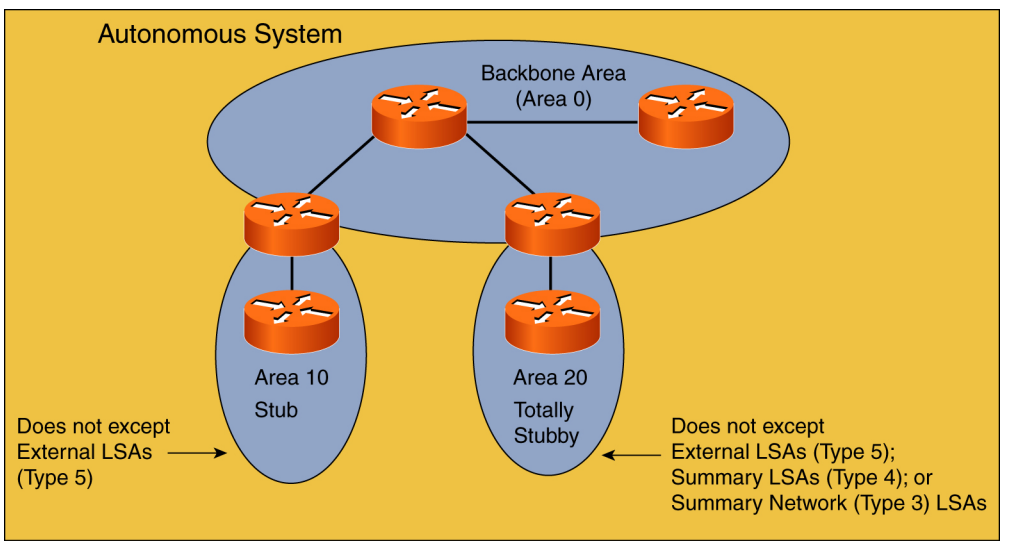
 Please see page 245 for more details.

```
Router(config)# router ospf <process-id>
Router(config-router)# area <#> stub no-summary
```

Unlike the configuration of a stub area, the **no-summary** command is required only on the ABR, not all routers within the area. All other routers (non-ABR) require only the **area <#> stub** command.

Like stub areas, ASBRs and virtual links are not allowed within totally stubby areas.

FIGURE 4-3
Stub and Totally Stubby Areas



- **Not-so-stubby areas:** One of the limitations of stub areas is that they do not enable ASBRs. Because ASBRs advertise Type 5 (external summary) LSAs into an area, they violate the objective of a stub area, namely to disallow such LSAs. There does exist a need, in some networks, to have an ASBR inject external routes into an area, while limiting external routes from ASBRs in other areas. To do this a Type 7, NSSA external, LSA was created. An ASBR can inject Type 7 LSAs into stub areas that are converted to Type 5 LSAs by the ABR connected to the backbone area.

 Please see page 245 for more details.

To configure an area as NSSA

```
Router(config)# router ospf <process-id>
Router(config-router)# area <#> nssa
```

Like the stub area, all routers within the area must agree that the area is NSSA.

When an NSSA area is created, the ABR does not create a default summary route. If a summary route is desired, a totally NSSA area can be used or **default-information-originate** can be added to the command:

```
Router(config-router)# area <#> nssa default-information orginate
```

- **Totally NSSA:** The Totally NSSA area is a Cisco proprietary enhancement to the NSSA concept that extends the NSSA concept to replace Type 3 and Type 4 LSAs with a default route. Like the NSSA area, it does enable Type 7 LSAs to be generated by an ASBR. Unlike the NSSA area, it converts the Type 5, Type 3, and Type 4 LSAs into a default route that is advertised in the area.

To configure an area as NSSA, the **no-summary** option simply needs to be added to the NSSA area command:

```
Router(config)# router ospf <process-id>
Router(config-router)# area <#> nssa no-summary
```

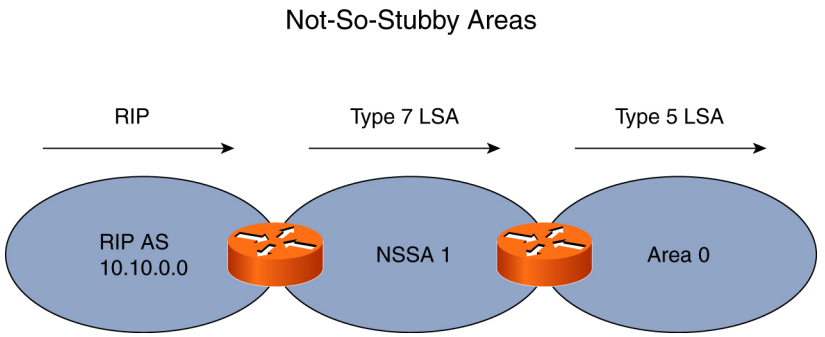
This command is required only on the ABR; all other routers require only the **nssa** option on the area.

Here is a summary of the LSA types permitted in each area.

Area	LSA 1	LSA 2	LSA 3	LSA 4	LSA 5	LSA 7
Backbone	Yes	Yes	Yes	Yes	Yes	No
Nonbackbone	Yes	Yes	Yes	Yes	Yes	No
Stub	Yes	Yes	Yes	Yes	No	No
Totally stubby	Yes	Yes	No	No	No	No
NSSA	Yes	Yes	Yes	Yes	No	Yes

 Please see page 245 for more details.

FIGURE 4-4
Not-so-stubby areas



Configuring Basic Single-Area OSPF

First, you must enable the OSPF routing process on the router using the following global configuration command:

```
router ospf process-id
```

Use the **network** command in router configuration mode to identify those interfaces that are to participate in OSPF:

```
network address inverse-mask area [area-id]
```

Verification commands include the following:

- **show ip protocols**
- **show ip route ospf**
- **show ip ospf interface**
- **show ip ospf**
- **show ip ospf neighbor [detail]**

OSPF Router ID

The router ID is how the router is identified in OSPF. The router ID also is used to break a tie for DR/BDR if the administrator has not set the OSPF priority values on routers using the **ip ospf priority** command. The router with the highest router ID wins the election in that case. Here is the process for router ID selection:

 Please see page 245 for more details.

1. The router ID is set with the router-id address router configuration command. If you use this command after OSPF has selected azz router ID, you should use **clear ip ospf process** to reset.
2. The highest IP address on a loopback interface.
3. The highest IP address on an active interface.

Use **show ip ospf** to verify the router ID selection.

Route Summarization

Two types of summarization exist in OSPF: interarea, which is performed on ABRs, and external route summarization, which is performed on routes redistributed into OSPF autonomous systems.

To configure interarea route summarization on the ABR, use the following router configuration command:

```
area area-id range address mask
```

To configure route summarization on an ASBR to summarize external routes, use the following router configuration command:

```
summary-address address mask [not-advertise] [tag tag]
```

The **not-advertise** optional keyword suppresses routes that match the specified prefix. The **tag** value can be used as a “match” value for controlling redistribution with route maps on the ABR.

Default Route Advertisements in OSPF

For an OSPF router to advertise a default route into an area, the command **default-information originate** must be used. If the advertising router does not possess a default route in its routing table, you can use the **always** keyword to still generate the default route to 0.0.0.0. The complete router configuration command syntax for generating default routes is as follows:

```
default-information originate [always] [metric metric_value] [metric-type type-value] [route-map map-name]
```

 Please see page 245 for more details.

If you do not specify a metric value, the default of 10 is used. The **metric-type** enables you to specify a Type 1 or Type 2 external route. Finally, the **route-map** option enables you to control the generation of the default route further. For example, the default route is generated only if the route map is satisfied.

Authentication

Type 1: clear text; least secure. To configure:

Step 1. Enable area authentication on all routers in the area; use the following router configuration command:

```
area area_id authentication
```

Step 2. Enter the clear-text password on the interface in interface configuration mode:

```
ip ospf authentication-key password
```

Type 2: MD5; most secure. To configure:

Step 1. Enable MD5 area authentication on all routers in the area using router configuration mode:

```
area area_id authentication message-digest
```

Step 2. Set the key and password on the interfaces using interface configuration mode:

```
ip ospf message-digest-key key_value md5 password
```

Changing the Cost Metric

The Cisco implementation of OSPF calculates the metric using the following formula:

```
cost = reference bandwidth / bandwidth
```

The default reference bandwidth is 100 Mbps. The bandwidth value is that which is configured on the interface using the **bandwidth** command. If you use many interfaces faster than 100 Mbps, consider resetting the reference bandwidth value. You can do so on each router using the following router configuration mode command:

```
auto-cost reference-bandwidth refbw
```

 Please see page 245 for more details.

Reference bandwidth is in megabits per second. For example, if you want to ensure Gigabit Ethernet interfaces evaluate to a cost of 5, set the *refbw* on each router to 5000. (Valid values are from 1 to 4,294,967.) You can also override the calculated cost value in any interface directly by using the following interface configuration command:

```
ip ospf cost value
```

Values range from 1 to 65,535.

Optional OSPF Interface Parameters

Additional optional interface parameters not covered elsewhere in this Quick Reference include the following:

- **ip ospf retransmit-interval:** Specifies the number of seconds between LSA retransmissions.
- **ip ospf transmit-delay:** Sets the number of seconds required to send a link-state update.
- **ip ospf hello-interval:** Specifies the time between hello packets; must match on all routers in the network.
- **ip ospf dead-interval:** Number of seconds before the router is considered dead; must match on all routers in the network.

Administrative Distance and OSPF

Three different administrative distance values are possible for OSPF: intra-area routes, interarea routes, and external routes. By default, all are set to 110; these can be changed with the following router configuration command:

```
distance ospf {[intra-area dist1] [inter-area dist2]
[external dist3]}
```

OSPF Passive Interface

To set a passive interface in OSPF, use the following router configuration command:

```
passive-interface interface-type interface-number
```

When used with OSPF, this command prevents the interface from sending hello packets and therefore prevents an adjacency from forming. It also prevents the sending or receiving of routing information through the interface. The specified interface address appears as a stub network in the OSPF domain.

 Please see page 245 for more details.

Configuring Route Calculation Timers

You can configure the delay between when a topology change is received and when the SPF calculation takes place. You can also configure the hold time between two consecutive SPF calculations. Use the following router configuration command:

```
timers spf spf-delay spf-holdtime
```

Changing LSA Group Pacing

Routers group LSAs and pace refreshing, checksumming, and aging functions so that the resource strain on the router is reduced. This is default behavior; it can be tweaked with the following router configuration command:

```
timers lsa-group-pacing seconds
```

Blocking LSA Flooding

You can prevent the default flooding behavior; to do so on a broadcast, nonbroadcast, or point-to-point network, use the following interface configuration command:

```
ospf database-filter all out
```

On point-to-multipoint networks, use the following router configuration command:

```
neighbor ip-address database-filter all out
```

Reducing LSA Flooding

Reduces the flooding of LSAs in stable topologies by setting LSAs to Do Not Age; this is accomplished with the following interface configuration command on a per-interface basis:

```
ip ospf flood-reduction
```

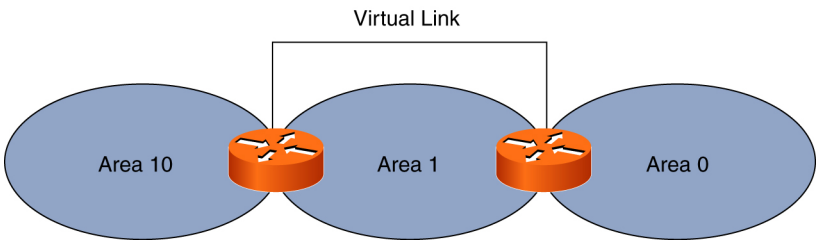
Virtual Links

A virtual link is a link to the backbone through a nonbackbone area. Virtual links are created between two ABRs, and the area cannot be a stub. Virtual links are typically implemented as a temporary fix for OSPF design issues. For example, they can be used to connect an area that has no direct connection to the backbone area. Or they can be used to connect to disconnected area 0s (backbones). The following command configures a virtual link:

```
area transit_area_id virtual-link router_id_of_remote
```

 Please see page 245 for more details.

FIGURE 4-5
Virtual Links



OSPF over On-Demand Circuits

On-demand circuit is an enhancement that enables efficient operations over dialup, ISDN, and other on-demand circuits. With this feature, periodic hellos are suppressed, and the periodic refreshes of LSAs are not flooded over the demand circuit. These types of packets bring up the link only the first time—or when you have a topology change that needs to be propagated.

To configure OSPF for on-demand circuits on a per-interface basis, use the following interface configuration command:

```
ip ospf demand-circuit
```

If the router is part of a point-to-point topology, only one end of the demand circuit must be configured with this command, although all routers must support the feature. If the router is part of a point-to-multipoint topology, only the multipoint end must be configured with this command. Also, this feature does not work in a broadcast-based topology. Finally, the feature is not supported for use with an asynchronous interface.

OSPF Graceful Restart

RFC 3623 defines OSPF Graceful Restart. This functionality is incorporated into Cisco routers because of the Nonstop Forwarding (NSF) capability that Cisco has engineered into the Border Gateway Protocol (BGP), Enhanced Interior Gateway Routing Protocol (EIGRP), OSPF, and Intermediate System-to-Intermediate System (IS-IS) protocols.

 Please see page 245 for more details.

The idea behind OSPF Graceful Restart/NSF is to enable the router to continue forwarding packets, even while undergoing specific well-known failure conditions. Perhaps a software upgrade is occurring, or a route processor crash is affecting the router. NSF enables for the continued forwarding of packets.

Before RFC 3623, Cisco offered a proprietary version of NSF. Cisco now refers to this version as Cisco NSF. The OSPF RFC 3623 Graceful Restart feature enables you to configure IETF NSF in multivendor networks. Cisco now refers to this version as IETF NSF.

OSPF NSF operates in one of two modes for failover operations. The first possible mode is Restarting mode. In Restarting mode, the OSPF router process performs nonstop forwarding recovery because of a route processor switchover. The second possible mode is Helper mode. In Helper mode, a neighboring router restarts, and the Helper mode router assists in the nonstop forwarding recovery process.

Enabling IETF NSF on the Cisco router is simple. Enter router configuration mode for the OSPF process and issue the following command:

```
nsf ietf
```

Troubleshooting OSPF Route Advertisements

OSPF neighbor is not advertising routes:

- OSPF is not enabled on interface.
- Advertising interface is down.
- Secondary interface is in a different area from primary interface.

ABR is not advertising summary route:

- Area is configured as totally stubby area.
- ABR lacks area 0 connectivity.
- A discontiguous area 0 exists.

 Please see page 245 for more details.

Neighbor is not advertising external routes:

- Area is configured as stub or NSSA.
- The NSSA ABR is not translating Type 7 into Type 5 LSAs.

Neighbor is not advertising default routes:

- No **default-information originate** command.
- No default route in the routing table.
- Stub area is in use.
- NSSA border router is not originating Type 7.

Troubleshooting OSPF Route Installation

OSPF installing no routes in routing table:

- Network type mismatch
- IP address or subnet mask misconfiguration
- Unnumbered/numbered point-to-point configuration
- Distribute list
- Broken permanent virtual circuit (PVC) in full-mesh broadcast mode Frame network

OSPF not installing external routes:

- Forwarding address not known through intra-area or interarea route
- ABR not generating Type 4 LSAs

 Please see page 245 for more details.

Troubleshooting Redistribution

Not advertising external routes:

- **subnets** keyword is missing.
- Distribute list.

Troubleshooting Route Summarization

Router not summarizing interarea routes:

- No **area range** command on ABR
- Router not summarizing external routes
- No **summary-address** command on ASBR

Troubleshooting CPUHOG Syslog Reports

CPUHOG messages during adjacency establishments:

- No packet-pacing code executing
- CPUHOG messages during LSA refresh
- No LSA group-pacing code

Troubleshooting Dial-on-Demand Routing Issues

Hello packets are bringing up the link:

- Hellos are permitted as interesting traffic.

Demand circuit keeps bringing up the link:

- Link flapping.
- Network type is broadcast.

 Please see page 245 for more details.

- PPP host route redistributed.
- One router is not demand-circuit-capable.

Troubleshooting SPF Calculations

SPF running constantly:

- Flapping route
- Neighbor flapping
- Duplicate router ID

Troubleshooting Common Error Messages

Could Not Allocate Router ID:

- No enabled interface with valid IP
- Not enough interfaces up with IP addresses for multiple OSPF processes

"%OSPF-4-BADLSATYPE: Invalid lsa: Bad LSA type" Type 6:

- Neighboring router is sending MOSPF packets not supported on Cisco routers.
- Eliminate the error with the **ignore lsa mospf** command.

"OSPF-4-ERRRCV":

- OSPF received an invalid packet because of a mismatched area ID, a bad checksum, or OSPF not enabled on a receiving interface.

"Bad Checksum":

- Device is corrupting the packet.
- Sending router's interface is bad, or a software bug exists.

 Please see page 245 for more details.

- Receiving router's interface is bad, or a software bug exists.

General troubleshooting commands

```
show ip ospf neighbor [interface-type interface-number] [neighbor-id] [detail] : Displays OSPF neighbor
information on a per-interface basis.
show ip ospf [process-id]
```

Displays general information about OSPF routing processes.

```
show ip ospf interface [interface-type interface-number]
```

Displays OSPF-related interface information.

```
show ip ospf database
```

Displays lists of information related to the OSPF database for a specific router.

```
debug ip ospf packet
```

This EXEC command displays information about each OSPF packet received:

```
Router# debug ip ospf packet
OSPF:  rcv. v: 2 t: 1 l: 48 rid: 200.0.0.116
aid: 0.0.0.0 chk: 0 aut: 2 keyid: 1 seq: 0x0
```

The possible output values are as follows:

- v: Version of OSPF
- t: Specifies the OSPF packet type (1: Hello, 2: DBD, 3: LSR, 4: LSU, 5: LAAck)
- rid: Provides the OSPF router ID
- aid: Shows the area ID

 Please see page 245 for more details.

- chk: Displays the checksum
- aut: Provides the authentication type (0: no, 1: simple password, 2: MD5)
- auk: Specifies the authentication key
- keyed: Displays the MD5 key ID
- seq: Provides the sequence number

BGP

Border Gateway Protocol (BGP) is an Exterior Gateway Protocol (EGP) used for routing between autonomous systems. It enables routing policies and improves security.

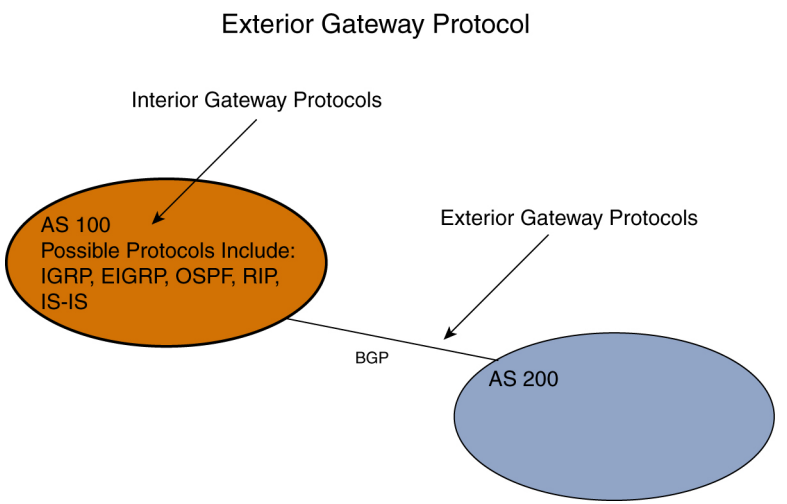


FIGURE 4-6
Exterior Gateway Protocol

 Please see page 245 for more details.

BGP is an advanced path vector protocol and includes the following:

- Reliable updates
- Triggered updates only
- Rich metrics (path attributes)
- Scalable to massive networks

Because of these enhancements, BGP is often described as advanced distance vector. Perhaps the most technically accurate description is path vector.

Common uses for BGP include the following:

- Customer connected to one Internet service provider (ISP) (not always required, however)
- Customer connected to several ISPs
- Service provider networks (transit AS)
- Network cores of very large enterprise networks

Session Establishment

BGP neighbors are not discovered; they must be configured manually on both sides of the connection. TCP port number 179 is used. Only one session remains if both connection attempts succeed. The **show ip bgp summary** command gives an overview of the session status. Indications include Idle, Active, OpenSent, OpenConfirm, and Established. Keepalives are sent every 60 seconds. Peers can use an MD5 shared secret.

Route Processing

All routes received after the neighbor establishment are saved in memory. If more than one way to reach a destination exists, the best is selected. Use the **show ip bgp** command to view all the routing information received from all neighbors.

 Please see page 245 for more details.

The best route selection criteria occurs in this order:

- Exclude any route with inaccessible next hop
- Prefer highest weight (local to router)
- Prefer highest local preference (global within AS)
- Prefer routes that the router originated
- Prefer shortest AS paths (compare length only)
- Prefer lowest origin code (IGP < EGP < Incomplete)
- Prefer lowest Multiexit Discriminator (MED)
- Prefer external paths over internal BGP (iBGP) paths
- For iBGP paths, prefer path through closest IGP neighbor
- For external BGP (eBGP) paths, prefer the oldest path
- Prefer paths from router with lower BGP router ID

The best routes (valid and reachable) are propagated to BGP neighbors.

The best BGP routes are copied into the IP routing table after the router checks administrative distance values.

The BGP process injects local routes in two different ways:

- Using the **network** configuration commands. This command lists networks that are candidates if they appear in the routing table.
- Using redistribution by another routing protocol.

 Please see page 245 for more details.

Route Summarization

Automatic classful summarization is enabled by default. When you disable automatic summarization, the routes introduced locally into the BGP table are not summarized.

Internal BGP (IBGP) Versus External BGP (EBGP)

BGP operates by establishing peer relationships with other BGP routers in either an external (eBGP) or internal (iBGP) manner. Internal BGP peers are those that share the same AS (AS) number. By contrast, external BGP peers are those which do not share the same AS number. Although these are minor configuration differences, they are handled in different ways:

1. Packets sent to eBGP peers use a TTL of 1.
2. The next-hop field is updated with the last eBGP peer. It is not updated when iBGP is used.
3. eBGP neighbors do not advertise routes to eBGP neighbors in a AS that is contained with in the AS_PATH.
4. iBGP routes have an AD of 200; eBGP routes have an AD of 20.
5. iBGP routes are subject to BGP synchronization (if enabled).

BGP synchronization is the major difference between eBGP and iBGP routes and is characterized by the BGP synchronization rule:

For an iBGP route to be added to the BGP table, the exact prefix must be in the routing table from an IGP.

The synchronization rule is a method that guarantees that a route is known to all routers within the AS even if they are not running BGP. If a route is advertised via iBGP and a non-BGP router sits logically between the BGP peers, the non-BGP router will black hole the traffic because the destination is not known via IGP first. The synchronization check can be turned off (and is by default as of IOS version 12.2(8)T) with the router configuration command:

```
Router(config-router)# no synchronization
```

 Please see page 245 for more details.

If disabled, it must be guaranteed that a routing black hole exists within the AS by creating a full-mesh iBGP network or using a BGP tool such as route reflectors or confederations.

BGP Basic Configuration

To start BGP on your router, use the following global configuration command:

```
router bgp as-number
```

A public AS number can be obtained from the appropriate agency, or a private AS number is possible in some situations (64,512 to 65,535). Only one BGP process is permitted per router.

To configure your BGP neighbors, use the following router configuration commands:

```
neighbor ip-address remote-as as-number
neighbor ip-address description neighbor description
```

To temporarily disable a neighborship, use the following router configuration command:

```
neighbor ip-address shutdown
```

To configure MD5 authentication between neighbors, use the following router configuration command. Keep in mind the password string must match on both routers.

```
neighbor ip-address password string
```

Announcing Networks

To disable automatic summarization, use the following router configuration command:

```
no auto-summary
```

To manually define a network for advertisement by BGP, use the following router configuration command:

```
network network-number [mask network-mask]
```

 Please see page 245 for more details.

If you use this command and auto-summarization is on (the default behavior), at least one of the subnets must be present in the forwarding table for the major network prefix to be advertised. If auto-summarization is disabled, an exact match is required in the forwarding table. You can use the **mask** keyword to specify a specific subnet with the **network** command.

If you would like to modify attributes before inserting prefixes into the BGP table, you can use a route map in the **network** command in router configuration mode:

```
network network-number [mask network-mask] [route-map map-tag]
```

This option might be used for one or more of the following:

- Change the weight of a locally sourced route.
- Manipulate source routes with BGP communities.
- Set the local preference.
- Change the value of the MED.

To advertise routes based on route redistribution, examine the following sample command syntax:

```
Router(config)# router bgp 64500
Router(config-router)# redistribute ospf 1
Router(config-router)# distribute-list prefix MY_PREFIX_LIST out
```

One caveat here is that the routes have an origin code of unknown. This makes them seem inferior to other routes per the BGP route-selection process. Notice the optional use of the distribute list syntax to suppress certain networks from being advertised in updates.

Redistribution can be configured with a route map to reset the origin code or set other attributes. Here is an example:

```
Router(config)# router bgp 64500
Router(config-router)# redistribute ospf 1 route-map MY_ROUTE_MAP
```

 Please see page 245 for more details.

Classless BGP

To manually announce a classless prefix, be sure to use the following router configuration command:

```
= network ip-prefix-address mask subnet-mask
```

You should also consider creating a static route pointing to null0 to create a matching prefix in the IP forwarding table to ensure the subnet is advertised.

Aggregation in BGP

Use the following router configuration command to configure route summarization to suppress the advertising of individual networks. Remember, at least one network of the summarized space must exist in the BGP table:

```
aggregate-address address-prefix mask summary-only
```

Route Selection Using Policy Controls

AS Path Filtering with Regular Expressions

String matching: A string of characters in the regular expression matches any equivalent substring in the AS path; 29 has three matches in | 210 291 1296 29 |, for example.

String matching alternatives: The pipe symbol (|) means "or."

String matching ranges and wildcards: Brackets ([ ]) can be used for ranges, and the period (.) can match any single character.

String matching delimiters: The caret (^) matches the beginning of string, the dollar sign ($) matches the end of the string, and an underscore (_) matches any delimiters.

String matching grouping: Parentheses can group smaller expressions into larger expressions.

 Please see page 245 for more details.

String matching special characters: You can use the backslash (\) to remove the special meaning of the character that follows.

String matching repeating operators: An asterisk (*) means the expression preceding repeats zero or more times; a question mark (?) means the expression preceding repeats zero or one time;, and a plus sign (+) means the expression preceding repeats one or more times.

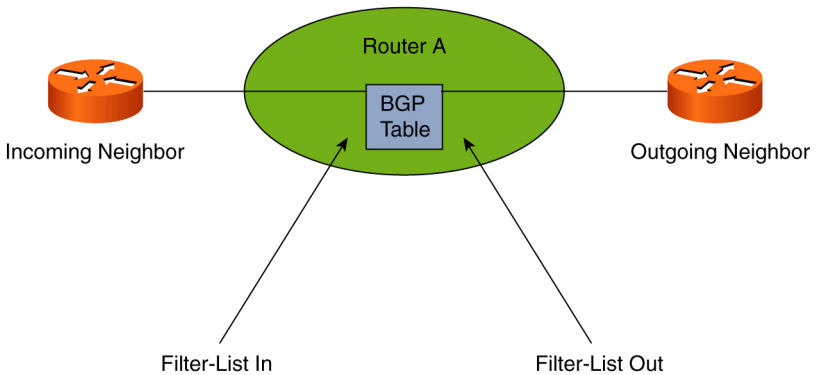
Here are some string matching examples:

_200_	All routes going through AS 200
^200$	Directly connected to AS 200
_200$	Originated in AS 200
^200_.	Networks behind AS 200
^[0–9]+$	AS paths one autonomous system long
^([0–9]+)(_\1)*$	Networks originating in the neighbor AS
^$	Networks originated in local AS
.*	Matches everything

AS path filters configured inbound on a router select those routes that are allowed.

 Please see page 245 for more details.

FIGURE 4-7
AS-Path Filters



Routes selected enter the local BGP table when the selection is applied on the incoming routes from a neighbor. Routes not selected are silently dropped. Routes selected if an outbound filter is used are transmitted to the neighbor when the selection is applied. Routes not selected are used locally but are never sent to the neighbor.

The commands used to configure an AS path list are relatively simple. First, configure an AS path access list as follows in global configuration mode:

```
ip as-path access-list access-list-number {permit | deny}
as-regular-expression
```

To set up a BGP filter, use the **neighbor filter-list** router configuration command:

```
neighbor {ip-address | peer-group-name} filter-list access-list-number {in | out}
```

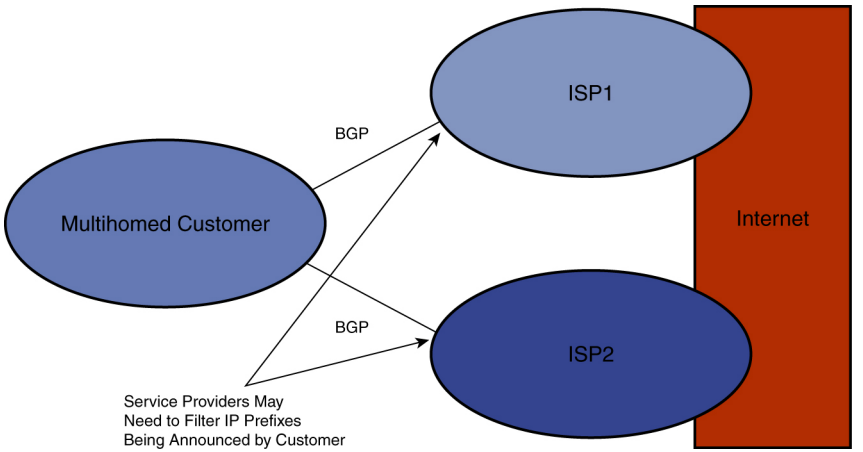
Monitoring the use of regular expressions is critical. To display routes matching the AS path regular expression, use the **show ip bgp regexp** command. To display routes that conform to a specified filter list, use the **show ip bgp filter-list** command. To display a specific access list or all AS path access lists in the router, use the **show ip as-path-access-list** command.

Prefix Lists

Prefix lists are a powerful method to control the updates coming from other BGP speaking routers.

 Please see page 245 for more details.

FIGURE 4-8
Prefix filtering



To create an entry in a prefix list, use the **ip prefix-list global configuration** command:

```
ip prefix-list list-name [seq seq-value] deny | permit
network/len [ge ge-value] [le le-value]
```

You can use the parameters **ge** (greater than) and **le** (less than) to specify the range of the prefix length to be matched for prefixes that are more specific than *network/len*. The exact match is assumed when neither **ge** nor **le** is specified. The range is assumed to be from *ge-value* to 32 only if the **ge** attribute is specified. The range is assumed to be from **le** to *le-value* only if the **le** attribute is specified.

To distribute BGP neighbor information as specified in a prefix list, use the following router configuration command:

```
neighbor {ip-address | peer-group-name} prefix-list prefix-listname {in | out}
```

This might be useful to suppress a more specific route or to change the path used to reach a certain destination.

To suppress networks from being advertised in updates, use the following router configuration command:

```
distribute-list {access-list-number | name | prefix-list
prefix-listname} out [interface-name | routing-process | autonomous-system-number]
```

To display information about a prefix list or prefix list entries, use the **show ip prefix-list** command.

 Please see page 245 for more details.

Outbound Route Filtering

Outbound Route Filtering (ORF) is a prefix-based BGP feature enabled through the advertisement of ORF capabilities to peer routers. The advertisement of the ORF capability indicates that a BGP-speaking router can accept a prefix list from a neighbor and apply the prefix list to locally configured ORFs (if any exist). When this capability is enabled, the BGP speaker can install an inbound prefix list filter to the remote peer as an outbound filter, which reduces unwanted routing updates.

An ORF message contains the following information:

- Address Family Information (AFI, IPv4 or IPv6) and Subsequent Address Family Information (SAFI, Unicast, Multicast, and so on) for which the filter should be used
- ORF type
- When to refresh (immediate or deferred refresh)
- List of ORF entries where the actual filter is defined

Commonly used ORF types are as follows:

- ORF type 1 filters based on Network Layer Reachability Information (NLRI)
- ORF type 2 filters based on standard BGP community attributes
- ORF type 3 filters based on extended BGP community attributes
- ORF type 128 filters based on Cisco proprietary implementation of prefix filtering (prefix lists)

An ORF type of NLRI-based filtering (type 1) uses the following actions:

- **ADD:** Adds a line to a prefix list filter on the remote peer
- **DELETE:** Removes a line from a filter that was previously installed on a remote peer
- **DELETE ALL:** Removes all previously installed filters on the remote peer

 Please see page 245 for more details.

To advertise ORF capabilities to a peer router, use the **neighbor orf *prefix-list*** command in address family or router configuration mode:

```
neighbor {ip-address} [capability] orf prefix-list [receive | send | both]
```

Use the **clear ip bgp neighbor** command with the **prefix-filter** keyword to push out the existing ORF prefix list so that a new route refresh can be received from a neighbor. The neighbor uses the ORF prefix list previously negotiated.

Filtering with Route Maps

Route maps are also a power filtering tool. They can be used to accomplish the following tasks:

- Filter on IP prefixes coming from a specific AS
- Filter on other BGP attributes
- Modify BGP attributes

Match clauses in the BGP route map can be based on the following:

- IP network numbers and subnet masks (prefix list or access list)
- Route originator
- Next hop
- Origin code
- Tag value attached to an Interior Gateway Protocol (IGP) route
- AS path
- Community
- IGP route type

 Please see page 245 for more details.

With a route map, the following can be set:

- Origin
- Next hop
- Weight
- Community
- Local preference
- MED

You can apply a route map on incoming or outgoing routing information for a neighbor. The routing information must be permitted by the route map to be accepted. If the route map has no statement explicitly permitting a route, the route is implicitly denied and dropped. The syntax required is as follows:

```
Router(config-router)# neighbor ip-address route-map name in | out
```

The **show ip bgp route-map** command displays selected routes from a BGP routing table based on the contents of a route map.

Implementing Changes in Policy

The traditional method of **clear ip bgp *** is disruptive. Soft reconfiguration was introduced in Cisco IOS Release 11.2 to facilitate nondisruptive changes in BGP. When you configure **soft-reconfiguration inbound** for a neighbor, the router stores all routes received from that neighbor as an extra copy in memory. This copy is taken before any filtering is applied by the router to routes it receives. When you have completed the changes to filters and route maps applied on incoming information, use **clear ip bgp ip-address soft** on the router in privileged EXEC mode.

When you have completed the changes to filters and route maps applied on the outgoing information, execute **clear ip bgp ip-address soft out** on the router in privileged EXEC mode.

 Please see page 245 for more details.

Route refresh is another new feature in the Cisco implementation of BGP. Routers use the route refresh feature to ask a neighbor to resend all the routing information when needed. Use the **clear ip bgp *** command to send a route refresh message to all neighbors or **clear ip bgp ip-address** to send a route refresh message to a specific neighbor.

BGP Path Attributes

Mandatory Well-Known Attributes

Origin: Specifies the router's origin

- IGP
- EGP
- Unknown—Route was redistributed

AS-Path: Sequence of AS numbers through which the route is accessible

Next-Hop: IP address of the next-hop router

Discretionary Well-Known Attributes

Local Preference: Used for consistent routing policy with an AS

Atomic Aggregate: Informs the neighbor AS that the originating router aggregated routes

Nontransitive Attributes

Multiexit Discriminator: Used to discriminate between multiple entry points into an AS

Transitive Attributes

Aggregator: IP address and AS of the router that performed aggregation

 Please see page 245 for more details.

Community: Used for route tagging

Influencing Route Selection Using Weights

Using Weight

You can use weight to provide local routing policy, and you can use local preference to establish AS-wide routing policy.

To assign a weight to a neighbor connection, use the **neighbor weight router configuration** command:

```
neighbor {ip-address | peer-group-name} weight weight
```

This approach assigns a weight value to all route updates from the neighbor. Higher weights are preferred.

You can also configure the router so that all incoming routes that match an AS filter receive the configured weight. Use the following router configuration command to do so:

```
neighbor {ip-address | peer-group-name} filter-list access-list-number {in | out | weight weight}
```

You can also set a weight with a route map in more complex scenarios.

The default weight value is 32,768 for locally originating networks (including those via redistribution) and is 0 for all other networks.

Using Local Preference

You can use local preference to influence route selection within the local AS; this attribute is stripped from outgoing updates via eBGP. You should decide between the use of weight or local preference. The default local preference for iBGP and local routes is 100; all others are 0 by default.

You can apply local preference in the following ways:

- Using a route map with the **set local-preference** command
- Using the **bgp default local-preference** command to change the default local preference value applied to all updates coming from external neighbors or originating locally

For verification, you can use the command **show ip bgp prefix** to display the locally applied value.

 Please see page 245 for more details.

AS Path Prepending

In networks where connections to multiple providers are required, it is difficult to specify a return path to be used for traffic returning to the AS. One BGP mechanism you can use is AS path prepending. AS path prepending potentially enables the customer to influence the route selection of its service providers.

You manipulate AS paths by prepending AS numbers to existing AS paths. Typically, you perform AS path prepending on outgoing eBGP updates over the unwanted return path. Because the AS paths sent over the unwanted link become longer than the AS path sent over the preferred path, the unwanted link is now less likely to be used as the return path. To avoid conflicts with BGP loop-prevention mechanisms, no other AS number, except that of the sending AS, should be prepended to the AS path attribute.

You can configure manual manipulation of the AS path attribute (prepending) using a route map with the **set as-path prepend** command.

BGP Multi Exit Discriminator (MED)

You can apply the MED attribute on outgoing updates to a neighboring AS to influence the route selection process in that AS. The MED attribute is useful only when you have multiple entry points into an AS.

The default value of the MED attribute is 0. A lower value of MED is more preferred. A router prefers a path with the smallest MED value but only if weight, local preference, AS path, and origin code are equal.

MED is not a mandatory attribute; no MED attribute is attached to a route by default. The only exception is if the router is originating networks that have an exact match in the routing table (through the **network** command or through redistribution). In that case, the router uses the metric in the routing table as the MED attribute value.

Using the **default-metric** command in BGP configuration mode causes all redistributed networks to have the specified MED value. You can use a route map to set MED on incoming or outgoing updates. Use the **set metric** command within route map configuration mode to set the MED attribute. You must use the command **bgp bestpath med confed** when you use MED within a confederation to influence the route selection process. A router compares

 Please see page 245 for more details.

MED values for those routes that originate in the confederation.

BGP Communities

A community is an attribute used to set an identifier's BGP routes. A router can apply it to any BGP route by using a route map. Other routers can then perform any action based on the tag (community) that is attached to the route.

Any BGP router can tag routes in incoming and outgoing routing updates or when doing redistribution. In addition, any BGP router can filter routes in incoming or outgoing updates or select preferred routes based on the community values. By default, communities are stripped in outgoing BGP updates.

The actual community attribute is a transitive optional attribute. The value of this attribute is a 32-bit number in the possible range of 0 to 4,294,967,200. You can tag each network in a BGP routing table with a set of communities. The default community is Internet (0).

The BGP standards define several well-known communities for your use:

- **no-export:** Do not advertise routes to real eBGP peers.
- **no-advertise:** Do not advertise routes to any peer.
- **local-as:** Do not advertise routes to any eBGP peers.
- **internet:** Advertise this route normally; this is the default community value.

Because the community attribute is a transitive optional attribute, routers that do not support communities pass them along unchanged.

To define your own communities, you use a 32-bit community value split into two parts:

- High-order 16 bits that contain the AS number of the AS that defines the community meaning
- Low-order 16 bits that have local significance

 Please see page 245 for more details.

You can specify a 32-bit community value as follows:

[AS-number]: [low-order-16-bits]

You use communities in a well-planned step-by-step way. Here are the steps that you should consider and examples of each:

Step 1. Define administrative policy goals.
Example: Solve asymmetric customer routing problems.

Step 2. Design filters and path selection policy to achieve administrative goals.
Example: Set local preference of customer routes to 75 for customers using the backup ISP.

Step 3. Define communities to be used to achieve individual goals.
Example: Community 367: 20 indicates that the local preference of the route should be lowered to 75.

To actually configure BGP communities, you can use the following steps:

Step 1. Configure route tagging with BGP communities.

Step 2. Configure BGP community propagation.

Step 3. Define BGP community access lists (community lists) to match BGP communities.

Step 4. Configure route maps that match on community lists and filter routes or set other BGP attributes.

Step 5. Apply route maps to incoming or outgoing updates.

Route tagging with communities is always done with a route map. You can specify any number of communities; communities specified in the **set** keyword overwrite existing communities unless you specify the **additive** option.

After you create the route map, you can apply it to inbound or outbound BGP updates using the following router configuration command:

```
neighbor ip-address route-map map in | out
```

 Please see page 245 for more details.

To apply a route map to redistributed routes, use the following router configuration command:

```
redistribute protocol route-map map
```

By default, communities are stripped in outgoing BGP updates; therefore, you must manually configure community propagation to BGP neighbors. You can do so using the following command:

```
neighbor ip-address send-community
```

Keep in mind that BGP peer groups are ideal for configuring BGP community propagation toward a large number of neighbors.

You can use a standard community access list to find community attributes in routing updates. A standard community list is defined by its assigned list number. The list number uses a range from 1 to 99. Community lists are similar to standard IP access lists in these ways:

- The router evaluates the lines in the community list sequentially.
- If no line matches communities attached to a BGP route, the route is implicitly denied.

Standard community lists differ from standard IP access lists in these ways:

- The keyword **internet** should be used to permit any community value.
- If more values are listed in a single line, they all have to be in an update to have a match.

Here is the global configuration mode syntax for the creation of the standard community list:

```
ip community-list 1-99 permit | deny value [ value ... ]
```

To create an extended community list, use the following global configuration mode syntax:

```
ip community-list 100-199 permit | deny regexp
```

These extended community lists are like simple community lists, but they match based on regular expressions.

 Please see page 245 for more details.

Specifically, communities attached to a route are ordered, converted to a string, and matched with regexp. You can use the **.*** syntax to match any community value. Community lists are used in match conditions in route maps to match on communities attached to BGP routes.

After you create your community lists, you can match to these lists in your route maps. A route map with a community list matches a route if at least some communities attached to the route match the community list. You can use the **exact** option to ensure that all communities attached to the route have to match the community list. Remember, you can use route maps to filter routes or set other BGP attributes based on communities attached to routes.

Route Reflectors

BGP requires that all BGP peers in the same AS form an iBGP session with all peers in the AS. This is too difficult in many environments. Route reflectors are fully functional iBGP speakers that form iBGP sessions with other iBGP speakers, and they also perform a second function—they forward routes from other iBGP speakers to route reflector clients. The route reflector clients form iBGP sessions only with the route reflectors. The route reflectors and the clients form a cluster.

To configure route reflectors, consider these initial tasks:

- Configure the proper cluster ID value on the route reflectors.
- Configure the route reflector with information about which iBGP neighbor sessions are reaching their clients.
- In the clients, remove all iBGP sessions to neighbors that are not a route reflector in the client cluster.
- Make sure that the iBGP neighbor is removed on both ends of the iBGP session.

The command used to configure the cluster ID if the BGP cluster has redundant route reflectors is as follows:

```
bgp cluster-id cluster-id
```

 Please see page 245 for more details.

The command used to configure the router as a BGP route reflector and configure the specified neighbor as its client is as follows:

```
neighbor ip-address route-reflector-client
```

Confederations

Confederations are another method of solving the iBGP full-mesh requirement. Confederations are smaller subautonomous systems created within the primary AS to decrease the number of BGP peer connections. Five steps are used in the configuration of confederations:

Step 1. Enable BGP using the member AS number.

Step 2. Configure the confederation identifier using **the bgp confederation identifier** command.

Step 3. Configure fully meshed iBGP sub-AS neighbor relationships using the sub-AS number as the remote AS number (ASN) for all internal iBGP peers.

Step 4. Configure other neighbors within the same parent AS by specifying their sub-AS number as the remote AS number; other confederation peers from different sub-ASs must also be identified as external confederation peers using the **bgp confederation peers** command.

Step 5. Configure any eBGP neighbors as you normally would.

Peer Groups

To configure one router with multiple BGP peer relationships, configurations can be quite complex. Peer groups simplify the configuration process. You make peer groups and assign neighbors with the same policies to the group. Peer group members inherit the policies assigned to the group.

To configure BGP peer groups on Cisco IOS routers, complete the following steps:

Step 1. Create a BGP peer group; use the **neighbor peer-group router** configuration command.

Step 2. Specify parameters for the BGP peer group.

 Please see page 245 for more details.

Step 3. Create a BGP neighbor.

Step 4. Assign a neighbor to the peer group; use the **neighbor peer-group router** configuration command.

network backdoor Command

The **network backdoor router** configuration command causes the administrative distance assigned to the network to be forced to 200. The goal is to make IGP-learned routes preferred. A network marked as a backdoor is not sourced by the local router, but should be learned from external neighbors. You should be sure to verify the route is in the BGP table for the command to have the desired effect.

Configuring the BGP maximum-prefix Function

To control how many prefixes a BGP router can receive from a neighbor, use the **neighbor maximum-prefix** router configuration command.

Route Dampening

Flapping routes create problems for BGP. An approach was created to remove the update about a flapping route until it can be guaranteed that the destination is more stable. This additional BGP scalability mechanism, called route flap dampening, was created to reduce route update processing requirements by suppressing unstable routes.

To enable route dampening, use the **bgp dampening** command.

Troubleshooting and Monitoring BGP

Important commands not included elsewhere in the BGP Short Cuts include the following:

- **show ip bgp neighbors ip-address:** Displays detailed neighbor information
- **show ip bgp:** Displays all the routes in the BGP table
- **show ip bgp ip-prefix [mask subnet-mask]:** Displays detailed information about all paths for a single prefix
- **debug ip tcp transactions:** Displays all TCP transactions

 Please see page 245 for more details.

- **debug ip bgp events:** Displays significant BGP events
- **debug ip bgp keepalives:** Debugs BGP keepalive packets
- **debug ip bgp updates:** Displays all incoming or outgoing BGP updates
- **debug ip bgp updates acl:** Displays all incoming and sent updates matching an ACL
- **debug ip bgp ip-address updates [acl]:** Displays all BGP updates received from or sent to a specific neighbor

EIGRP

Enhanced Interior Gateway Routing Protocol (EIGRP) is a hybrid routing protocol—combining features of both distance vector and link-state routing protocols. Advantages include the following:

- VLSM support
- Rapid convergence thanks to Diffusing Update Algorithm (DUAL)
- Low CPU utilization—with typically only hellos and partial updates being sent on a link
- Incremental updates
- Scalability
- Ease of configuration
- Automatic route summarization, or manual route summarization
- MD5 route authentication

EIGRP uses IP protocol 88. It uses a multicast address of 224.0.0.10 for hellos and routing updates.

EIGRP's Metric

EIGRP uses a composite metric such as Interior Gateway Routing Protocol (IGRP), but it is modified with a multiplier of 256. Bandwidth and delay are the defaults enabled. EIGRP calls the metric feasible distance. All the possible metric values are as follows:

 Please see page 245 for more details.

- **Bandwidth:** Expressed in kilobytes; to adjust the bandwidth value assigned to an interface, use the **bandwidth** command.
- **Delay:** Expressed in microseconds; it can be adjusted using the **delay** command; when manipulating metrics, consider **delay** because bandwidth would affect other protocols, too.
- **Reliability:** Expressed as a number in the range of 1 to 255; 1 is a completely unreliable link.
- **Load:** Expressed as a number in the range of 1 to 255; 1 is a minimally loaded link.
- **MTU:** Maximum transmission unit; the smallest recorded MTU in the path. Note that MTU is not used in metric calculation.

The metric formula used by EIGRP is as follows:

metric = [K1 * BW + ((K2 * BW) / (256 – load)) + K3 * delay]

By default, K1 = 1, K2 = 0, K3 = 1, K4 = 0, K5 = 0.

If you manipulate the K values on one router, you must manipulate on all.

EIGRP uses a 32-bit metric as opposed to the 24-bit metric of IGRP; the two are compatible automatically during redistribution, however.

EIGRP Packets

- **Hello:** Establish neighbor relationships.
- **Update:** Send routing updates.
- **Query:** Ask neighbors about routing information.
- **Reply:** Respond to queries.
- **Ack:** Acknowledge reliable packets.

 Please see page 245 for more details.

The address used for hello packets is 224.0.0.10; AS numbers must match. Hellos are sent every 5 seconds on broadcast links and point-to-point serial links, point-to-point subinterface links, and multipoint circuits greater than T1. They are sent every 60 seconds on other link types. The hold time defaults to 3 times the hello time. Neighborships form even if the values do not match.

EIGRP Reliability

Packets that require acknowledgment are as follows:

- Update
- Query
- Reply

Packet that do not are as follows:

- Hello
- Ack

Neighbor reset after retry limit (16) is reached. Slow neighbors are sent unicast packets instead.

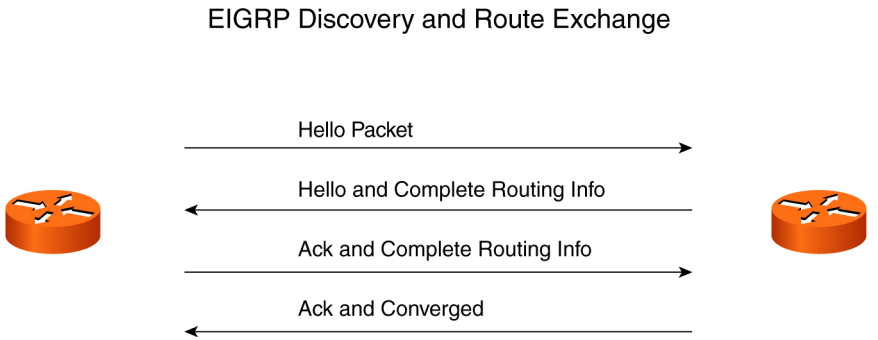
Initial Route Discovery

Router discovery and route exchange happen simultaneously as follows:

1. Router comes up and sends hellos.
2. Reply from a neighbor includes Update.
3. Ack packets are sent.
4. Update process occurs in the opposite direction.

 Please see page 245 for more details.

FIGURE 4-9
EIGRP Discovery and Route Exchange



EIGRP DUAL

The lowest-cost route is calculated by adding the cost between the next-hop router and the destination (advertised distance [AD]) and the cost between the local router and the next hop. This sum is referred to as the feasible distance (FD).

A successor is a neighboring router that the local router has selected to forward packets to the destination. Multiple successors can exist if they have equal-cost paths.

The next-hop router for a backup path is called the feasible successor. To qualify as a feasible successor, a next-hop router must have an AD less than the FD of the current successor route. More than one feasible successor can exist.

The feasible successor means that a new path can be selected without recalculation and is a major advantage in EIGRP for convergence.

Remember, EIGRP acts classful by default and automatically summarizes on major network boundaries. You typically want to disable this feature with the **no auto-summary router** configuration command.

EIGRP Queries

If EIGRP detects a change to the network topology, an input event, that requires a change to a route, it must perform a check to determine the existence of a Feasible Successor (FS). If an FS is not found, the Query process must be initiated, which is going Active on a route.

 Please see page 245 for more details.

When a route is active, an EIGRP router uses a multicast query to ask all its neighbors for a valid route to the subnet. Because a received query is considered an input event, EIGRP follows a similar process before responding. If the neighbor router receives a query for a subnet to which it does not have a route to, it sends a unicast reply stating that it has no route. If the neighbor router does have a route to the subnet, that route can be affected by the original Query. In this case, EIGRP goes Active on the route as well. If not, or if the router has an FS, it responds with a unicast EIGRP reply message with the route details. If the query causes the router to go active on the route, it does not immediately respond but instead generates a Query to all of its neighbors. If no router in the EIGRP domain contains a route to the subnet, the route is removed from all routing tables. Otherwise when an FS is found, it is propagated to all the querying routers.

A route is considered stuck-in-active if no response to the query has been received for a configured amount of time (default 3 minutes). After this time, the EIGRP drops all neighbors that it has not received replies from.

Configuring EIGRP

To enable EIGRP, use the following global configuration command:

```
router eigrp autonomous-system-number
```

To identify the interfaces participating in EIGRP, use the following router configuration command:

```
network network-number [wildcard-mask]
```

Using the default-network Command

Using the command, you can configure a default route for the EIGRP process so that it propagates to other EIGRP routers within the same AS. A router configured with the command considers the network listed in that command as the last-resort gateway. You should define the default route using a static route to ensure it is advertised.

Verification

A command that deserves some elaboration is the **show ip eigrp topology** command. The codes in the output are as follows:

 Please see page 245 for more details.

- **Passive:** This network is available, and installation can occur in the routing table.
- **Active:** This network is currently unavailable, and installation cannot occur in the routing table.
- **Update (U):** Applies if a network is updated (placed in an update packet); this code also applies if the router is waiting for an acknowledgment for this update packet.
- **Query (Q):** Applies if an outstanding query packet exists for this network other than in the active state; also applies if the router is waiting for an acknowledgment for a query packet.
- **Reply (R):** Applies if the router is generating a reply for this network or is waiting for an acknowledgment for the reply packet.
- **Stuck in active (SIA) status:** Signifies an EIGRP convergence problem for the network with which it is associated.

EIGRP Route Summarization

EIGRP performs auto-summarization by default. You can enable manual summarization. Keep the following in mind about manual summarization:

- Summarization is configurable on a per-interface basis in any router within a network.
- When summarization is configured on an interface, the router immediately creates a route pointing to null0. This is a loop-prevention mechanism.
- When the last specific route of the summary goes away, the summary is deleted.
- The minimum metric of the specific routes is used as the metric of the summary route.

To disable auto-summarization, use the **no auto-summary** command in EIGRP router configuration mode. Use the **ip summary-address eigrp** interface command to manually create a summary route at an arbitrary network boundary within an EIGRP domain.

 Please see page 245 for more details.

Unequal-Cost Load Balancing

The degree to which EIGRP performs load balancing is controlled with the **variance** command. You set the variance to a number from 1 to 128. The default is 1, which indicates equal-cost load balancing. The multiplier defines the range of metric values that are accepted for load balancing by the EIGRP process. For example, if you want load balancing to occur between two links, and one has a metric of 1000 and the other has a metric of 2000, you need to set the variance to 2 to cause load balancing between the two links. A route must be in the topology table by meeting the initial feasibility test; otherwise, it won't be used in load balancing regardless of the variance.

Bandwidth Utilization

By default, EIGRP uses up to 50 percent of the bandwidth of an interface or subinterface, which is set with the bandwidth parameter. This percentage can be changed on a per-interface basis by using the **ip bandwidth-percent eigrp nnn** interface configuration command. In this command, *nnn* is the percentage of the configured bandwidth that EIGRP can use. This percentage can be greater than 100. This is useful if the bandwidth is configured artificially low for routing policy reasons.

EIGRP Stub Routing

Often used in a hub-and-spoke topology. Only routes you specify are propagated from the stub router. The stub router responds to all queries with the message "inaccessible." A router configured as a stub sends a special peer information packet to all neighboring routers to report its status as a stub router. Nonstub routers do not query stub routers. The stub routing feature does not prevent routes from being advertised to the stub router. You must configure the summarization or default route behavior. To configure the stub router, use the following router configuration command:

```
eigrp stub [receive-only | connected | static | summary]
```

The optional keywords with this command control which routes the router advertises to its nonstub peers.

 Please see page 245 for more details.

Route Filtering and Policy Routing

Distribute lists

You can filter routing update traffic for any protocol by defining an access list and applying it to a specific routing protocol. You use the **distribute-list** command and link it to an access list to complete the filtering of routing update traffic.

For outbound traffic, the appropriate router configuration mode command is as follows:

```
distribute-list {access-list-number | name} out [interface-name | routing-process | [autonomous-system
number]]
```

For inbound traffic, the appropriate router configuration command is as follows:

```
distribute-list {access-list-number | name} in [type number]]
```

Using a distribute list with redistribution helps prevent route feedback. Route feedback occurs when routes originally learned from one routing protocol are redistributed back into that protocol. Route feedback can help lead to routing loops caused by redistribution.

Route Maps

Route maps are complex access lists that enable conditions to be tested against a packet or route using the **match** commands. If the conditions match, actions can be taken to modify attributes of the packet or route. These actions are specified by **set** commands.

Several of the more common applications for route maps are as follows:

- Route filtering during redistribution
- Policy-based routing (PBR)
- Network Address Translation (NAT)
- Implementing BGP policies

 Please see page 245 for more details.

To define the route map conditions and set the sequence number of route map lines, use the following global configuration mode commands:

```
route-map map-tag [permit | deny] [sequence-number]
```

To define the conditions to match, use the following command:

```
match {conditions}
```

To define the actions to be taken, use the following command:

```
set {actions}
```

Policy Routing

PBR enables you to implement policies that selectively cause packets to take different paths; this enables you to vary from the typical destination-based approach of IP. For example, you can easily configure routes to flow based on source address information. You can also mark traffic with different type of service (ToS) configurations. You implement PBR through the use of route maps to implement policy.

To identify a route map to use for PBR on an interface, use the following command:

```
ip policy route-map map-tag
```

PBR must be configured before PBR fast switching can be enabled. Fast switching of PBR is disabled by default. To configure fast-switched PBR, use the **ip route-cache policy** command in interface configuration mode.

Redistribution <Mods>

There are quite often occasions in which two or more routing protocols are used within a domain. Because these two routing protocols might contain information about different networks, if full connectivity is to be created, there must be a way to feed information from one protocol into another. Route Redistribution takes information from one protocol and inserts it into another protocol. A common example of this is found in most modest-sized networks when BGP is run as EGP with a service provider and an IGP is run internally to create full network connectivity inside the domain. BGP can advertise the internal network routes to the rest of the Internet and provide a path to the Internet. Because each routing protocol uses different methods for metric calculation, it is difficult to equate a

 Please see page 245 for more details.

metric in one protocol to a second protocol. Other problems arise, such as learning the same prefix from more than one routing protocol. A decision must be made as to which protocol is more trustworthy. It's also possible to create routing loops as routes are advertised from one protocol to another.

Although redistribution between certain protocols has unique concerns and characteristics, the following generic steps apply to all routing protocol combinations:

Step 1. Locate the boundary router that requires configuration of redistribution.

Step 2. Determine which routing protocol is the core or backbone protocol.

Step 3. Determine which routing protocol is the edge or short-term protocol.

Step 4. Select a method for injecting the required edge protocol routes into the core.

To redistribute routes into RIP:

```
redistribute protocol [process-id] [match route-type][metric metric-value] [route-map map-tag]
```

RIP requires a metric to be specified within the normal valid limits of a RIP route. This must be specified after the **metric** option. A route-map can also be optionally be applied that enables control over which routes will be redistributed into RIP. The metric type can also be matched that enables a specific type of route to be the only routes redistributed.

To redistribute routes into OSPF:

```
redistribute protocol [process-id] [metric metric-value][metric-type type-value] [route-map map-tag]
[subnets] [tag tag-value]
```

OSPF also requires a metric when importing routes. The subnets option is also almost always used when redistributing routes into OSPF. Otherwise only networks that are not subnetted will be added. You can use Route maps to specify with granularity specific subnets to be added. Use the tag option to add a tag to the routes. This helps when determining where a route came from. This is often helpful when trying to stop routing loops due to redistribution.

 Please see page 245 for more details.

To redistribute routes into EIGRP:

```
redistribute protocol [process-id] [match {internal | external 1 | external 2}] [metric metric-value]
[route-map map-tag]
```

EIGRP mirrors OSPF and RIP with some of the same options but additionally adds the match option. This enables only specific route types from OSPF to be inserted into the EIGRP routing process. The metric also must be specified using the EIGRP metrics of bandwidth, delay, load, reliability, and MTU.

Route Tagging

Various routing protocols support tag fields. This tag field provides a location where additional information about a route can be stored. This field is commonly used to identify the AS from which a route was obtained when a route is learned from a different AS. Route tagging enables you to customize routing and maintain flexible policy controls.

 Please see page 245 for more details.

Chapter 5

Quality of Service (QoS)

Introduction

Voice, video, and data travel side by side over today's converged networks. Some of these traffic types (for example, VoIP) need better treatment (that is, higher priority) than other types of traffic (for example, FTP). Fortunately, Cisco offers a suite of QoS tools for providing special treatment for special traffic.

In the absence of QoS, traffic might suffer from one or more of the following symptoms:

- **Delay (latency):** Excessive time required for a packet to traverse the network
- **Delay variation (jitter):** The uneven arrival of packets, which in the case of VoIP can be interpreted by the listener as dropped voice packets
- **Packet loss:** Dropping packets, especially problematic for User Datagram Protocol (UDP) traffic (for example, VoIP), which does not retransmit dropped packets

You have two categories of QoS tools: Integrated Services (IntServ) and Differentiated Services (DiffServ). IntServ provides QoS by guaranteeing treatment to a particular traffic flow. A commonly used IntServ tool is RSVP (Resource Reservation Protocol).

As the name suggests, DiffServ differentiates (that is, classifies) between different types of traffic and provides different levels of service based on those distinctions. Instead of forcing every network device to classify traffic, DiffServ can mark packets with a particular priority marking that can be referenced by other network devices.

 Please see page 245 for more details.

ToS and IP Precedence

Packet marking can be accomplished by altering bits in an IPv4 header's ToS byte. Two common markings that use the ToS byte are IP Precedence and Differentiated Services Code Point (DSCP).

IP Precedence is an older approach than DSCP and uses the 3 left-most bits in the ToS byte. With 3 bits to use, IP Precedence values can range from 0 to 7. Cisco recommends that IP Precedence values 6 and 7 never be used because they are reserved for network use.

Cisco IOS Software accepts either an IP Precedence number or its equivalent name, as shown in Table 5-1.

Table 5-1 IP Precedence Numbers

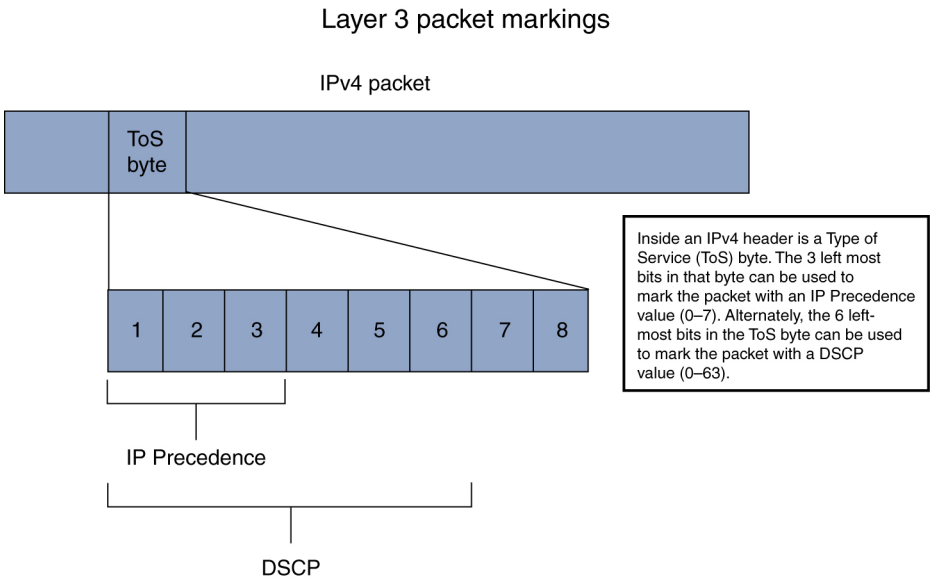
IP Precedence Value	Name
0	Routine
1	Priority
2	Immediate
3	Flash
4	Flash-override
5	Critical
6	Internet
7	Network

Differentiated Services Code Point

Differentiated Services Code Point (DSCP) uses the 6 left-most bits in an IPv4 header's ToS byte. With 6 bits at its disposal, DSCP has up to 64 DSCP values (0 to 63) assigned to various classes of traffic. With so many values to select from, to maintain relative levels of priority among routers, the IETF recommends selected DSCP values for use. These values, called Per-Hop Behaviors (PHB), determine how packets are treated at each hop along the path from the source to the destination.

 Please see page 245 for more details.

FIGURE 5-1
Layer 3 Packet Markings



When configuring a router to mark or recognize a DSCP value, the decimal number can be used. However, a more convenient method is to use the name of specific DSCP values. Assured Forwarding (AF) PHBs are typically used to identify different levels of priority for data applications. For latency-sensitive applications, however, the Expedited Forwarded (EF) PHB can be used. A listing of commonly used PHB names and their corresponding DSCP values is shown in Table 5-2.

Table 5-2 PHB Names and DSCP Values

PHB	Low Drop Preference	Medium Drop Preference	High Drop Preference
Class 1	AF11 (10)	AF12 (12)	AF13 (14)
Class 2	AF21 (18)	AF22 (20)	AF23 (22)
Class 3	AF31 (26)	AF32 (28)	AF33 (30)
Class 4	AF41 (34) EF (46)	AF42 (36)	AF43 (38)

 Please see page 245 for more details.

Notice that the AF PHBs are grouped into four classes. Examining these DSCP values in binary reveals that the 3 left-most bits of all the Class 1 AF PHBs are 001 (that is, a decimal value of 1); the 3 left-most bits of all the Class 2 AF PHBs are 010 (that is, a decimal value of 2); the 3 left-most bits of all the Class 3 AF PHBs are 011 (that is, a decimal value of 3); and the 3 left-most bits of all the Class 4 AF PHBs are 100 (that is, a decimal value of 4). Because IP Precedence examines these 3 left-most bits, all Class 1 DSCP values would be interpreted by an IP Precedence–aware router as an IP Precedence value of 1. The same applies to Class 2, 3, and 4 PHB values.

In a similar fashion, the 3 left-most bits of the EF PHB are 101 (that is, a decimal value of 5). Therefore, the EF PHB would be interpreted by an IP Precedence–aware router as an IP Precedence of 5, the highest IP Precedence value that we should assign. Because of these associations that exist between DSCP markings and IP Precedence, DSCP is backward compatible with IP Precedence.

Class of Service

Although an IP header's ToS byte can be used for Layer 3 markings, a class of service (CoS) marking can be used for Layer 2 markings. Specifically, CoS markings are applied to frames crossing an IEEE 802.1Q or an Inter-Switch Link (ISL) trunk. Regardless of the trunk type, CoS markings use 3 bits. So, like IP Precedence, CoS values range from 0 through 7, and again, values 6 and 7 are reserved.

Network-Based Application Recognition

Cisco offers multiple approaches to identify packets to mark. For example, packets can be classified and marked if they match a particular access list or if they come into a router on a particular interface. However, one of the most powerful Cisco IOS tools for performing packet classification is Network-Based Application Recognition (NBAR), which can look beyond Layer 4 information, all the way up to the application layer, where NBAR can recognize such packet attributes as character strings in a URL.

NBAR is accomplished using the MQC, the Modular quality of service(QoS) command-line interface (CLI). This tool is shown later in this section. NBAR is used in a class map to identify traffic. The **match protocol** keywords are used to trigger NBAR, as follows:

 Please see page 245 for more details.

```
class-map IDENTIFY_HTTP
match protocol http
```

Queuing Techniques

Marking a packet does not change its operation, unless QoS tools are enabled that can reference that marking. Fortunately, multiple QoS tools can make forwarding or dropping decisions based on these markings. Queuing techniques are often referred to as congestion management tools.

Queuing tools decide how packets are emptied from an interface's output queue. Several queuing tools are available in the Cisco IOS Software:

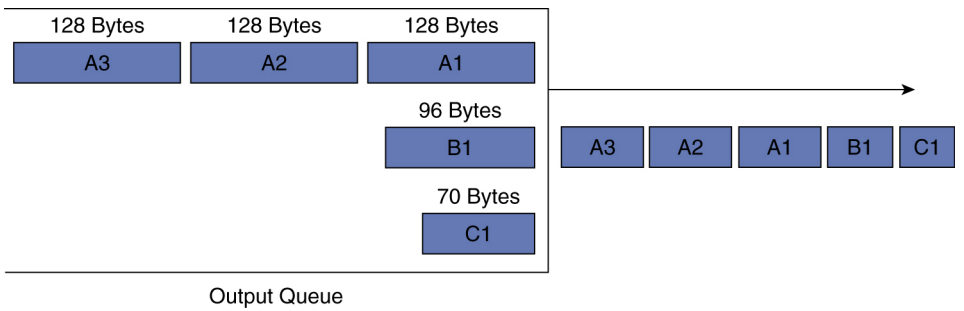
- **First-In, First-Out (FIFO):** The default queuing mechanism on high-speed interfaces (that is, greater than 2.048 Mbps), which does not reorder packets
- **Weighted Fair Queuing (WFQ):** The default queuing mechanism on low-speed interfaces, which makes forwarding decisions based on a packet's size and Layer 3 priority marking
- **Low latency queuing (LLQ):** The preferred queuing method for voice and video traffic, in which traffic can be classified in up to 64 different classes, with different amounts of bandwidth given to each class; includes the capability to give priority treatment to one or more classes
- **Priority queuing:** A legacy queuing approach with four queues, in which higher-priority queues must be emptied before forwarding traffic from any lower-priority queues
- **Custom queuing:** A legacy queuing approach that services up to 16 queues in a round-robin fashion, emptying a specified **number of bytes from each queue during each round-robin cycle**
- **Class-based weighted fair queuing (CBWFQ):** Similar to LLQ, with the exception of having no priority queuing mechanism
- **IP RTP priority:** A legacy queuing approach for voice traffic that placed a range of UDP ports in a priority queue, with all other packets treated with WFQ

 Please see page 245 for more details.

Weighted fair queuing (WFQ) is enabled by default on slow-speed interfaces (that is, 2.048 Mbps and slower). WFQ allocates a queue for each flow, for as many as 256 flows by default. WFQ uses IP Precedence values to provide a weighting to fair queuing (FQ). When emptying the queues, FQ, sometimes called "flow-based queuing," does "byte-by-byte" scheduling. Specifically, FQ looks 1 byte deep into each queue to determine whether an entire packet can be sent. FQ then looks another byte deep into the queue to determine whether an entire packet can be sent. As a result, smaller traffic flows and smaller packet sizes have priority over bandwidth-hungry flows with large packets.

In the following example, three flows simultaneously arrive at a queue. Flow A has three packets, which are 128 bytes each. Flow B has a single 96-byte packet. Flow C has a single 70-byte packet. After 70 byte-by-byte rounds, FQ can transmit the packet from flow C. After an additional 26 rounds, FQ can transmit the packet from flow B. After an additional 32 rounds, FQ can transmit the first packet from flow A. Another 128 rounds are required to send the second packet from flow A. Finally, after a grand total of 384 rounds, the third packet from flow A is transmitted.

FIGURE 5-2
Fair Queuing



With WFQ, a packet's IP Precedence influences the order in which it is emptied from a queue. Consider the previous scenario with the addition of IP Precedence markings. In this scenario, flow A's packets are marked with an IP Precedence of 5, whereas flow B and flow C have default IP Precedence markings of 0. The order of packet servicing with WFQ is based on sequence numbers, in which packets with the lowest sequence numbers are emptied first.

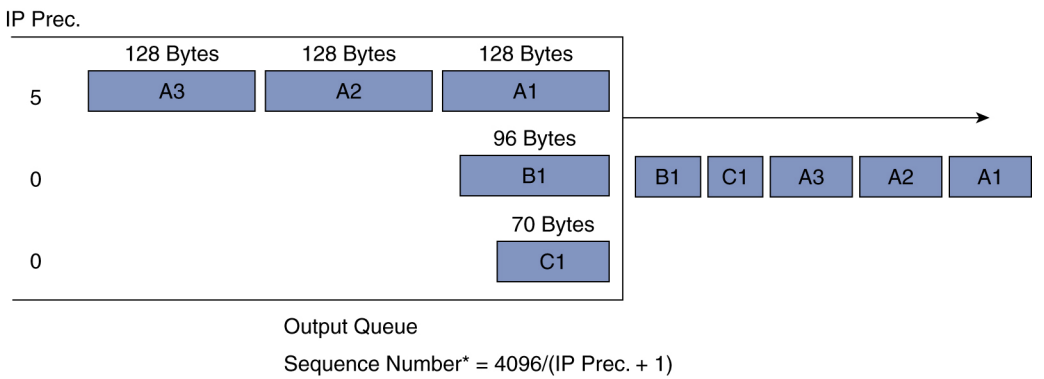
The sequence number is the "weight" of the packet multiplied by the number of byte-by-byte rounds that must be completed to service the packet (that is, just as in the FQ example). The Cisco IOS Software calculates a packet's weight differently depending on the Cisco IOS version. Before Cisco IOS Release 12.0(5)T, the formula for weight was WEIGHT = 4096 / (IP Prec. + 1).

 Please see page 245 for more details.

In more recent versions of the Cisco IOS Software, the formula for weight is WEIGHT = 32768 / (IP Prec. + 1). Using the pre-Cisco IOS Release 12.0(5)T formula, the sequence numbers are as follows:

A1 = 4096 / (5 + 1) * 128 = 87,381
A2 = 4096 / (5 + 1) * 128 + 87,381 = 174,762
A3 = 4096 / (5 + 1) * 128 + 17,4762 = 262,144
B1 = 4096 / (0 + 1) * 96 = 393,216
C1 = 4096 / (0 + 1) * 70 = 286,720

FIGURE 5-3
Weighted Fair Queuing



Therefore, after the weighting is applied, WFQ empties packets from the queue in the following order: A1—A2—A3—C1—B1. With only FQ, packets are emptied from the queue in the following order: C1—B1—A1—A2—A3.

Custom queuing (CQ) enhances some of the characteristics of WFQ by enabling the administrator to specify which traffic goes into a particular queue. Also, a weight can be assigned to each of the queues, which specifies how many bytes are emptied from a queue during each round-robin servicing of the queues. Consider the following custom queuing example:

```
Router(config)# queue-list 1 protocol ip 1 tcp www
Router(config)# queue-list 1 protocol ip 2 tcp telnet
Router(config)# queue-list 1 default 3
Router(config)# queue-list 1 queue 1 byte-count 1500 limit 512
```

 Please see page 245 for more details.

```
Router(config)# queue-list 1 queue 2 byte-count 1500 limit 512
Router(config)# queue-list 1 queue 3 byte-count 3000 limit 512
!
Router(config)# interface serial 0/1
Router(config-if)# bandwidth 128
Router(config-if)# custom-queue-list 1
```

In the preceding example, a queue list (numbered 1) is defined. The queue list specifies that World Wide Web traffic goes in queue 1. Telnet traffic goes in queue 2, and other traffic (that is, default traffic) goes in queue 3. CQ services these queues in a round-robin fashion. As CQ empties the queues, the number of bytes emptied from each queue is influenced with the **byte-count** option, as shown in the example. The number of packets that can be placed in a particular queue can also be specified with the **limit** option. In the preceding example, each queue can accommodate 512 packets. Finally, the queue list is applied to interface serial 0/1.

In the preceding example, 1500 bytes are emptied from queue 1 and from queue 2 during each round-robin cycle, and 3000 bytes are emptied from queue 3 during each round-robin cycle. Therefore, a bandwidth percentage for each traffic type can be calculated as follows:

Total number of bytes serviced during each round-robin cycle = 1500 + 1500 + 3000 = 6000
Percentage of bandwidth for World Wide Web traffic = 1500 / 6000 = .25 = 25 percent
Percentage of bandwidth for Telnet traffic = 1500 / 6000 = .25 = 25 percent
Percentage of bandwidth for default traffic = 3000 / 6000 = .5 = 50 percent

CQ does, however, have a deficit issue. Specifically, when CQ empties bytes from a queue, it cannot send a partial packet. Consider a situation in which two packets are in queue 1: a 1499-byte packet and a 1500-byte packet. Queue 1 is configured to forward 1500 bytes per round. After the 1499-byte packet is transmitted, the 1500-byte level has not yet been reached. CQ therefore sends the following packet. Because CQ cannot send a partial packet, it sends the entire 1500-byte packet. As a result, even though queue 1 was configured to send only 1500 bytes per round, in this example, 2999 bytes were forwarded.

 Please see page 245 for more details.

On the Cisco 12000 series of routers, this deficit issue is overcome with MDRR (Modified Deficit Round Robin). MDRR keeps track of the extra bytes sent and adjusts how many bytes can be sent in subsequent rounds. MDRR can operate in either of two modes:

- **Strict priority:** Defines a priority queue that must be completely empty before any other traffic is sent.
- **Alternate priority:** Is a low-latency queue that alternates with each of the other queues so that traffic is not starved out. For example, consider queues 1, 2, and 3, where queue 1 is a low-latency queue. With alternate priority mode, the queues would be serviced as follows: 1, 2, 1, 3, 1.

Also, with DRR queuing, the number of bytes transmitted in one round is defined as maximum transmission unit (MTU) + (weight – 1) * 512. This number of bytes is transmitted from a queue or until the queue is empty. If more than this number of bytes is sent, to finish servicing a packet that had already started to be serviced, the DRR remembers this deficit, and in the next round, the deficit is subtracted from the number of bytes to service from the queue.

Priority queuing (PQ) can give strict priority to latency-sensitive applications (for example, e-commerce applications). PQ gives priority to specific packets by placing those packets in a high-priority queue. Other packets are placed in a medium, normal, or low queue. However, if any packets are in the high queue, none of the packets in lower-priority queues are sent. Similarly, when packets are in the medium queue, no packets are sent from the normal or low queues. Although this approach does accomplish the goal of giving priority to specific traffic, it can lead to protocol starvation. Consider the following PQ example:

```
Router(config)# priority-list 1 protocol ip high tcp www
Router(config)# priority-list 1 protocol ip medium tcp telnet
Router(config)# priority-list 1 default low
!
Router(config)# interface serial 0/1
Router(config-if)# priority-group 1
```

In the preceding example, a priority list (numbered 1) is created. The priority list specifies that World Wide Web traffic goes in

 Please see page 245 for more details.

the high queue. Telnet traffic goes in the medium queue, and all other traffic (that is, default traffic) goes in the low queue.

The priority-list is then applied to interface Serial 0/1. The potential for protocol starvation exists, because if at any time you have World Wide Web packets in the high queue, none of the packets from lower priority queues are forwarded until all of the World Wide Web packets have been forwarded.

IP Real-time Transport Protocol (RTP) priority combines some of the best aspects of PQ and WFQ. Specifically, IP RTP priority enables a range of UDP ports to be placed in a priority queue, whereas all other packets are treated with WFQ. Therefore, VoIP packets, which use UDP ports, can be assigned to the priority queue. Fortunately, to prevent protocol starvation, a bandwidth limit is set for the priority queue. IP RTP priority is configured using the following interface configuration mode command:

```
Router(config-if)# ip rtp priority starting-udp-port
port-number-range bandwidth
```

The *port-number-range* is not the last port number in the range. Rather, it is the number of ports in the range. For example, the following command specifies that 64 kbps of bandwidth should be made available for packets using UDP ports in the range 16,384 through 32,767:

```
Router(config-if)# ip rtp priority 16384 16383 64
```

The sum of the *starting-udp-port* and the *port-number-range* equals the last UDP port number in the range (that is, 16,384 + 16,383 = 32,767). The main drawback of IP RTP priority is its inability to place TCP ports in the priority queue. For example, H.323 call setup uses TCP ports. These call setup packets, however, cannot be placed in a priority queue using IP RTP priority.

CBWFQ and LLQ

With modern versions of the Cisco IOS Software, Cisco recommends CBWFQ or LLQ approaches to queuing. Both methods are configured using MQC.

The first step of MQC is to create class maps, which categorize traffic types. The following command enters class map configu-

 Please see page 245 for more details.

ration mode:

```
Router(config)# class-map [match-any ¦ match-all] class name
```

When in class map configuration mode, multiple **match** statements can be used to match traffic, and all traffic meeting the criteria specified by the **match** command is categorized under the class map. If multiple **match** statements are specified, by default all **match** statements must be met before a packet is classified by the class map. However, by using the **match-any** option, if any individual match condition is met, the packet is classified by the class map.

After the class maps are defined, the first step of MQC is complete. The second step is to create a policy map to assign characteristics (for example, marking) to the classified traffic.

To enter policy map configuration mode, issue the following command:

```
Router(config)# policy-map policy name
```

From policy map configuration mode, enter policy-map-class configuration mode with this command:

```
Router(config-pmap)# class class name
```

From policy-map-class configuration mode, QoS policies can be assigned to traffic classified by the class map. Finally, in the third step, the policy map is applied to an interface, Frame Relay map class, or ATM virtual circuit with this command:

```
Router(config-if)# service-policy {input ¦ output} policy map name
```

Here is an LLQ example that illustrates the MQC approach:

```
Router(config)# class-map SURFING
Router(config-cmap)# match protocol http
Router(config-cmap)# exit
Router(config)# class-map VOICE
Router(config-cmap)# match protocol rtp
Router(config-cmap)# exit
```

 Please see page 245 for more details.

```
Router(config)# policy-map CCIESTUDY
Router(config-pmap)# class SURFING
Router(config-pmap-c) # bandwidth 128
Router(config-pmap-c) # exit
Router(config-pmap)# class VOICE
Router(config-pmap-c)# priority 256
Router(config-pmap-c)# exit
Router(config-pmap)# exit
Router(config)# interface serial 0/1
Router(config-if)# service-policy output CCIESTUDY
```

In the preceding example, NBAR is used to recognize HTTP traffic, and that traffic is placed in the SURFING class. NBAR is invoked with the **Router(config-cmap)# match protocol** command. Voice packets are placed in the VOICE class. The CCIESTUDY policy map gives 128 kbps of bandwidth to the HTTP traffic while giving 256 kbps of priority bandwidth to voice traffic. The policy map is then applied outbound to interface serial 0/1.

Weighted RED

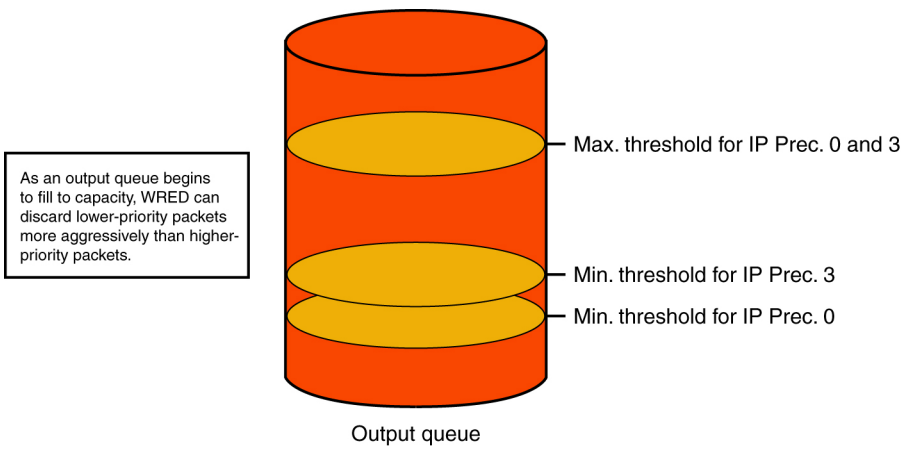
The purpose of Weighted Random Early Detection (WRED) is to prevent an interface's output queue from filling to capacity, because if a queue is completely full, all newly arriving packets are discarded. Some of those packets might be high priority, and some might be low priority. However, if the queue is full, no room exists for any packet. WRED is referred to as a congestion-avoidance QoS tool. It can also prevent a global synchronization problem, in which all TCP senders back off as packets at a full queue are dropped, and then all senders begin to increase the amount of traffic sent, until another synchronized back-off is triggered. Global synchronization results in poor utilization of interface bandwidth.

With a congestion-avoidance tool, drop thresholds are defined for various markings (for example, DSCP markings). Therefore, as a queue begins to fill, lower-priority packets are dropped more aggressively than higher-priority packets, thus preventing the queue from ever filling to capacity. The Cisco congestion-avoidance tool of choice is WRED.

 Please see page 245 for more details.

WRED can be configured in interface configuration mode. However, an MQC approach is also supported. Three parameters that can be configured for each IP Precedence value or DSCP value include the minimum threshold, maximum threshold, and mark probability denominator. The minimum threshold specifies the number of packets in a queue before the queue considers discarding packets having a particular marking. The probability of discard increases until the queue depth reaches the maximum threshold. After a queue depth exceeds the maximum threshold, all other packets with a particular marking that attempt to enter the queue are discarded. However, the probability of packet discard when the queue depth equals the maximum threshold is 1 / (mark probability denominator). For example, if the mark probability denominator were set to 10, when the queue depth reached the maximum threshold, the probability of discard for the specified marking would be 1 / 10 (that is, a 10 percent chance of discard).

FIGURE 5-4
Weighted Random Early Detection (WRED)



When configuring WRED, the Cisco IOS Software automatically assigns default values to these parameters. However, these parameters can be altered, and the marking WRED pays attention to (that is, IP Precedence or DSCP) can be specified. Following is the syntax to enable WRED in interface configuration mode:

```
random-detect [dscp-based ¦ prec-based]
```

If neither **dscp-based** nor **prec-based** is specified, WRED defaults to **prec-based**. Following is the syntax to specify WRED parameters for both IP Precedence values and DSCP values:

 Please see page 245 for more details.

```
random-detect precedence
random-detect dscp
```

To specify WRED parameters for a specific class of traffic, using the MQC approach, the exact commands just shown can be entered in policy-map-class configuration mode.

To reinforce this syntax, consider the following example, where the goal is to configure WRED on interface Ethernet 0/0. After the output queue depth reaches 25 packets, the possibility is introduced that a DSCP value of AF13 be discarded. Packets marked with a DSCP value of AF12 should not be discarded until the queue depth reaches 30 packets. Finally, packets marked with a DSCP value of AF11 should not have any chance of discard until the queue depth reaches 35 packets. If the queue depth exceeds 100 packets, there should be a 100 percent chance of discard for these three DSCP values. However, when the queue depth is exactly 100 packets, the percent chance of discard for these various packet types should be 25 percent:

```
Router(config)# interface ethernet 0/0
Router(config-if)# random-detect dscp-based
Router(config-if)# random-detect dscp af13 25 100 4
Router(config-if)# random-detect dscp af12 30 100 4
Router(config-if)# random-detect dscp af11 35 100 4
```

Examine the solution; the mark probability denominator is 4. This value was chosen to meet the requirement that there be a 25 percent chance of discard when the queue depth equals the maximum threshold (that is, 1 / 4 = .25). Also, a DSCP value of AF13 is dropped before a DSCP value of AF12, which is dropped before a DSCP value of AF11. This approach is consistent with the definition of these PHBs, because the last digit in the AF DSCP name indicates its drop preference. For example, a value of AF13 would drop before a value of AF12.

WRR/Queue Scheduling

Some Cisco Catalyst switches also support their own queuing method, called weighted round robin (WRR). For example, a Catalyst 2950 switch has four queues, and WRR can be configured to place frames with specific CoS markings in certain queues. (For example, CoS values 0 and 1 are placed in queue 1.)

 Please see page 245 for more details.

Weights can be assigned to the queues, influencing how much bandwidth the various markings receive. The queues are then serviced in a round-robin fashion. On some platforms, one of the switch's queues can be designated as an expedite queue, which gives priority treatment to frames in that queue. Specifically, the expedite queue must be empty before any additional queues are serviced. This behavior can lead to protocol starvation.

The following is an example of a WRR configuration:

```
Switch(config)# interface gig 0/5
Switch(config-if)# wrr-queue bandwidth 1 2 3 4
Switch(config-if)# wrr-queue cos-map 4 5
```

In the preceding example, the **wrr-queue** command assigns the weights 1, 2, 3, and 4 to the switch's four queues. The first queue, with a weight of 1, gets only one-third the bandwidth given to the third queue, which has a weight of 3. The **wrr-queue cos-map** command instructs frames marked with a CoS of 5 to enter the fourth queue.

Shaping Versus Policing

Although some of the congestion-management techniques can guarantee bandwidth amounts, you might want to limit bandwidth usage in some situations. For example, you might need to prevent oversubscription of a link. Two categories of traffic conditioning exist:

- **Policing:** Limits traffic rates, with excess traffic being dropped
- **Shaping:** Limits traffic rates, with excess traffic being delayed (that is, buffered)

As shown in the preceding description, shaping buffers excess traffic, whereas policing drops excess traffic. These characteristics suggest that policing is more appropriate on high-speed interfaces, whereas shaping is more appropriate on low-speed interfaces.

For policing and shaping tools to limit bandwidth, they don't transmit all the time. Specifically, they send a certain number of bits or bytes at line rate, and then they stop sending until a specific timing interval (for example, one-eighth of a second) is reached. When the timing interval is reached, the interface again sends a specific amount of traffic at line rate; it stops; and it waits for the next timing interval. This process repeats over and over, enabling an interface to send an average bandwidth that might be below the physical speed of the interface.

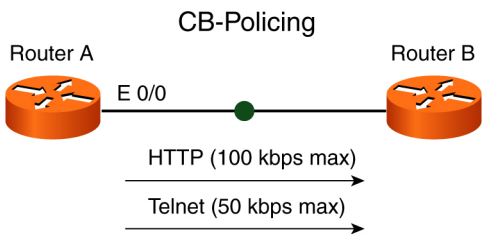
 Please see page 245 for more details.

Both policing and shaping configurations can specify a committed information rate (CIR), committed burst (Bc), and excess burst (Be). The CIR is the average number of bits sent during 1 second. The Bc indicates how many bits or bytes can be sent at line rate during a timing interval. The Be enables more than Bc bits or bytes to be sent during a timing interval if some bits or bytes were unused during a previous timing interval.

Although policing and shaping can be configured using the MQC method previously described, legacy methods include committed access rate (CAR) for policing, Generic Traffic Shaping (GTS) for shaping, and Frame Relay Traffic Shaping (FRTS) for shaping. GTS can be applied to an interface or a subinterface, and FRTS can be applied to an interface, subinterface, or Frame Relay data-link connection identifier (DLCI). Other queuing methods, such as PQ, CQ, or WFQ, can be applied to traffic after GTS shapes it. However, GTS uses WFQ in its shaping queue.

A modern approach to policing is Class-Based Policing (CB-Policing), which uses the previously described MQC process. The goal of the following CB-Policing example is to limit outgoing web traffic to 100 kbps and Telnet traffic to 50 kbps on interface ethernet 0/0.

FIGURE 5-5
CB-Policing




```
RouterA(config)# class-map WEB
RouterA(config-cmap)# match protocol http
RouterA(config-cmap)# exit
RouterA(config)# class-map TELNET
RouterA(config-cmap)# match protocol telnet
RouterA(config-cmap)# exit
RouterA(config)# policy-map POLICING_EXAMPLE
RouterA(config-pmap)# class WEB
RouterA(config-pmap-c)# police 100000
```

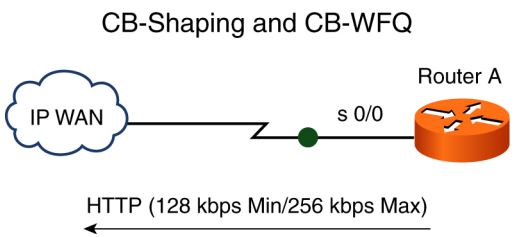
 Please see page 245 for more details.

```
RouterA(config-pmap-c)# exit
RouterA(config-pmap)# class-map TELNET
RouterA(config-pmap-c)# police 50000
RouterA(config-pmap-c)# exit
RouterA(config-pmap-c)# exit
RouterA(config-pmap)# exit
RouterA(config)# interface ethernet 0/0
RouterA(config-if)# service-policy output POLICING_EXAMPLE
```

Shaping can also be configured using this MQC approach. When configuring CB-Shaping, traffic can be shaped to either average or peak. If **shape average** is specified, traffic is sent at the CIR, with bursting of Be bits per timing interval enabled. If **shape peak** is specified, the router attempts to forward traffic at the peak rate: Peak Rate = CIR * (1 + Be/Bc). The shaping to peak method can result in occasional packet loss, requiring retransmission.

In Figure 5-6, CBWFQ is combined with CB-Shaping to specify that HTTP traffic can have at least 128 kbps but no more than 256 kbps as the packets exit the serial 0/0 interface. Note that the units of measure for the CIR are in bits per second.

FIGURE 5-6
CB-Shaping



CB-Shaping and CB-WFQ

```
RouterA(config)# class-map HTTP
RouterA(config-cmap)# match protocol http
RouterA(config-cmap)# exit
RouterA(config)# policy-map WEB
RouterA(config-pmap)# class HTTP
RouterA(config-pmap-c)# shape average 256000
```

 Please see page 245 for more details.

```
RouterA(config-pmap-c)# bandwidth 128
RouterA(config-pmap-c)# exit
RouterA(config-pmap)# exit
RouterA(config)# interface serial 0/0
RouterA(config-if)# service-policy output WEB
```

Modified Deficit Round Robin (MDRR)

Modified Deficit Round Robin (MDRR) is a QoS mechanism implemented on the Cisco 12000 series routers. It defines the manner in which traffic is removed from queues based upon two factors:

- **Quantum Value:** The average number of bytes which can be sent at each round
- **Deficit Value:** The number of bytes services during a round; initialized to the quantum value at the start of the round

Each round, packets are serviced from each queue until the number of bytes serviced, the deficit value, is at or below zero. At that point, the next queue is serviced. MDRR also enables for a priority queue that can be run in two modes:

- **Strict priority mode:** The queue is always serviced when packets are in the queue. This can lead to starvation of other queues.
- **Alternate priority mode:** The priority queue is serviced every other round. That is, a nonpriority queue is serviced, and then the priority queue is served.

MDDR can be configured using the MQC by creating a **class-map** and **priority-map** and then applying it to an interface.

To create the *class-map* that uses the IP precedence value

```
Router(config)# class-map <CLASS MAP NAME>
Router(config-cmap)# match ip precedence <PRECEDENCE VALUE>
```

Next, create a policy map that defines the bandwidth and whether the priority queue is used:

 Please see page 245 for more details.

```
Router(config)# policy-map <POLICY MAP NAME>
Router(config-pmap)# class <CLASS MAP NAME>
Router(config-pmap)# bandwidth percent <% bandwidth>
```

or

```
Router(config-pmap)# bandwidth <kbps bandwidth>
```

If this class-map defines the priority queue traffic

```
Router(config-pmap)# priority
```

Last, the policy-map is applied to an interface:

```
Router(config-if)# service-policy output <POLICY MAP NAME>
```

Generic Traffic Shaping

Traffic shaping is the mechanism within Cisco IOS that regulates the flow of data out an interface. Following are three general types of traffic shaping available:

- Frame Relay Traffic Shaping
- Class-based Traffic Shaping
- Generic Traffic Shaping

Generic Traffic Shaping is an older syntax that has been phased out with new IOS version, but it's important to know the syntax of Generic Traffic Shaping because there is legacy equipment floating around. Generic Traffic Shaping uses ACLs to match interesting traffic that will be shaped and is applied on a per-interface basis.

To shape all the traffic leaving an interface, an ACL is not used. The **traffic-shape rate** command specifies the bit rate, and optionally the burst and excess burst rates:

```
Router(config-if)# traffic-shape rate bit-rate [burst-size] [excess-burst-size]
```

 Please see page 245 for more details.

When an ACL is created to match traffic, a numbered ACL must be used. After the creation of the ACL, the **traffic-shape group** command shapes the traffic leaving the interface:

```
Router(config-if)# traffic-shape group <ACL> bit-rate [burst-size [excess-burst-size]]
```

If traffic shaping is performed on a frame-relay interface, the **traffic-shape adaptive** command can change the shaping parameters when a BECN is received. The bit-rate specified is used when the BECN is received:

```
Router(config-if)# traffic-shape adaptive <bit rate>
```

Shaping can also reflect received frames with the FECN bit set with the BECN bit set:

```
Router(config-if)# traffic-shape fecn-adapt
```

You need to view the status of the traffic shaping configuration and statistics on an interface. To view the traffic-shaping configuration on an interface

```
Router# show traffic-shape <INT>
```

To view the traffic shaping statistics on an interface

```
Router# show traffic-shape statistics <INT>
```

Resource Reservation Protocol (RSVP)

Resource Reservation Protocol (RSVP) is a change to typical QoS deployments in that it does not rely on Per-Hop Behaviors (PHB), which are the building blocks of the DiffServ model of QoS. Another implementation of QoS is the IntServ model that uses RSVP to enable applications that require specific bandwidth requirements to allocate that bandwidth through the network end to end.

RSVP works by enabling hosts to send a special PATH packet toward the intended destination that signals to all the routers along the way that the sender is trying to allocate bandwidth for a particular application. The receiver of the packet then sends a reservation (RESV) packet back to the sender that tells the router to allocate the resources. The router can then allocate the resources if they are available or deny the request.

 Please see page 245 for more details.

By default, RSVP functionality is turned off. Enabling it on an interface is simple:

```
Router(config-if)# ip rsvp bandwidth <kbps>
```

This tells the router to enable RSVP and allocate a specific number of kbps to resource allocation. You can add the option of a single flow reservation limit to the command:

```
Router(config-if)# ip rsvp bandwidth <kbps> <single flow kbps>
```

For troubleshooting and diagnosis purposes, commands are available to simulate the resource allocation process:

```
Router(config)# ip rsvp sender session-ip-address sender-ip-address [tcp ¦ udp ¦ ip-protocol] session-
dport sender-sport previous-hop-ip-address previous-hop-interface bandwidth burst-size
```

which simulates an RSVP host generating the PATH message through the network. Receivers sending a RESV message can be simulated using the

```
ip rsvp reservation session-ip-address sender-ip-address [tcp ¦ udp ¦ ip-protocol] session-dport sender-
sport next-hop-ip-address next-hop-interface {ff ¦ se ¦ wf} {rate ¦ load} bandwidth burst-size
```

Performance Routing

Performance Routing (PfR) is an idea to expand upon the existing Optimized Edge Routing (OER) feature set within IOS. Currently, the two terms can be used relatively interchangeably.

In a network in which there are multiple exit points, OER can be configured to intelligently choose the exit point that maximizes performance based on metrics, such as packet loss, response time, load distribution, or cost minimization. These metrics are not available in standard BGP path selection. OER uses two main architectural components: the Master Controller (MC) and the Border Router (BR). The MC controls all OER functions within a domain by providing the monitoring of the data flows and changes to flow policies through control of the border routers. The BR is the device on the edge of the network, through which the data flows. The BR reports on the link measurements to the MC and implements the policy changes that affect the traffic flow. The MC can also live on a BR.

 Please see page 245 for more details.

Cisco OER can be characterized in five phases: Profile, Measure, Apply Policy, Control, and Verify. The Profile phase is the selec-tion of a subset class of traffic through either configuration or automated means. The Measure phase uses either active or passive monitoring to determine the metrics associated with each traffic flow. During the Apply Policy phase, the performance metrics gathered in the Measure phase are compared to determine if the traffic is Out-of-policy (OOP). The Control phase enhances the network by controlling the traffic flow through the BRs. The Verify phase is used to determine if the traffic is still OOP or has been corrected.

Cisco Optimized Edge Routing

The implementation of OER and PfR follows the two components previously described. The majority of the configuration for OER exists on the master controller. Typically a key-chain is set up on the master controller and BRs to ensure authentication occurs be-tween the two. This prevents the BRs from being controlled by a rouge MC:

```
Router(config)# key chain <NAME>
Router(config-keychain)# key <#>
Router(config-keychain-key)# key-string <KEY>
```

Now the MC can be configured by entering the OER master configuration mode:

```
Router(config)# oer master
```

The border routers are statically configured:

```
Router(config-oer-mc)# border <IP> key-chain <KEY CHAIN>
```

The interfaces of the BRs are entered. The external interfaces participate in OER, whereas the internal interfaces are used for com-munication to the inside network:

```
Router(config-oer-mc-br)# int <INTERFACE> [internal¦external]
```

For OER to function properly, at least two interfaces within the domain must be configured as external interfaces. These can be on separate routers, but at least one external interface must be configured on each BR.

 Please see page 245 for more details.

The configuration steps for a BR are easier. Because the MC performs the majority of the work, it makes sense that it contains more configuration steps. The BR must be configured with the same key chain information as the MC to perform the authentica-tion. Then the router can be configured as a OER BR:

```
Router(config)# oer border
```

Next, the local interface performing the OER communication to the MC is configured:

```
Router(config-oer-br)# local <INT>
```

Finally, the master controller must be configured:

```
Router(config-oer-br)# master <IP> key-chain <KEY CHAIN>
```

From the MC you can view the status of the BRs:

```
Router# show oer border
```

Link-Efficiency Tools

As a final category of QoS tools, consider how to make the most of the often-limited bandwidth on WAN links. Data can be compressed before it is sent, or large payloads can be fragmented, so that smaller payloads can be interleaved among those fragments to prevent excessive serialization delay (the time it takes for packets to exit an interface). This approach is referred to as link fragmentation and interleaving (LFI). The category of tools under which compression and LFI fall is called link-efficiency tools. First, consider header compression.

One way to preserve bandwidth on the WAN is to compress the TCP and UDP headers. However, this compression does not actually run any sort of compression algorithm. Instead, header compression leverages that most of the information in a packet's header does not change during the session. For example, the source and destination IP addresses usually remain the same during the session. Likewise, the source and destination TCP/UDP port numbers typically do not vary during the session. Therefore, information that does not change during the session is cached in the routers at each end of a link. A much slimmed-down header contains things such as the session context ID (CID), which identifies the particular flow that the packet is associated with, and perhaps a checksum is sent as a compressed header. The routers at each end of the link combine the compressed header with the cached header to generate a standard header, which is applied to a packet before sending the packet to the destination.

 Please see page 245 for more details.

Following is the syntax to configure TCP header compression in interface configuration mode for both PPP or High-level Data Link Control (HDLC) links and Frame Relay circuits:

- **ip tcp header-compression [passive]:** Enables TCP header compression on a PPP or HDLC interface
- **frame-relay ip tcp header-compression [passive]:** Enables TCP header compression on a Frame Relay interface

Voice is carried by the RTP, which is encapsulated inside UDP. When combined, the IP, UDP, and RTP headers on voice packets total approximately 40 bytes in size. However, after enabling RTP Header Compression (cRTP), the header size is reduced to approximately 2 to 4 bytes, thus permitting more voice calls on a WAN link. Following is the syntax to configure RTP header compression in interface configuration mode for PPP, HDLC, or Frame Relay circuits:

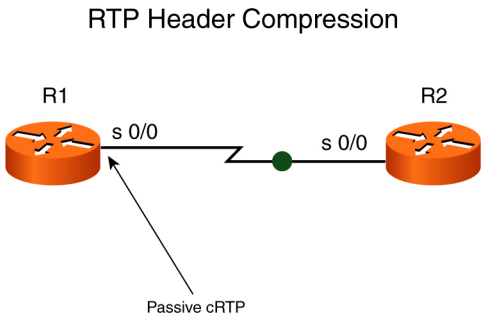
- **ip rtp header-compression [passive]:** Enables RTP header compression on a PPP or HDLC interface
- **frame-relay ip rtp header-compression [passive]:** Enables RTP header compression on a Frame Relay interface

Notice the optional **passive** keyword in the preceding commands. When the **passive** keyword is specified, these interfaces send compressed headers only if they receive compressed headers.

In the following example, routers R1 and R2 are interconnected using their serial 0/0 interfaces. The goal is to configure cRTP between the routers.

 Please see page 245 for more details.

FIGURE 5-7
RTP Header Compression




```
R1(config)# interface serial 0/0
R1(config-if)# ip rtp header-compression passive

R2(config)# interface serial 0/0
R2(config-if)# ip rtp header-compression
```

Only one side of the link uses the **passive** keyword. If both sides are set to be passive, cRTP does not occur because neither side of the link ever sends compressed headers.

To reduce the latency experienced by a large packet exiting an interface (that is, serialization delay), Multilink PPP (MLP) can be used in a PPP environment, and FRF.12 can be used in a VoIP over Frame Relay environment. First, consider MLP. Multilink PPP, by default, fragments traffic. This characteristic can be leveraged for QoS purposes, and MLP can be run even over a single link. The MLP configuration is performed under a virtual multilink interface, and then one or more physical interfaces can be assigned to the multilink group. The physical interface does not have an IP address assigned. Instead, the virtual multilink interface has an IP address assigned. For QoS purposes, a single interface is typically assigned as the sole member of the multilink group. Following is the syntax to configure MLP:

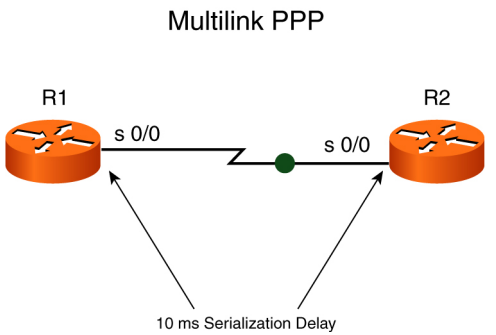
- **`interface multilink [multilink_interface_number]:`** Creates a virtual multilink interface
- **`ip address ip_address subnet_mask:`** Assigns an IP address to the virtual multilink interface
- **`ppp multilink:`** Configures fragmentation on the multilink interface

 Please see page 245 for more details.

- **`ppp multilink interleave`:** Shuffles the fragments
- **`ppp fragment-delay [serialization_delay]`:** Specifies how long it takes for a fragment to exit the interface
- **`encapsulation ppp`:** Enables PPP encapsulation on the physical interface
- **`no ip address`**: Removes the IP address from the physical interface
- Associates the physical interface with the multilink group

In the following example, the goal is to configure MLP on Routers R1 and R2 so that they have a serialization delay of 10 ms on their serial 0/0 interfaces.

FIGURE 5-8
Multilink PPP




```
R1(config)# interface multilink 1
R1(config-if)# ip address 10.1.1.1 255.255.255.0
R1(config-if)# ppp multilink
R1(config-if)# ppp multilink interleave
R1(config-if)# ppp fragment-delay 10
R1(config-if)# exit
R1(config)# interface serial 0/0
R1(config-if)# encapsulation ppp
R1(config-if)# no ip address
```

 Please see page 245 for more details.

```
R1(config-if)# multilink-group 1

R2(config)# interface multilink 1
R2(config-if)# ip address 10.1.1.2 255.255.255.0
R2(config-if)# ppp multilink
R2(config-if)# ppp multilink interleave
R2(config-if)# ppp fragment-delay 10
R2(config-if)# exit
R2(config)# interface serial 0/0
R2(config-if)# encapsulation ppp
R2(config-if)# no ip address
R2(config-if)# multilink-group 1
```

LFI can also be performed on a Frame Relay link using FRF.12. The configuration for FRF.12 is based on an FRTS configuration. Only one additional command is given, in map-class configuration mode, to enable FRF.12. The syntax for that command is as follows:

- **Router(config-map-class)#frame-relay fragment fragment-size:** Specifies the size of the fragments

As a rule of thumb, the packet size should be set to the line speed divided by 800. For example, if the line speed is 64 kbps, the fragment size can be calculated as follows:

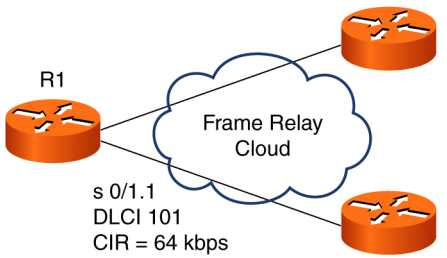
```
fragment size = 64,000 / 800 = 80 bytes
```

This rule of thumb specifies a fragment size (80 bytes) that creates a serialization delay of 10 ms.

The following example shows an FRF.12 configuration to create a serialization delay of 10 ms on a link clocked at a rate of 64 kbps. Because FRF.12 is configured as a part of FRTS, CIR and Bc values are also specified.

 Please see page 245 for more details.

FIGURE 5-9
FRF.12




```
R1(config)# map-class frame-relay FRF12-EXAMPLE
R1(config-map-class)# frame-relay cir 64000
R1(config-map-class)# frame-relay bc 640
R1(config-map-class)# frame-relay fragment 80
R1(config-map-class)# exit
R1(config)# interface serial 0/1
R1(config-if)# frame-relay traffic-shaping
R1(config-if)# interface serial 0/1.1 point-to-point
R1(config-subif)# frame-relay interface-dlci 101
R1(config-fr-dlci)# class FRF12-EXAMPLE
```

AutoQoS

Optimizing a QoS configuration for VoIP can be a daunting task. Fortunately, Cisco added a feature called AutoQoS to many of its router and switch platforms to automatically generate router-based or switch-based VoIP QoS configurations.

The following router platforms support AutoQoS:

- 1700 series
- 2600 series

 Please see page 245 for more details.

- 3600 series
- 3700 series
- 7200 series

Cisco also supports the AutoQoS feature on the following Catalyst switch series:

- 2950 (EI)
- 3550
- 4500
- 6500

On a router platform, the following command enables AutoQoS from either interface configuration mode or DLCI configuration mode (for a Frame Relay circuit):

```
auto qos voip [trust] [fr-atm]
```

The **trust** option indicates that AutoQoS should classify voice traffic based on DSCP markings, instead of using NBAR. The **fr-atm** option enables the AutoQoS feature for Frame Relay-to-ATM links and is issued from DLCI configuration mode.

Before enabling AutoQoS on a router interface, consider these prerequisites:

- CEF must be enabled.
- A QoS policy must not be currently attached to the interface.
- The correct bandwidth should be configured on the interface.
- An IP address must be configured on an interface if its speed is less than 768 kbps.

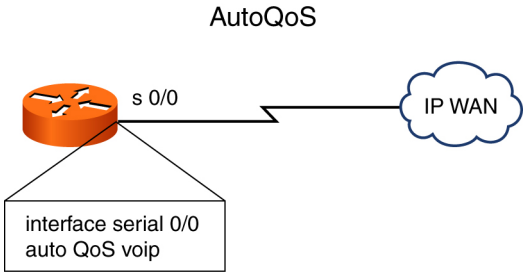
The interface's bandwidth determines which AutoQoS features are enabled. If an interface's bandwidth is less than 768 kbps, it is considered a low-speed interface. On a low-speed interface, AutoQoS configures MLP, which requires an IP address on the physical interface. AutoQoS takes that IP address from the physical interface and uses it for the virtual multilink interface it creates.

 Please see page 245 for more details.

To verify that AutoQoS is configured for a router interface, you can use the following command:

```
show auto qos voip [interface interface-identifier]
```

FIGURE 5-10
AutoQos



The Catalyst 6500 running in Hybrid mode (that is, using the CatOS for switch functions) also supports AutoQoS. To enable AutoQoS on a Hybrid mode Catalyst 6500, you must first enable AutoQoS globally and then for a specific port. Following are the required commands:

- **`set qos autoqos:`** Globally enables AutoQoS
- **`set port qos <mod/port> autoqos trust [cos ¦ dscp]:`** Enables AutoQoS for a specific port

The Catalyst 6500 can trust either CoS or DSCP values for its queuing decision. If the port trusts DSCP markings, you can add the following command, which recognizes that the port is connected to a Cisco IP Phone or a Cisco SoftPhone, which is software that runs on a PC:

```
set port qos <mod/port> autoqos voip
 [ciscosoftphone ¦ ciscoipphone]
```

The port must have CDP (Cisco Discovery Protocol) Version 2 enabled to recognize an attached Cisco IP Phone. Although they do not recognize a Cisco SoftPhone, AutoQoS can be configured on Catalyst 2950 (EI) and 3550 switches, and their AutoQoS feature recognizes a Cisco IP Phone. To configure AutoQoS on these platforms, issue the following commands from interface configuration mode:

 Please see page 245 for more details.

- **`auto qos voip trust`:** Configures the interface to trust CoS markings for classifying VoIP traffic
- **`auto qos voip cisco-phone`**: Detects the presence of a Cisco IP Phone, using CDP

To troubleshoot and verify AutoQoS on a Catalyst switch, you can use the following commands:

- **`show auto qos [interface interface-identifier]`:** Displays the configuration applied by AutoQoS
- **`show mls qos interface [interface-identifier]`:** Displays interface-level QoS statistics

This section has broadly addressed the features enabled by AutoQoS. The specific features are shown in Table 5-3.

Table 5-3 AutoQoS Features

QoS Mechanism	Router Feature	Switch Feature
Classification	NBAR and DSCP	Port trust states
Marking	CB-Marking	CoS to DSCP re-marking
Congestion Management	LLQ	WRR
Shaping	CB-Shaping or FRTS	
Link efficiency	Header compression and LFI	

 Please see page 245 for more details.

Chapter 6

Network Optimization

IP Service Level Agreement (SLA)

One of the most important aspects in maintaining a network is providing a guarantee of a specific level of service to customers. To ensure that such an agreement is met at all times, IOS provides a mechanism to actively test specific metrics, called IP SLA. When configured, the IP SLA service actively monitors a specific aspect of the network, such as UDP VOIP jitter, DNS response time, ping latency, and so on. If the IP SLA thresholds are not met, IOS sends a notification, such as an SNMP trap or syslog message.

To create a basic IP SLA monitor, the type, options, and frequency must be specified. After the monitor has been created, a schedule is build that kicks off the monitor. To monitor the round-trip response time between a router and an IP, you can use the ICMP Echo Operation:

```
Router(config)# ip sla monitor <OPERATION #>
Router(config-sla-monitor)# type echo protocol ipIcmpEcho <DESTINATION>
Router(config-sla-monitor)# frequency <SECONDS>
Router(config-sla-monitor)# exit
Router(config)# ip sla monitor schedule <OPERATION #> [life {forever | seconds}] [start-time {hh:mm[:ss]
[month day | day month] | pending | now | after hh:mm:ss] [ageout seconds] [recurring]
```

Some monitors require that a responder be configured (UDP Jitter, UDP Echo, and TCP Connect) on one router:

```
Router(config)# ip sla monitor responder
```

 Please see page 245 for more details.

Following are other IP SLA monitor operations:

UDP Jitter: **`type jitter`**
VOIP Jitter: **`type`**
VOIP Gatekeeper Delay: **`type voip delay gatekeeper registration`**
UDP Echo: **`type udp echo`**
HTTP Connect: **`type http operation`**
TCP Connect: **`type tcpConnect`**
ICMP Echo: **`type echo protocol ipIcmpEcho`**
ICMP Path Echo: **`type pathEcho protocol ipIcmpEcho`**
ICMP Path Jitter: **`type pathJitter`**
FTP Operations: **`type ftp`**
DNS Operations: **`type dns`**
DHCP Operations: **`type dhcp`**

NetFlow

As packets are sent through router interfaces, they can be classified into flows. This information can be sent from the router to a monitoring server that can provide valuable information about the traffic traversing the network. A flow can be described by a number of fields:

- Source IP address
- Destination IP address
- Source port
- Destination port
- Protocol type
- Type of Service
- Interface

 Please see page 245 for more details.

To determine if a packet belongs in a particular flow, the seven packet fields are inspected. If any one of the fields is different, the packet in question can be considered a new flow. NetFlow statistics can be collected on the following types of networks: IP, Frame Relay, MPLS, and ATM.

To enable NetFlow on an interface, the **ip flow** command set is used. To configure NetFlow on an interface:

```
Router(config-if)# ip flow ingress
Egress support can also be added:
Router(config-if)# ip flow egress
```

To configure the router to export the NetFlow data to a NetFlow server

```
Router(config)# ip flow-export {destination {ip-address | hostname} udp-port | source {interface-name}
| version {1 | [{5 | 9} [origin-as | peer-as] [bgp-nexthop]]} | template {refresh-rate packets | time-
out-rate minutes} [options {export-stats | refresh-rate packets | sampler | timeout-rate minutes}]}
```

To be exported to the NetFlow collection server, the flow must have been exported from the flow cache. Active flows (when there is an ongoing conversation) live for 30 minutes by default before they are exported. Inactive flows, those that have been terminated, are sent after 15 seconds. These values are configurable:

```
Router(config)# ip flow-cache timeout [active minutes | inactive seconds]
```

The number of flows that can be collected can be modified with:

```
Router(config)# ip flow-cache entries <#>
```

SPAN, RSPAN, and Router IP Traffic Export (RITE)

Viewing the packets between two devices is often the best way to troubleshoot a networking issue. Many organizations deploy network monitoring servers for both network functionality and security purposes. To provide these servers a stream of data from all segments of the network, the Switched Port Analyzer (SPAN) mechanism built into Cisco switches can be used. A SPAN port is a physical port that is configured to send data received on other ports or VLANs. When the data is sent out the SPAN port, it is simply a copy of all the data that has been sent through the configured source ports.

To configure the source of the data to be sent out the SPAN port

 Please see page 245 for more details.

```
Router(config)# monitor source <#> source {interface interface-id | vlan vlan-id} [, | -] [both | rx | tx]
```

Either a source interface or entire VLAN can be specified. You can use the command multiple times with the same session number if multiple sources are used.

To configure the destination of the data to be sent:

```
Router(config)# monitor destination <#> destination interface <INT>
```

The data received in the source interface is sent to the specified destination interface.

The SPAN functionality assumes that the destination of the traffic is directly connected to the switch. If the network is large, it might not be possible to wire a single monitoring station to multiple switches with SPAN ports. Fortunately a special remote SPAN (RSPAN) VLAN can be created that transports the SPAN port information to another switch. This enables an aggregation of monitoring data into a single VLAN that can be sent to the monitoring station.

To configure the RSPAN VLAN, the **remote-span** command must be specified within the VLAN configuration mode:

```
Router(config-vlan)# remote-span
Specify the destination of the SPAN port as the remote-span VLAN:
Router(config)# monitor session <#> destination remote vlan <VLAN>
```

This is great for switches, but is there a similar technology for a router? Yes! The Router IP Traffic Export (RITE) mechanism can export traffic to specific devices defined by the MAC address. To configure, a RITE profile is created and then it's applied to an interface. The traffic that RITE applies to can be limited by ACLs.

To configure the profile

```
Router(config)# ip traffic-export profile <PROFILE NAME>
Specify the outgoing interface:
Router(config-rite)# interface <INT>
Specify the MAC address to send the traffic to:
Router(config-rite)# mac-address <MAC>
```

 Please see page 245 for more details.

Specify whether bidirectional traffic is necessary. Without this command, only packets incoming to the router are sent:

```
Router(config-rite)# bidirectional
```

Optionally, ACLs can limit the traffic sent:

```
Router(config)# incoming {access-list {standard | extended | named} | sample one-in-every packet-number}
Router(config)# outgoing {access-list {standard | extended | named} | sample one-in-every packet-number}
```

Apply the RITE policy to an interface:

```
Router(config-if)# ip traffic-export apply <POLICY NAME>
```

Cisco IOS Embedded Event Manager (EEM)

In the normal operations of the switch or router, events such as a CLI command, a syslog message, or an SNMP trap, for example, are constantly occurring. These events are detected by the EEM Event Detectors that send their information to the EEM server. The EEM server can then be programmed to implement an EEM policy. The two different types of EEM policies are applets and TCL scripts, which can be programmed to perform a variety of tasks, such as send syslog messages and SNMP traps, fire off emails, and even open raw sockets.

An applet can be configured directly within the IOS CLI by first creating a policy and registering it:

```
Router(config)# event manager applet <APPLET NAME>
```

The applet must be configured to detect a specific event. There are a number of different event detectors, each with their own syntax:

```
Router(config-applet)#  event <EVENT DETECTOR>
```

- **application**: Application-specific event
- **cli:** CLI event
- **counter:** Counter event
- **interface:** Interface event
- **ioswdsysmon:** IOS WDSysMon event

 Please see page 245 for more details.

- **none:** Manually run policy event
- **oir:** OIR event
- **snmp:** SNMP event
- **syslog: Syslog event**
- **timer:** Timer event

The actions of the applet that are performed when the programmed event occurs are entered line by line:

```
Router(config-applet)# action <LABEL> <ACTION>
```

Some of the actions are predefined within IOS:

- Executing a Cisco IOS command-line interface (CLI) command (cli)
- Setting or modifying a named counter (counter)
- Switching to a secondary processor in a fully redundant hardware configuration (force-switchover)
- Requesting system information when an event occurs (info)
- Sending a short e-mail (mail)
- Manually running an EEM policy (policy)
- Publishing an application-specific event (publish-event)
- Reloading the Cisco IOS software (reload)
- Generating an SNMP trap (snmp-trap)
- Generating prioritized syslog messages (syslog)

Unfortunately, a TCL script cannot be created on the CLI, it must be uploaded to the router. When uploaded, it must be registered:

```
Router(config)# event manager policy <FILENAME>
```

 Please see page 245 for more details.

Remote Monitoring (RMON)

There are times when it is convenient to configure a router to alert when certain thresholds are exceeded. Remote Monitoring (RMON) can be configured to send an SNMP trap when the conditions of the RMON are satisfied. An RMON statement consists of rules about limits on OID values. The RMON rules use either an absolute value of an OID or the change in the value of an OID over time. If the value used rises above a certain threshold, IOS can send an SNMP trap; when the value drops below another value, IOS can send a different SNMP trap or syslog message.

To configure an RMON rule

```
Router(config)# rmon alarm <#> <OID> <INTERVAL> {delta | absolute} rising-threshold value [event-number] falling-threshold value [event-number] [owner string]
```

The interval specifies the amount of time between checks of the OID's value. The delta or absolute value specifies whether RMON should use the value of the OID or changes in the value of the OID. The absolute value of the OID is useful if values such as temperature are queried. It isn't as important to have the change in temperature, just the actual temperature. Changes in the OID value are useful if metrics such as CPU usage are monitored. Changes in the CPU usage can be tracked easily. The value after the rising threshold attribute specifies when an SNMP trap or syslog message will be sent if exceeded. It can also be tied to an RMON event that is used in the SNMP notified. Likewise the value following the falling-threshold attribute specifies when an SNMP trap or syslog message will be sent if fallen below. An RMON event can also be specified.

An RMON event is a custom notification that is tied to a RMON alarm. As the thresholds within the alarm are crossed, the event can be used. To configure an RMON event

```
Router(config)# rmon event <#> [log] [trap community] [description string]
```

If the log option is used, a syslog message will be sent. The trap option specifies the trap community to be used. The description contains a custom message which describes the threshold which has been crossed.

FTP

File Transfer Protocol (FTP) is a protocol that was invented in the early days of the Internet. It defines a way that files can be transferred between two TCP/IP hosts. Currently, FTP is strongly discouraged because of its lack of any security features. It does not encrypt the traffic, so privacy is nonexistent and authentication services are laughable (when the username and password are transmitted in cleartext). A Cisco device can act as an FTP client to transfer files from the device to an FTP server. This is common when core dump files are generated and need to be saved:

 Please see page 245 for more details.

To configure the username that should be used when connecting to the FTP server:

```
Router(config)# ip ftp username <USERNAME>
```

To configure the password used:

```
Router(config)# ip ftp password <PASSWORD>
```

Problems can occur, particularly when traffic traverses a firewall when using active FTP. With active FTP the client initiates the control session but the server initiates the data session.

Passive FTP solves this problem by having the client initiate both sessions. Passive FTP is used instead of active with the following command:

```
Router(config)# ip ftp passive
```

The source interface to be used for the FTP connection can also be explicitly specified:

```
Router(config)# ip ftp source-interface <INT>
```

To configure the router to send core dumps via FTP

```
Router(config)# exception protocol ftp
Router(config)# exception dump <IP>
Router(config)# exception core-file <FILENAME>
```

TFTP

When attempting to transfer files, it is convenient to use a simple, lightweight protocol. The Trivial File Transfer Protocol (TFTP) provides a mechanism to quickly transfer small files using a Cisco device without having to worry about authentication. Cisco devices have contained the capability to TFTP files from a TFTP servers since the early days.

To copy a file from a TFTP server from the Cisco CLI, use the **copy** command with the tftp: option:

```
Router# copy tftp://<IP>/<FILE> flash:
```

The Cisco device contacts the TFTP server and attempts to download the file and save it to the location specified, usually flash or nvram. A problem that occurs when using TFTP to transfer bigger files is the maximum file size limitation. Other file transfer protocols such as FTP or SCP do not have limitations.

 Please see page 245 for more details.

A device can also be set up as a TFTP server that enables it to provide system images to requesting TFTP clients. This enables devices to boot up without a local image. Instead they can point to a device that serves the TFTP-bootable image.

To point to the TFTP-bootable image located on the local Cisco IOS filesystem and to enable the TFTP server processes

```
Router(config)# ip tftp-server flash <FILE>
```

An ACL can also be used to allow only specific hosts to access the TFTP server. The ACL must be configured to permit hosts that should be allowed access and deny all other hosts. After the ACL is build, it can be referenced by the **tftp-server** command:

```
Router(config)# ip tftp-server flash <FILE> <ACL>
```

Only authorized hosts can download the image file.

Secure Copy Protocol (SCP)

Secure Copy Protocol (SCP) adds encryption to file transfer protocols such as FTP and TFTP. In environments which require more, secure protocols can take advantage of SCP. There also aren't any problems with maximum file size when using SCP. Because SCP is encrypted, the underlying authentication is stronger as well. Underlying SCP is the secure shell (SSH) protocol that must be enabled on the Cisco device before SCP can be used. AAA authentication must also be used to provide an authentication and authorization mechanism to the IOS.

To enable SCP, AAA must be enabled:

```
Router(config)# aaa new-model
Router(config)# aaa authentication login default local
Router(config)# aaa authorization exec default local
```

A central authentication server can also provide authentication services:

```
Router(config)# aaa new-model
Router(config)# aaa authentication login default group tacacs+
Router(config)# aaa authorization exec default group tacacs+
```

 Please see page 245 for more details.

When the AAA services have been configured and SSH is properly configured, the SCP server can be enabled:

```
Router(config)# ip scp enable
```

In addition to the configuration of an SCP server, IOS can also act as an SCP client. From the CLI, files can be transferred from or to an SCP server. This enables the same file transfer capabilities without requiring a service to run on the network infrastructure devices. It also can be used instead of the more insecure file transfer protocols, but the syntax is largely the same.

To copy a file from an SCP server:

```
Router# copy scp://<IP>/<FILE> flash:<FILE>
```

To copy a file to an SCP server:

```
Router# copy flash:<FILE> scp://<IP>/<FILE>
```

HTTP and HTTPS

A less common way to configure Cisco routers and switches is through the HTTP/HTTPS servers that are built in. They provide a graphical interface to some of the functionality available on the devices. The server is actually enabled by default, but to take advantage of it, standard authentication must occur first. This authentication can happen with local accounts, TACACS, or using standard AAA mechanisms.

Originally the HTTP server was implemented, but it did not provide encrypted transmission. This was added with the addition of the HTTPS server. If web management is used, the HTTPS server should be used rather than the unencrypted HTTP server.

To enable the HTTP server

```
Router(config)# ip http server
```

To enable the HTTPS server, cryptographic keys must first generate all the encryption of all of data:

```
Router(config)# crypto keys generate rsa usage-keys modules 2048
```

After the keys have been created, you can start the HTTPS server:

```
Router(config)# ip http secure-server
```

 Please see page 245 for more details.

Typically the standard port for running HTTP services is TCP/80 and for running HTTPS is TCP/443. These defaults can be changed if necessary. To change the port on which the HTTP server runs:

```
Router(config)# ip http port <PORT #>
```

To change the port on which the HTTPS server runs:

```
Router(config)# ip http secure-port <PORT #>
```

It is also possible to supply an ACL that defines which hosts can access the HTTP or HTTPS server. The IPs allowed through the ACL specify which hosts will be allowed access. To apply the ACL to the HTTP/HTTPS server

```
Router(config)# ip http access-class <ACL>
```

Telnet

One of the most common protocols used to manage Cisco devices is telnet. Like FTP it is an insecure protocol created when the Internet was a safer place. Because it isn't encrypted like SSH, it should be avoided if at all possible. Unfortunately, it is universally enabled and must be manually disabled. As a management protocol it is lightweight and understood by a large number of client applications and works extremely well. The only major concern when using telnet is the security implications. It is extremely easy for an attacker to sniff telnet traffic, capture passwords, configuration files, and passively learn about a network without actively attacking it. As networks grow larger and larger, management traffic can pass through many different companies, even countries before it reaches it's destination. Because there aren't any encryption keys to worry about, setup is out-of-the-box , it's footprint is extremely small and has low overhead.

Telnet can also be used to establish a management connection from a Cisco router; the exec mode enables the command **telnet <HOST>**.

To enable telnet usage on a vty line, it must be listed in the transport input protocols:

```
Router(config-line)#  transport input telnet
```

To disable telnet on a vty line, it must not be listed as one of the transport input protocols. For instance, if SSH is used as a management protocol instead

```
Router(config-line)#  transport input ssh
```

 Please see page 245 for more details.

To view the available transport input mechanisms:

```
Router# show line vty <#>
```

The capability to telnet from the device can be limited by not specifying it among the allowed output protocols:

```
Router(config-line)# transport output ssh
```

To re-enable the ability to telnet from the device:

```
Router(config-line)# transport output telnet
```

To stop all IP addresses from accessing the telnet service on the device, an ACL can be applied that specifies which IPs are allowed access. The ACL must be applied on the vty lines:

```
Router(config-line)# access-class <ACL>
```

SSH

Arguably the best choice for device management in terms of security and usability is the Secure Shell (SSH) protocol. It is infinitely more secure than telnet (because telnet doesn't encrypt the traffic), and it offers full control over the device. More and more businesses and organizations rely on the network for mission critical data transport. As the value of data has risen, the consequences of compromise of that data skyrocket. As a result, small steps such as using SSH over telnet help to avert risk within the network infrastructure. Initially, Cisco supported only the telnet protocol. The usage of SSH began with support of SSHv1 that has some large known security vulnerabilities. Currently, SSHv2 (the industry standard) is supported on all modern Cisco equiFundamentally, the encryption of data requires the use of some sort of key. Encryption keys must be generated with IOS before SSH can be used. They are not created by default. To generate the keys you must configure a hostname and domain name that will identify them. Then the key can be generated:

```
Router(config)# hostname R1
R1(config)# ip domain-name cisco.com
R1(config)# crypto key generate rsa general-keys modulus <KEY SIZE>
```

 Please see page 245 for more details.

Encryption keys should be created in the largest supported size. Generally, 2048 is the smallest size that should be used in modern computing. As the power and speed of computation increases, so does the ability for an attacker to compromise the keys with a brute force attack. After the keys are generated, the SSH service automatically starts. To ensure that only version 2 runs, it must be directly specified:

```
Router(config)# ip ssh version 2
```

```
pment.
```

 Please see page 245 for more details.

Chapter 7

WAN

Implement High-Level Data Link Control (HDLC) and PPP HLDC

High-level Data Link Control (HDLC) is a common link layer technology in use today. The two main functions of HDLC are link management and data transfer. As part of the data transfer, HDLC provides error detection and flow control.

Although HDLC is the default encapsulation on a Cisco serial interface, it can be manually configured using the following interface command:

```
Router(config-int)# encapsulation hdlc
```

The current encapsulation on a serial interface can be viewed with the **show interface** command:

```
Router# sho int s0/0/0
Serial0/0/0 is administratively down, line protocol is down
  Hardware is GT96K Serial
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
```

 Please see page 245 for more details.

PPP

One of the most important networking protocols is the Point-to-Point Protocol (PPP). Not only can you use it to transport many network protocols over point-to-point links, but it also provides capabilities to add authentication and link quality testing, allocate IP addresses, and provide error detection. The four major parts to PPP are network datagram encapsulation (such as HDLC), link setup, termination and maintenance, and network layer protocol establishment and configuration.

When you establish a PPP link, the link setup and maintenance protocol, LCP, is used. LCP is responsible for PPP link establishment, termination, and maintenance. When the link is established, different parameters are exchanged and the link can optionally be quality tested. After the link establishment phase, the Network Control Protocols (NCP) takes over and facilitates the transfer of Layer 3 protocols, such as IP.

To configure a serial link to use PPP, simply set the encapsulation:

```
Router(config-if)# encapsulation ppp
```

One of the features of PPP encapsulation is the capability to authenticate the link. PPP can use either Password Authentication Protocol (PAP) or Challenge-Handshake Authentication Protocol (CHAP). CHAP authentication is considered more secure than PAP authentication because the username and password are not transferred in the clear as they are in PAP. Instead CHAP uses more secure hashs and frequent challenges that add security to the authentication process.

To configure PAP on a serial interface, the authentication credentials must be created first. This is typically done by creating a username and password combination on each router. The interface then contains a command that specifies what will be sent to the peer router. If the two match, authentication will succeed:

```
Configuration on R1 -
R1(config)# username Router2 password Cisco
R1(config)# int <INTERFACE>
R1(config-if)# encapsulation ppp
R1(config-if)# ppp authentication pap
R1(config-if)# ppp pap sent-username Router1 password Cisco
```

 Please see page 245 for more details.

```
Configuration on R2 -
R2(config)# username Router1 password Cisco
R2(config)# int <INTERFACE>
R2(config-if)# encapsulation ppp
R2(config-if)# ppp authentication pap
R2(config-if)# ppp pap sent-username Router2 password Cisco
```

CHAP authentication is configured slightly different because the hostname of the routers are involved. A username is created matching the hostname of the peer router, and the password supplied must match:

```
Router(config)# hostname R1
R1(config)# username R2 password Cisco
```

This means that when R1 tries to authenticate to R2, it will use the password Cisco.

```
R1(config)# int <INTERFACE>
R1(config-if)# encapsulation ppp
R1(config-if)# ppp authentication chap

Router(config)# hostname R2
R2(config)# username R1 password Cisco
R2(config)# int <INTERFACE>
R2(config-if)# encapsulation ppp
R2(config-if)# ppp authentication chap
```

Unfortunately, configuring PAP and CHAP on a PPP over Frame Relay is not be as easy as enabling or disabling the authentication. When dealing with Frame Relay interfaces, a virtual-template must be configured and applied to the interface.

First, a virtual-template interface must be created, and it's encapsulation must be PPP:

```
Router(config)# hostname R1
R1(config)# int virtual-template <#>
R1(config-if)# encapsulation ppp
```

 Please see page 245 for more details.

The IP address from the serial interface is moved to the virtual-template:

```
R1(config)# int <SERIAL INTERFACE>
R1(config-if)# no ip address
R1(config-if)# int virtual-template <#>
R1(config-if)# ip address <IP> <MASK>
```

The **frame-relay interface-dlci** is programmed to reference the virtual-template:

```
R1(config-if)# frame-relay interface-dlci <DLCI> ppp virtual-template <#>
```

Now authentication can be configured on the virtual-template as before:

```
R1(config-if)# ppp authentication chap
```

or

```
R1(config-if)# ppp authentication pap
```

The details of how to configure the authentication are the same using the new virtual-template interfaces.

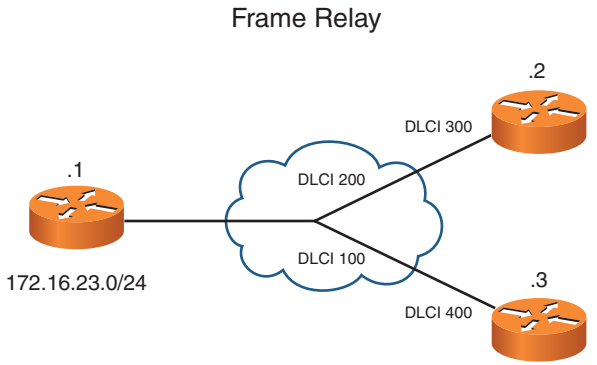
Frame Relay Local Management Interface (LMI)

Connection between customer (data terminal equipment, DTE) and service provider (data communications equipment, DCE) is User-Network Interface (UNI). It is in this area in which Frame Relay operates. The Network-to-Network Interface (NNI) describes how different Frame Relay provider networks interconnect. Providers often use ATM in the cloud to carry the Frame Relay data.

Frame Relay multiplexes many virtual circuits (VC) over a single physical transmission link. It uses data-link connection identifiers (DLCI) on each DTE to identify the different virtual circuits. The DLCI is typically only locally significant between the DTE and the frame switch. Some providers allow the customers to choose the DLCI. DLCIs 0–15 and 1008–1023 are reserved. The specific range of DLCIs available is dependent upon the Local Management Interface (LMI) type in use. DLCIs must be mapped to a remote IP address to direct traffic over the correct VC. Cisco routers support dynamic (Inverse Address Resolution Protocol, IARP) and manual mappings of DLCIs to remote IP addresses.

 Please see page 245 for more details.

FIGURE 7-1
Frame Relay



LMI provides signaling and status updates between the DTE and DCE. It also provides the DTE with its DLCI. The LMI can be autosensed on Cisco IOS Release 11.2 or later. One of three types is used: Cisco, American National Standards Institute (ANSI), or Q.933. Possible LMI status indications include the following:

- **Active:** Connection is active, and the routers can exchange data.
- **Inactive:** Local connection is functioning, but the remote connection is not.
- **Deleted:** No LMI received from switch, DLCI removed from switch, or no service from DTE to DCE.

Implement Frame Relay Discard Eligible (DE)

A Frame Relay frame contains a number of fields, one of which is the Discard Eligible bit. A frame with the Discard Eligible bit set indicates the preference of that frame to be dropped due to congestion over those without the bit set. In periods of congestion, the service provider drops frames above the CIR, and the DE bit provides a way to specify prioritization. This enables less important information to be dropped prior to critical packets. A router can be configured to set the DE bit of specific types of traffic using QoS mechanisms such as MQC and de-lists.

Full Mesh

Full mesh is a networking topology in which each router in the network directly connects to every other router in the network. This means that a virtual circuit (VC) exists between every router, creating a total of n(n–1) / 2 VCs. This topology is most efficient because only a single hop is necessary to reach any other node on the network. Full-mesh topologies can tolerate a large

 Please see page 245 for more details.

number of link failures and maintain connectivity because the frames can traverse around the failure through other connected nodes. The costs involved in building and maintaining a full-mesh topology are large because of the number of links—not only because of the cost of the links themselves, but also the cost to create and maintain those links. As a result, large full-mesh topologies are often not practical.

Hub and Spoke

Hub and spoke is a network topology in which one router, the hub, connects to every other router in the network. All other routers, the spokes, are connected only to the hub, not other spokes. Network traffic between spokes must travel from the source spoke, through the hub, toward the destination spoke. To create this topology, only one link is required for each spoke, bringing the total number of links to n–1, where n is the number of routers (assuming only one hub). This topology is scalable and less expensive than full mesh but is not as fault-tolerant. A loss of the hub results in complete communication failure between all nodes. As a result, many networks employ more than one hub that can maintain connectivity to the spokes if the loss of one hub occurs.

Nonbroadcast Multiaccess (NBMA) Networks

NBMA enables the customer to communicate with any remote site provided the provider has established a VC. A hub and spoke is often used because of the per-VC charge that typically exists. Permanent virtual circuit (PVC) or switched virtual circuit (SVC) can be used—typically PVC.

Configuring Basic Frame Relay

To set the encapsulation to Frame Relay, use the following command:

```
Router(config-if)# encapsulation frame-relay [cisco | ietf]
```

If you must specify the LMI type, use this command:

```
Router(config-if)# frame-relay lmi-type {ansi | cisco | q933a}
```

For dynamic address mapping (IARP), no further configuration is required. If IARP has been disabled on an interface, you can enable it with the following command:

```
Router(config-if)# frame-relay inverse-arp
```

 Please see page 245 for more details.

To configure a static mapping, use the following interface configuration command:

```
frame-relay map protocol protocol-address dlci [broadcast] [ietf | cisco]
```

The keywords indicate the following:

- **protocol-address:** Specifies the destination protocol address
- **dlci:** The DLCI number needed to connect to the remote protocol address
- **broadcast:** Specifies that broadcasts/multicasts should be forwarded; often used to ensure that routing protocol traffic should be sent across the PVC
- **ietf/cisco:** Used to specify the Frame Relay encapsulation type

Subinterfaces

Subinterfaces can solve split-horizon issues that arise with distance vector protocols and hub-and-spoke topologies. Subinterfaces might be configured as point-to-point or multipoint. Split horizon can still be an issue in the multipoint environment. Multipoint does offer an advantage in that a single subnet is needed as opposed to multiple subnet addresses. The steps for a Frame Relay subinterface configuration include the following:

Step 1. Remove any network layer addressing assigned at the physical interface level.

Step 2. Configure Frame Relay encapsulation at the physical interface level.

Step 3. Create the subinterface using the following command:

```
Router(config)# interface serial number.subinterface-number {multipoint | point-to-point}
```

Step 4. Assign the subinterface a network address; you can use the **ip unnumbered** command if you want to reference an address from another interface, such as a loopback interface.

Step 5. If you configured a point-to-point subinterface—or if you configured multipoint and are not using IARP—you must configure the local DLCI using the following command:

```
Router(config-subif)# frame-relay i nterface-dlci dlci-number
```

NOTE
You cannot assign a subinterface to point-to-point communications and then reassign to multipoint without rebooting the router. To work around this, just select a new subinterface number.

 Please see page 245 for more details.

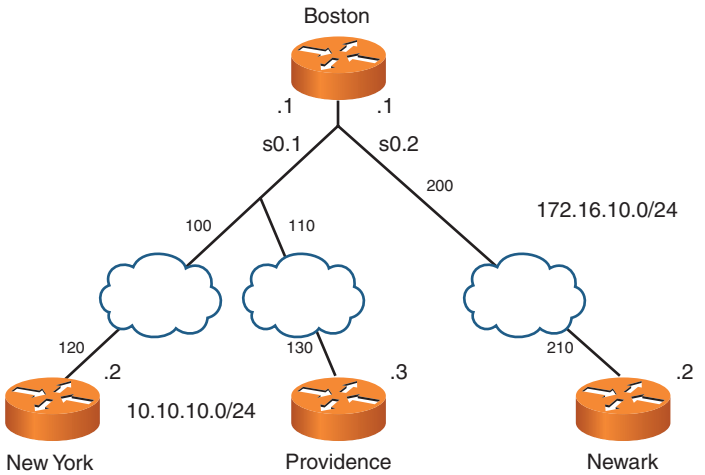


FIGURE 7-2
Frame Relay Example

Following is an example:

```
Boston(config)# interface serial 0
Boston(config-if)# encapsulation frame-relay
Boston(config-if)# no ip address
Boston(config-if)# no shutdown
Boston(config-if)# interface serial 0.1 multipoint
Boston(config-if)# ip address 10.10.10.1 255.255.255.0
Boston(config-if)# frame-relay map ip 10.10.10.2 100 broadcast
Boston(config-if)# frame-relay map ip 10.10.10.3 110 broadcast
Boston(config-if)# interface serial 0.2 point-to-point
Boston(config-if)# ip address 172.16.10.1 255.255.255.0
Boston(config-if)# frame-relay interface-dlci 200

New York(config)# interface serial 0
New York(config-if)# encapsulation frame-relay
New York(config-if)# ip address 10.10.10.2 255.255.255.0
New York(config-if)# frame-relay map ip 10.10.10.1 120 broadcast
New York(config-if)# frame-relay map ip 10.10.10.3 120 broadcast
New York(config-if)# no shutdown
```

 Please see page 245 for more details.

```
Providence(config)# interface serial 0
Providence(config-if)# encapsulation frame-relay
Providence(config-if)# ip address 10.10.10.3 255.255.255.0
Providence(config-if)# frame-relay map ip 10.10.10.1 130 broadcast
Providence(config-if)# frame-relay map ip 10.10.10.2 130 broadcast
Providence(config-if)# no shutdown

Newark(config)# interface serial 0
Newark(config-if)# encapsulation frame-relay
Newark(config-if)# ip address 172.16.10.2 255.255.255.0
Newark(config-if)# frame-relay interface-dlci 210
Newark(config-if)# no shutdown
```

Traffic Shaping Flow Terminology

- **Local Access Rate:** Clock speed of the connection to the Frame Relay cloud; rate at which data flows into or out of the network.
- **Committed Information Rate (CIR):** Rate in bits per second (bps) at which the Frame switch agrees to transfer data; usually averaged over time, called committed rate measurement interval (Tc).
- **Oversubscription:** The sum of all the CIRs of the VCs coming into the device exceeds the access line speed.
- **Committed Burst (Bc):** Maximum data in bits that the Frame switch agrees to transfer during any Tc; Bc (in bits) = CIR (in bits/s) ÷ TC (in s).
- **Excess Burst (Be):** Maximum number of bits the Frame switch attempts to transfer beyond the CIR for the first time interval only.
- **Forward Explicit Congestion Notification (FECN):** Frame switch sets this bit to indicate congestion is experienced.
- **Backward Explicit Congestion Notification (BECN):** Another bit that can be set to indicate congestion on the switch; Cisco IOS Release 11.2 and later enable a router to respond to this bit setting.

 Please see page 245 for more details.

Frame Relay traffic shaping is often used when a speed mismatch exists between sites or you notice that Frame Relay connections are occasionally congested.

Configuring traffic shaping involves the following steps:

Step 1. Specify a map class with the following command:

```
Router(config)# map-class frame-relay map-class-name
```

Step 2. Configure the options for traffic shaping; the following options are available:

- Define the average and peak rates on the VC associated with the map class; use the following command:
  ```
  Router(config-map-class)# frame-relay traffic-rate average [peak]
  ```
- Specify that the router dynamically fluctuates the rate based on BECNs; use the following command:
  ```
  Router(config-map-class)# frame-relay adaptive-shaping
  ```
- Specify a queuing strategy for the virtual circuit; see the QoS section configurations.

Step 3. Map the map class to virtual circuits on the interface; use the following command:

```
Router(config-if)# frame-relay class map-class-name
```

Step 4. Enable traffic shaping with the following command:

```
Router(config-if)# frame-relay traffic-shaping
```

Dynamic Multipoint VPN

Dynamic Multipoint VPN (DMVPN) uses generic routing encapsulation (GRE) tunnels, IPsec encryption, and the Next-Hop Resolution Protocol (NHRP) to better scale IPsec virtual private networks (VPN). Specifically, these protocols combine to provide much easier configurations of VPNs and the dynamic discovery of tunnel endpoints. Ease of configuration is provided because of crypto profiles. These crypto profiles replace the need for defining static crypto maps.

 Please see page 245 for more details.

Remember that DMVPN relies on two Cisco-enhanced standards-based technologies. NHRP is a client/server protocol with the hub as the server and the spokes as clients. This protocol enables the hub to maintain a database of the public IP addresses used on the spokes. Clients can query the database for the address of endpoint spoke systems for the creation of tunnels between them.

mGRE Tunnel Interface enables a single GRE interface to support multiple GRE destinations.

Verifying Frame Relay

- **show interface:** Encapsulation verification
- **show frame-relay pvc:** Status and traffic statistics; BECN and FECN data
- **show frame-relay map:** View DLCI mappings
- **show frame-relay lmi:** LMI traffic statistics
- **debug frame-relay lmi:** Displays LMI information
- **clear frame-relay-inarp:** Clears dynamically created mappings
- **show traffic-shape:** Displays the current traffic shaping configuration
- **show traffic-shape statistics:** Displays the current traffic shaping statistics
- **debug frame-relay lmi**: Displays information on the LMI packet exchange
- **debug frame-relay packet**: Displays packet level of Frame Relay activities

 Please see page 245 for more details.

Chapter 8

IP Multicasting

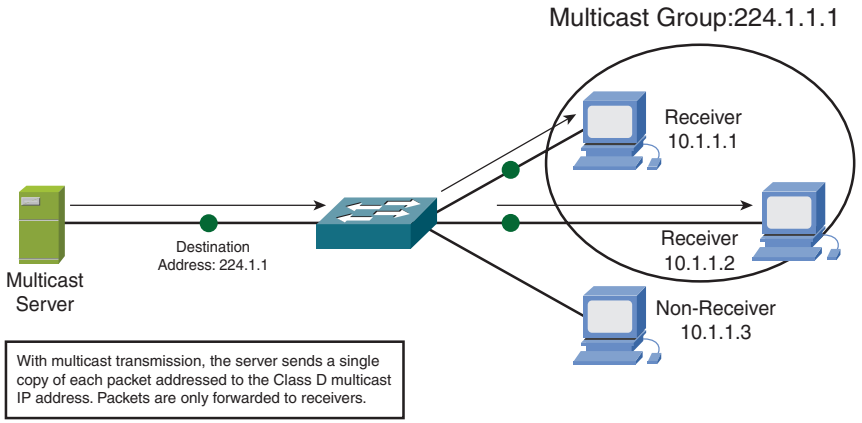
Introduction

Consider a video stream that needs to be sent to multiple recipients in a company. One approach is to unicast the traffic. The source server sends a copy of every packet to every receiver. Obviously, this approach has serious scalability limitations. An alternative approach is to broadcast the video stream so that the source server has to send each packet only one time. However, everyone in the network receives the packet in that scenario, even if they do not want it. IP multicast technologies provide the best of both worlds. With IP multicast, the source server sends only one copy of each packet, and packets are sent only to intended recipients.

Specifically, receivers join a multicast group, denoted by a Class D IP address (that is, in the range 224.0.0.0 through 239.255.255.255). The source sends traffic to the Class D address, and through switch and router protocols, packets are forwarded only to intended stations. These multicast packets are sent via User Datagram Protocol (UDP) (that is, best effort). Therefore, congestion-avoidance mechanisms such as weighted random early detection (WRED), which causes TCP flows to go into TCP slow start, are not effective for multicast. When doing a multicast design, also be aware of the potential for duplicate packets received and the potential for packets to arrive out of order.

 Please see page 245 for more details.

FIGURE 8-1
IP Multicast



Internet Group Management Protocol/Cisco Group Management Protocol

The protocol used between clients (PC) and routers to let routers know which of their interfaces have multicast receivers attached is Internet Group Management Protocol (IGMP). There are three versions of IGMP; however, only two versions are in widescale deployment:

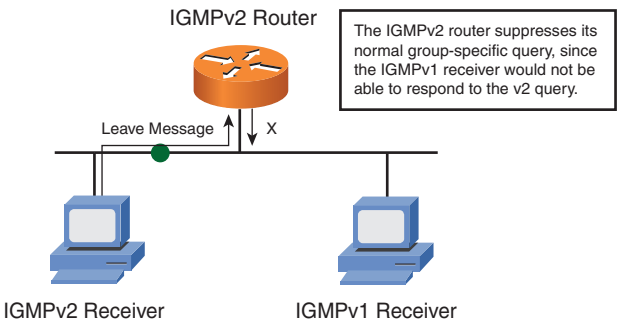
- **IGMP Version 1:** When a PC wants to join a multicast group, it sends an IGMP Report message (often called a Join message) to the router, letting the router know it wants to receive traffic for a specific group. Every 60 seconds, by default, the router sends an IGMP Query message to determine whether the PC still wants to belong to the group. There can be up to a 3-minute delay before the router realizes that the receiver left the group. The destination address of this router query is 224.0.0.1, which addresses all IP multicast hosts.
- **IGMP Version 2:** Similar to IGMP Version 1, except that IGMP Version 2 can send queries to a specific group, and a Leave message is supported. Specifically, a receiver can proactively send a Leave message when it no longer wants to participate in a multicast group, enabling the router to prune its interface earlier.
- **IGMP Version 3:** Introduces Source-Specific Multicast (SSM) capabilities to the protocol. This enables hosts to signal group membership with filtering capabilities for possible sources. A host can signal either that it wants to receive traffic from all sources or that it wants to receive traffic from only specific sources.

IGMP Version 1 and Version 2 hosts and routers do have some interoperability. When an IGMPv2 host sends an IGMPv2 report to an IGMPv1 router, the IGMP message type appears to be invalid, and it is ignored. Therefore, an IGMPv2 host must send IGMPv1 reports to an IGMPv1 router.

 Please see page 245 for more details.

In an environment with an IGMPv2 router and a mixture of IGMPv1 and IGMPv2 receivers, the Version 1 receivers respond normally to IGMPv1 or IGMPv2 queries. However, the Version 2 router must ignore any Leave message while IGMP receivers are present, because if the router processed the IGMPv2 Leave message, it would send a group-specific query, which would not be correctly interpreted by an IGMPv1 receiver.

FIGURE 8-2
IGMP V2 Router with V1 and V2 Receivers



As previously mentioned, multicast routers can periodically send queries out of an interface to determine whether any multicast receivers still exist off that interface. However, you might have a situation in which more than one multicast router exists on a broadcast media segment (for instance, Ethernet). Therefore, one router must be designated as the "querier" for that segment. This IGMP-designated querier is the router that has the lowest unicast IP address.

To determine which router on a multiaccess network is the querier, issue the following command:

```
show ip igmp interface [interface-id]
```

The output from the preceding command identifies the IP address of the IGMP querier. In addition, the following command displays the IP multicast groups that a router is aware of:

```
show ip igmp group
```

When a Layer 2 switch receives a multicast frame on an interface, by default the switch floods the frame out all other interfaces. To prevent this behavior, the switch needs awareness of what interfaces are connected to receivers for specific multicast groups. Approaches for training the switch include the following:

- **Cisco Group Management Protocol (CGMP):** A Cisco proprietary approach used on lower-end switches that enables a Cisco router to tell a Cisco switch which of its interfaces are connected to multicast receivers for specific multicast groups.

 Please see page 245 for more details.

- **IGMP snooping:** Used on higher-end switches; enables a switch to autonomously determine which interfaces are connected to receivers for specific multicast groups by eavesdropping on the IGMP traffic exchanged between clients and routers.
- **GARP Multicast Registration Protocol (GMRP):** A standards-based approach for letting a receiver proactively inform its upstream switch that the receiver wants to belong to a specific multicast group. Here GARP is not Gratuitous Arp, but Generic Attribute Registration Protocol.
- **Router-Port Group Management Protocol (RGMP):** A proprietary approach that enables a switch to send IP multicast packets to only multicast-enabled routers that want to receive traffic for specific IP multicast groups.

Addressing

In a multicast network, the source sends multicast packets with a Class D destination address. The 224.0.0.0 through 239.255.255.255 address range is the Class D address range, because the first 4 bits in the first octet of a Class D address are 1110.

Some ranges of addresses in the Class D address space are dedicated for special purposes:

224.0.0.0 to 224.0.0.255 (Reserved link-local addresses)
224.0.1.0 to 238.255.255.255 (Globally scoped addresses)
232.0.0.0 to 232.255.255.255 (Source-specific multicast addresses)
233.0.0.0 to 233.255.255.255 (GLOP addresses)
239.0.0.0 to 239.255.255.255 (Administratively scoped addresses)

- **Reserved link-local addresses:** Used, for example, by many network protocols. Open Shortest Path First (OSPF) uses 224.0.0.5 and 224.0.0.6. RIPv2 uses 224.0.0.9, and Enhanced Interior Gateway Routing Protocol (EIGRP) uses 224.0.0.10. Other well-known addresses in this range include 224.0.0.1, which addresses all multicast hosts, and 224.0.0.2, which addresses all multicast routers.
- **Globally scoped addresses:** Used for general-purpose multicast applications. Can extend beyond the local autonomous system (AS).

 Please see page 245 for more details.

- **Source-specific multicast (SSM) addresses:** Usedwith IGMPv3 to enable a multicast receiver request, not only for membership in a group, but also to request specific sources to receive traffic from. Therefore, in an SSM environment, multiple sources with different content can all be sending to the same multicast destination address.
- **GLOP addresses:** Provide a globally unique multicast address range based on AS numbers. For example, if a company had an AS number of 65000, its globally unique range of multicast IP addresses would be 233.253.232.0 to 233.253.232.255. The AS number calculates the second and third octets in this address range. First, convert the AS number to hexadecimal. (That is, 65000 in decimal equals FD-E8 in hexadecimal.) FD in hexadecimal equals 253 in decimal, and E8 in hexadecimal equals 232 in decimal. The first octet of a GLOP address is always 233.
- **Limited-scope addresses:** Used for internal multicast applications (that is, traffic that doesn't leave the AS), much like the RFC 1918 address space is a "rivate address space.

In addition to Layer 3 addresses, multicast applications must also have Layer 2 addresses (that is, MAC addresses). Fortunately, these Layer 2 addresses can be constructed directly from the Layer 3 multicast addresses. A MAC address is a 48-bit address, and the first half (24 bits) of a multicast MAC address (in hex) is 01-00-5e. The twenty-fifth bit is always 0. The last 23 bits of the multicast MAC address come directly from the last 23 bits of the multicast IP address. Consider the following examples:

- Given a multicast IP address of 224.1.10.10, calculate the corresponding multicast MAC address. First, convert the last three octets to binary:

  ```
  0000.0001.0000.1010.0000.1010
  ```

  If the leftmost bit is not already 0, it should be changed to 0, because the twenty-fifth bit of a multicast MAC address is always 0:

  ```
  0000.0001.0000.1010.0000.1010
  ```

  Convert each nibble (that is, 4-bit section) into its hexadecimal equivalent:

  ```
  01-0a-0a
  ```

  Prepend 01–00–5e to the calculated address to produce the multicast MAC address:

  ```
  01-00[nd]5e-01-0a-0a
  ```

 Please see page 245 for more details.

- Given a multicast IP address of 224.129.10.10, calculate the corresponding multicast MAC address. First, convert the last three octets to binary:

  `1000.0001.0000.1010.0000.1010`

  If the left-most bit isn't already 0, it should be changed to a 0, because the twenty-fifth bit of a multicast MAC address is always 0:

  `0000.0001.0000.1010.0000.1010`

  Convert each nibble (that is, 4-bit section) into its hexadecimal equivalent:

  01–0a–0a

  Prepend 01–00–5e to the calculated address to produce the multicast MAC address:

  01–00–5e–01–0a–0a

Notice that both Layer 3 IP addresses translate into the same Layer 2 MAC address. This overlap permits 32 Layer 3 multicast addresses to map to the same Layer 2 multicast MAC address. So, care must be taken when selecting Layer 3 multicast addresses to avoid this overlap.

Distribution Trees

To combat the issue of receiving duplicate packets, Cisco routers perform a Reverse Path Forwarding (RPF) check to determine whether a multicast packet enters a router on the correct interface. An RPF check examines the source address of an incoming packet and checks it against the router's unicast routing table to see what interface should be used to get back to the source network. If the incoming multicast packet is using that interface, the RPF check passes, and the packet is forwarded. If the multicast packet comes in a different interface, the RPF check fails, and the packet is discarded.

 Please see page 245 for more details.

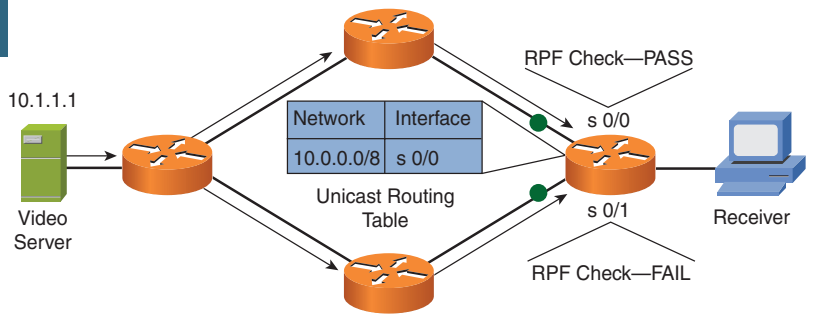


FIGURE 8-3
RPF Check

The RPF check compares incoming packets with the unicast routing table to determine if a packet is arriving on the correct interface.

Only members of a multicast group receive packets destined for that group; however, the sender does not need to be a member of the group.

Multicast traffic flows from a source to a destination over a distribution tree, which is a loop-free path. The two types of distribution trees are as follows:

- **Source distribution tree:** Creates an optimal path between each source router and each last-hop router (that is, a router connected to a receiver) at the expense of increased memory usage. Source distributions trees place (S, G) states in a router's multicast routing table to indicate the address of the source (S) and the address of the group (G).

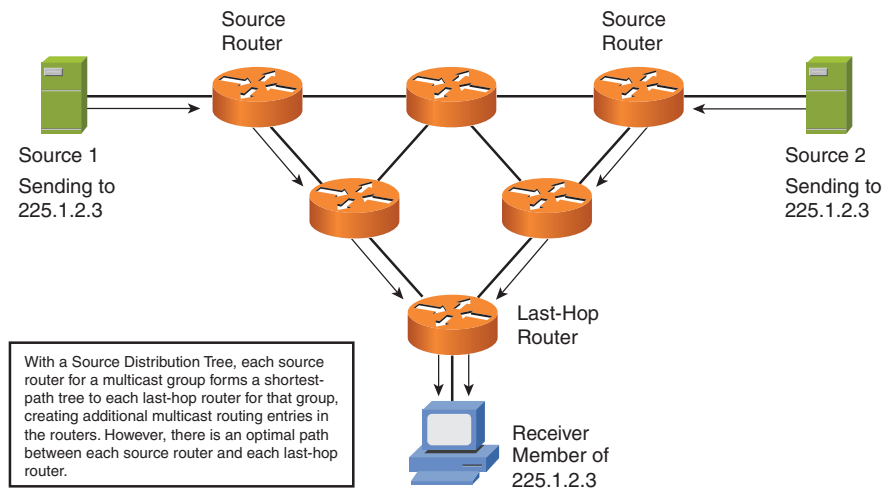


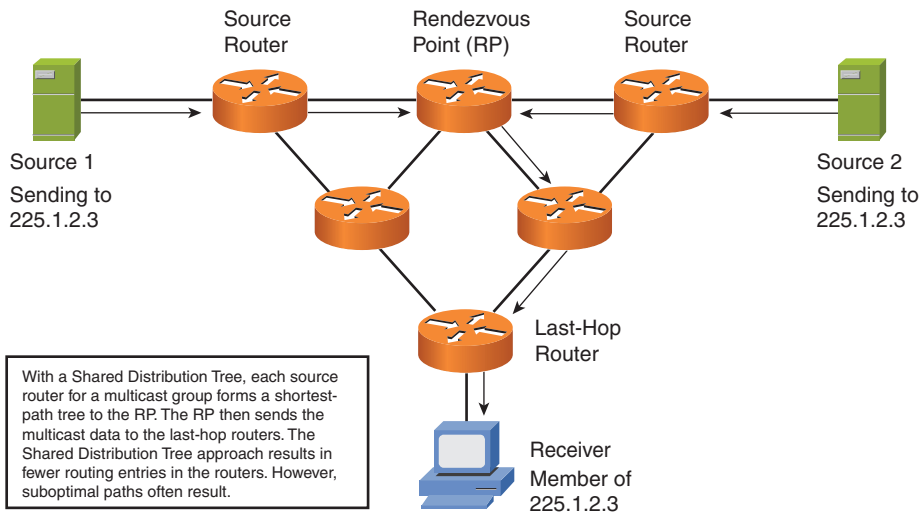
FIGURE 8-4
Source Distribution Tree

With a Source Distribution Tree, each source router for a multicast group forms a shortest-path tree to each last-hop router for that group, creating additional multicast routing entries in the routers. However, there is an optimal path between each source router and each last-hop router.

 Please see page 245 for more details.

- **Shared distribution tree**: Creates a tree from a central rendezvous point (RP) router to all last-hop routers, with source distribution trees created from all sources to the RP, at the expense of increased delay. Shared distribution trees place (*, G) states in a router's multicast routing table to indicate that any device could be the source (that is, using the wildcard [*] character) for the group (G). This (*, G) state is created in routers along the shared tree from the RP to the last-hop routers. Because each source for a group does not require its own (S, G), the memory requirement is less for a shared tree compared to a source tree.

FIGURE 8-5
Shared Distribution Tree



Protocol Independent Multicast (PIM) Dense Mode

PIM Dense Mode is the simplest method of multicast distribution. It is used when the majority of the hosts on a network are interested in receiving the multicast stream. Dense mode operates by flooding the data on the networks, making the assumption that all nodes are interested. If segments of the network do not want to receive the stream, the traffic is pruned back. As a result, dense mode multicast operations are sometimes referred to as flood and prune.

PIM Dense mode operates using the concepts of source distribution trees. Because the data is flooded out from the source outward, the multicast tree is rooted at the stream source.

To configure a router interface to use multicast dense mode

```
Router(config-int)# ip pim dense-mode
```

 Please see page 245 for more details.

PIM-DM Mechanics

Cisco routers use the Protocol-Independent Multicast (PIM) protocol to construct IP multicast distribution trees. PIM's protocol independence suggests that it can run over an IP network, regardless of the underlying unicast routing protocol, such as OSPF or EIGRP. The two varieties of PIM are PIM-Dense Mode (PIM-DM) and PIM-Sparse Mode (PIM-SM). PIM-DM uses a source distribution tree, whereas PIM-SM uses a shared distribution tree.

A router is globally enabled for multicast routing with the following global configuration mode command:

```
Router(config)#ip multicast-routing
```

After IP multicast has been globally enabled, individual interfaces need to be configured for PIM support. To configure an interface to participate in an IP multicast network using PIM, issue the following interface configuration mode command:

```
Router(config-if)#ip pim {dense-mode | sparse-mode | sparse-dense-mode}
```

Cisco recommends **sparse-dense-mode**, which uses Dense Mode to automatically learn the location of an RP, after which the interface runs in Sparse Mode. First, consider the formation of a PIM-Dense Mode distribution tree:

Step 1. A multicast source comes up and begins flooding multicast traffic throughout the network.

Step 2. If more than one router is forwarding over a common broadcast medium (for example, an Ethernet link), Assert messages determine the PIM forwarder. The router with the better metric or (by default) the highest IP address wins the election.

Step 3. Some routers might not have multicast receivers for the group whose traffic is currently flooded. Those routers send a Prune message to their upstream router, requesting that their branch of the distribution tree be pruned. However, if another router is on the same broadcast medium as the router that sent the prune, and if that other router has IP multicast receivers attached, the Prune message is ignored. The Prune message is ignored because the router attached to IP multicast receivers sends a Join Override message.

Step 4. If a receiver comes up on a router previously pruned from the tree, that router can rejoin the tree by sending a Graft packet.

A major consideration for PIM-DM, however, is that this flood-and-prune behavior repeats every 3 minutes. Therefore, PIM-DM does not scale well. A better alternative is PIM-SM.

 Please see page 245 for more details.

Protocol Independent Multicast (PIM) Sparse Mode

Typically, sparse mode multicast is used when a small number of receiving hosts are on a subnet and especially if they spread over a number of networks. If the multicast stream is flooded onto a network and only a relatively small number of hosts are interested, wasted bandwidth results. When sparse mode is used, hosts must specifically request data from the multicast stream rather than automatically receiving it.

PIM Sparse Mode operates using the concepts of a shared distribution trees. Because hosts must request membership to a multicast stream, the routers need to know where to send the membership reports, thus the need for rendezvous points. The rendezvous point provides a point where the source can send the data for distribution to the requesting hosts, and routers can send membership requests.

To configure an interface on a router for sparse mode multicast:

```
Router(config-int)# ip pim sparse-mode
```

PIM-SM Mechanics

Next, consider the formation of a PIM-SM distribution tree:

Step 1. A receiver sends an IGMP Report message to its router indicating that it wants to participate in a particular multicast group. The receiver's router (that is, the last-hop router) sends a Join message to the RP, creating (*, G) state along a shared tree between the RP and the last-hop router.

Step 2. A source comes up and creates a source tree between its router (that is, the first-hop router) and the RP. (S, G) state is created in routers along this path. However, before the source tree is completely established, the source sends its multicast packets to the RP encapsulated inside unicast Register messages.

Step 3. After the RP receives the first multicast packet over the source tree, it sends a Register Stop message to the source, telling the source to stop sending the multicast traffic inside Register messages. Two trees now exist: a source tree from the first-hop router to the RP and a shared tree from the RP to the last-hop router. However, this might not be the optimal path.

Step 4. The last-hop router observes from where the multicast traffic is arriving, and the last-hop router sends a Join message directly to the first-hop router to form an optimal path (that is, a source path tree) between the source and the receiver.

 Please see page 245 for more details.

Step 5. Because the last-hop router no longer needs multicast traffic from the RP, because it is receiving the multicast traffic directly from the first-hop router, it sends an (S, G) RP-bit prune message to the RP, asking the RP to stop sending multicast traffic.

Step 6. With the shared tree to the last-hop router pruned, the RP no longer needs to receive multicast traffic from the first-hop router. So the RP sends an (S, G) Prune message to the first-hop router. At this point, traffic flows in an optimal path from the first-hop router to the last-hop router. The process of cutting over from the path via the RP to the direct path is called shortest path tree (SPT) switchover.

Comparing PIM-DM to PIM-SM suggests that PIM-SM offers the benefits of PIM-DM (that is, optimal pathing) without PIM-DM's flood-and-prune behavior.

A distribution tree's topology can be determined by examining the multicast routing table of multicast routers in the topology. The **`show ip mroute`** command displays a router's multicast routing table:

```
Router#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group,
 s - SSM Group, C - Connected, L - Local,
 P - Pruned, R - RP-bit set, F - Register flag,
 T - SPT-bit set, J - Join SPT,
 M - MSDP created entry,
 X - Proxy Join Timer Running,
 A - Candidate for MSDP Advertisement,
 U - URD,
 I - Received Source Specific Host Report,
 Z - Multicast Tunnel,
 Y - Joined MDT-data group,
 y - Sending to MDT-data group
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD,
 State/Mode
```

 Please see page 245 for more details.

```
(*, 224.0.100.4), 02:37:12, RP is 192.168.47.14,
 flags: S
  Incoming interface: Serial0, RPF neighbor
 10.4.53.4
  Outgoing interface list:
    Ethernet1, Forward/Sparse, 02:37:12/0:03:42
    Ethernet2, Forward/Sparse, 02:52:12/0:01:23

(192.168.46.0/24, 224.0.100.4), 02:37:12,
 flags: RT
  Incoming interface: Ethernet1, RPF neighbor
 10.4.53.4
  Outgoing interface list:
    Ethernet2, Forward/Sparse, 02:44:21/0:01:47
```

Notice the (*, G) and (S, G) entries. Other valuable information contained in the mroute table includes the Incoming Interface List (IIF), which shows on which interface traffic is entering the router, and the Outgoing Interface List (OIL), which shows the router interfaces over which the multicast traffic is forwarded.

Rendezvous Points

In a PIM-SM network, one or more routers need to be designated as RPs. These routers are the central point to which multicast servers send traffic to be dispersed to clients who want to receive it. Non-RPs can be configured to point to a statically defined RP with the global configuration mode command **ip pim rp-address ip-address**. However, in larger topologies, Cisco recommends that RPs be automatically configured.

Auto-RP

Cisco routers support two methods for automatically configuring an RP: Auto-RP and Bootstrap Router (BSR). Routers that serve as an RP are called candidate RPs, and they make their candidacy known to other routers called mapping agents using the multicast address 224.0.1.39. A mapping agent then makes the location of an RP known to other multicast routers in the network using the multicast address 224.0.1.40. By default, the mapping agent advertises the candidate RP with the highest IP address.

 Please see page 245 for more details.

The global configuration command **`ip pim send-rp-announce interface scope ttl [group-list acl]`** is issued on candidate RPs. To identify a router as a mapping agent, use the global configuration mode command **`ip pim send-rp-discovery scope ttl`**.

Whereas Auto-RP is a Cisco approach, PIMv2 added a standards-based approach to make the location of RPs known throughout the multicast network. Specifically, PIMv2, which uses protocol 103, supports a feature called BSR, which performs a similar function to Auto-RP. Routers that are candidates to become the RP can be configured with the global configuration mode command **`ip pim rp-candidate interface ttl group-list acl`**. Routers that are candidates to become the bootstrap router (similar to an Auto-RP mapping agent) can be configured with the global configuration mode command **`ip pim bsr-candidate interface hash-mask-length [priority]`**. Because BSR leverages PIM messages, reserved multicast group addresses (for example, 224.0.1.40, used by Auto-RP) are not required for RP advertisement.

Anycast RP

Anycast RP provides load sharing and redundancy in PIM-SM networks. This technology enables multiple RPs to load-share and act as hot backup routers for each other. Multicast Source Discovery Protocol (MSDP) makes Anycast RP possible.

In Anycast RP, two or more RPs are configured with the same IP address and 32-bit mask on loopback interfaces. All the downstream routers are configured with this address as the RP address. IP routing automatically selects the topologically closest RP for each source and receiver.

Because sources can register with one RP and receivers can join a different RP, a method is needed for the RPs to exchange information about active sources. This information exchange is done using MSDP.

In Anycast RP, all the RPs are configured to be MSDP peers with each other. When a source registers with one RP, an SA message is sent to the other RPs, informing them that an active source exists for a particular multicast group. The result is that each RP knows about the active sources in the area of the other RPs. If any of the RPs were to fail, IP routing would converge, and one of the RPs would become the active RP in more than one area.

 Please see page 245 for more details.

Implement PIM Rendezvous Points Bootstrap Router (BSR)

When PIM sparse mode is employed, a rendezvous point must be configured in each router. Because static configuration is labor-intensive and not scalable, Cisco created the Auto-RP protocol to dynamically discover the RP within a multicast network. Bootstrap Router (BSR), part of PIMv2, was created as a similiar, nonpropritary protocol that serves the same purpose. BSR works using a couple of different components including the candidate RPs and the bootstrap router. (There can also be candidate bootstrap routers in a complex configuration.) To inform each multicast router within the network of the proper mappings, the bootstrap router advertises all the known group-to-RP mappings that it has learned from the candidate RPs. Based on a specific algorithm, the multicast routers within the network chooses the RP for a group from the advertisements from the bootstrap router.

The candidate RPs (c-RP) send their willingness to serve as the RP for a specific group to the bootstrap router. This is done using a unicast connection from the c-RP to the BSR. To solve the chicken-and-egg problem of how do the c-RP routers, know where the BSR is without a domainwide static configuration, the BSR advertise their existance using a special flooding algorithm within PIMv2.

To configure a BSR, use the **ip pim bsr-candidate** command:

```
Router(config)# ip pim bsr-candidate [interval seconds] [priority value]
```

After this command, the router advertises to be a BSR using PIMv2 bootstrap messages. This is treated special by PIMv2 so the messages are transmitted even in sparse mode. The router with the highest priority wins. If a priority tie occurs, the highest IP address wins.

To configure a candidate RP, use the **ip pim rp-candidate** command :

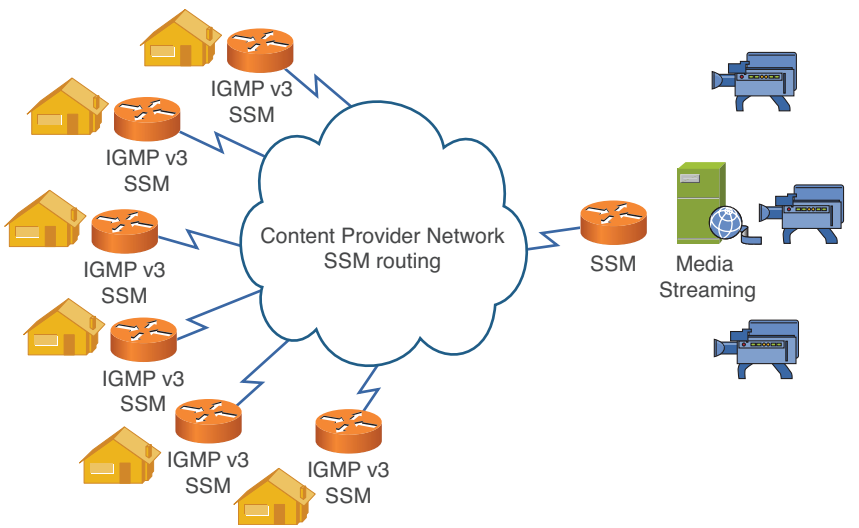
```
Router(config)# ip pim rp-candidate <INTERFACE> [group-list access-list] [interval seconds] [priority value]
```

The group-list option can advertise the candidancy for an RP for specific multicast groups as indicated by the access list. The priority influences the routers that decide which c-RP to use for specific groups. The BSRs then advertise the group-to-RP mappings to all routers using the multicast group of 224.0.0.13.

 Please see page 245 for more details.

Implement Interdomain Multicast Routing
Implementing Multicast Features
Source Specific Multicast

Imagine a host interested in listening to a specific multicast stream. In this particular stream, a multitude of uninteresting senders and one interesting one exist. It is useful to join a stream for a particular sender rather than any sender. This is the essence of source specific multicast (SSM), which provides a mechanism of protection from denial-of-service (DoS) attacks in which a flood of data is sent by an attacker to a particular multicast address. Care must be taken to ensure that any IGMP snooping or CMGP is supported if used. SSM support is part of the IGMPv3 (also the IGMP v3lite or URD) protocol; support is not available in IGMPv1 or IGMPv2. IGMPv3 provides a mechanism that a host can specify the (S,G) pair during the join process. PIM-SM can create a shortest path tree (SPT) to the source requested at the first-hop router.



The first-hop router needs to be configured to support SSM:

```
Router(config)# ip pim ssm
```

 Please see page 245 for more details.

All interfaces must be configured to support IGMPv3:

```
Router(config-if)# ip igmp version 3
```

SSM has been allocated its own multicast address range of 232.0.0.0/8. This range is used when SSM is implemented within the network.

Multicast Listener Discovery (MLD)

The advent of IPv6 has led to some changes to some IPv4 support protocols—one of which is IGMP, which is used for host multicast group management. The Multicast Listener Discovery (MLD) protocol was incorporated into IPv6 that provides the same functionality. Instead of being considered a completely separate protocol (with it's own IP protocol number), MLD uses the ICMP functionality within IPv6. MLDv1 contains the capability for hosts to join multicast groups through the Multicast Listener Report MLD message typeand query network segments for active listeners through the Multicast Listener Query message type, and the capability for hosts to notify the router that it not longer wants to listen to a particular multicast group through the Multicast Listener Done message type. When an MLDv1 listener leaves a multicast group (using the Done MLD message), it must notify the IPv6 router. The router then must determine if more subscribers are on the segment or it can stop sending traffic. It must send a Query MLD message and wait for a response. This creates a leave latency of a few seconds. MLDv2 includes functionality from IGMPv3, primarily source specific capabilities. To support this a new message type, version 2 Message Report, was created. This enables a host to request membership in a (S,G) pair rather than only (*,G).

MLD supports many of the same options within a router that IGMP does such as query timeouts and intervals and the capability for a router to join a group as a listener.

To configure an IPv6 router interface as a multicast group member:

```
Router(config-if)# ipv6 mld join-group <GROUP ADDRESS>
```

To configure the MLD maximum response time advertised in queries:

```
Router(config-if)# ipv6 mld query-max-reponse-time <SECONDS>
```

 Please see page 245 for more details.

To configure the MLD timeout before the router takes over querying responsibilities on the segment:

```
Router(config-if)# ipv6 mld query-timeout <SECONDS>
```

To configure the interval at which the MLD router sends queries on a segment:

```
Router(config-if)# ipv6 mld query-interval <SECONDS>
```

 Please see page 245 for more details.

Chapter 9

Security

Access Lists / Extended IP Access Lists

Many types of access lists are available in Cisco IOS Software for many different protocols. See Table 9-1 for a complete list.

Table 9-1 Cisco IOS Software Access Lists

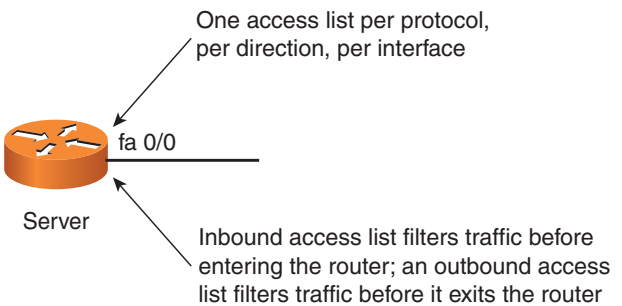
Protocol	Range
IP	1—99, 1300—1999
Extended IP	100—199, 2000—2699
Ethernet type code	200—299
Ethernet address	700—799
Transparent bridging (protocol type)	200—299
Transparent bridging (vendor code)	700—799
Extended transparent bridging	1100—1199
DECnet and extended DECnet	300—399
Xerox Network Systems (XNS)	400—499
Extended XNS	500—599
AppleTalk	600—699
Source-route bridging (protocol type)	200—299
Source-route bridging (vendor code)	700—799
IPX	800—899

 Please see page 245 for more details.

Extended IPX	900—999
IPX SAP	1000—1099
Standard Virtual Integrated Network Service (VINES)	1—100
Extended VINES	101—200
Simple VINES	201—300

You are permitted one access list per protocol, per interface, per direction.

FIGURE 9-1
Access Control Lists



At the end of every access list is an implied deny-all-traffic access control entry (ACE). Therefore, if a packet does not match any of your criteria statements, it is blocked.

Remember that the order of access list statements is important! For example, if you create a criteria statement that explicitly permits all traffic, no statements added later are ever checked.

When you edit an access list and need to reorder entries, you should first delete the old list with the **no access-list** command. If you do not first delete the previous version of the access list, when you copy or type commands on your router, you append additional access control list (ACL) statements to the end of the existing access list.

The following ACLs are supported for IP:

- Standard access lists for filtering based on source address
- Extended access lists for filtering on source or destination address or port numbers

 Please see page 245 for more details.

- Dynamic extended IP access lists that grant access per user to a specific source or destination host basis through a user authentication process
- Reflexive access lists that enable IP packets to be filtered based on session information

To create a standard access list, use the following global configuration mode syntax:

```
access-list access-list-number {deny | permit} source [source-wildcard] [log]
```

The Cisco IOS Software can provide logging messages about packets permitted or denied by a standard IP access list. The first packet that triggers the access list causes an immediate logging message, and subsequent packets are collected over 5-minute intervals before they are displayed or logged. You can use the **ip access-list log-update** command to set the number of packets that cause the system to generate a log message. If you enable Cisco Express Forwarding (CEF) and then create an access list that uses the **log** keyword, the packets that match the access list are not CEF switched.

To create an extended access list, use the following global configuration mode command:

```
access-list access-list-number {deny | permit} protocol source source-wildcard destination destination-
wildcard [precedence precedence] [tos tos] [established] [log | log-input] [time-range time-range-name]
[fragments]
```

You can identify IP access lists with a name rather than a number. To create a standard access list, use the following command:

```
ip access-list standard name
```

To create an extended access list, use the following command:

```
ip access-list extended name
```

You can specify whether the system examines noninitial IP fragments of packets when applying an IP extended access list. Before this option was added, nonfragmented packets and the initial fragment of a packet were processed by IP extended access lists, but noninitial fragments were permitted by default. The IP Extended Access Lists with Fragment Control feature enable more granularity of control over noninitial packets.

 Please see page 245 for more details.

The optional **fragments** keyword is available with four IP access list commands (**access-list [IP extended]**, **deny [IP]**, **dynamic**, and **permit [IP]**). By specifying the **fragments** keyword in an access list entry, that particular access list entry applies only to noninitial fragments of packets; the fragment is either permitted or denied accordingly.

The Turbo Access Control Lists (Turbo ACL) feature processes access lists more expediently than conventional access lists.

To enable the Turbo ACL feature, use the following command:

```
access-list compiled
```

Use the **show access-list compiled** EXEC command to verify that the Turbo ACL feature has been successfully configured on your router.

You can implement access lists based on the time of day and week using the **time-range global configuration** command. To do so, first define the name and times of the day and week of the time range and then reference the time range by name in an access list to apply restrictions to the access list.

To restrict access to a vty and the addresses in an access list, use the following command:

```
access-class access-list-number {in | out}
```

To restrict access to an interface, use the following command:

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

Zone-Based Firewall

Traditionally, access control on a Cisco router is controlled through (ACLs. These ACLs would be configured as a list of access control entries (ACE), which spelled out exactly what traffic was permitted or denied. Furthermore, most ACLs would list a series of traffic that was allowed in an interface and deny everything else. They were often limited to IPs and ports. The Cisco Zone-Based Firewall abstracted the concept by creating zones. A zone is a group of interfaces that have a common security policy. Networks are broken into zones, and a policy is configured between the zones to enable traffic. Following are the three basic steps in configuring a zone based firewall:

 Please see page 245 for more details.

1. Create the security zones
2. Create the policies between the zones
3. Assign interfaces to the zones

The first step is to configure a zone:

```
Router(config)# zone security <NAME>
```

Of these three steps, the creation of the policies is by far the most complicated and requires the most planning.

To configure a policy between the zones, the rules must be developed. Zone-based firewalls take advantage of the flexible policy/class map-based structure common in QoS. To create an inspect policy map:

```
Router(config)# policy-map type inspect <POLICY MAP NAME>
```

The policy map is filled with references to class-maps that define the traffic of interest:

```
Router(config-pmap)# class-type inspect <CLASS MAP NAME>
```

The specific policy is then applied to the traffic matched within the class-map. Following are the actions that can be taken:

* **inspect:** Traffic is enabled and inspected at the application layer using stateful inspection.
* **drop:** Traffic is dropped.
* **pass:** Traffic is permitted in one way but not entered in the state table.
* **police: T**raffic is policed using options. Must be used with inspect or pass options.

```
Router(config-pmap)# inspect|pass|drop|police
```

To create the class maps that match traffic using access-lists (**access-group**), individual protocols (protocol) or other class-maps (**class-map**):

```
Router(config)# class-map type inspect <CLASS MAP NAME>
Router(config-cmap)# match access-group | protocol | class-map
```

 Please see page 245 for more details.

To assign an interface to a zone

```
Router(config-if)# zone-member security <ZONE NAME>
```

The policy map must be applied to the zones:

```
Router(config-)# service-policy type inspect <POLICY MAP NAME>
```

Unicast Reverse Path Forwarding

The Unicast Reverse Path Forwarding feature (Unicast RPF) helps the network guard against malformed or "spoofed" IP packets passing through a router. A spoofed IP address is one that is manipulated to have a forged IP source address. Unicast RPF enables the administrator to drop packets that lack a verifiable source IP address at the router. Note how similar this is to the Reverse Path Forwarding check with multicast traffic. In that case, traffic was dropped to avoid loops.

Unicast RPF is enabled on a router interface. When this feature is enabled, the router checks packets that arrive inbound on the interface to see whether the source address matches the receiving interface. Cisco Express Forwarding (CEF) is required on the router because the Forwarding Information Base (FIB) is the mechanism checked for the interface match.

Administrators can decide to drop packets that arrive on an interface without a return path to the source in the FIB, or they can just have counters increment in the global IP traffic statistics for Unicast RPF drops and in the interface statistics for Unicast RPF.

Whether packets that fail the Unicast RPF check are dropped is controlled by the use of an ACL with the **ip verify unicast reverse-path** command. If an ACL is specified in the command and a packet fails the Unicast RPF check, the ACL is checked to see whether the packet should be dropped or forwarded. If no ACL is specified in the Unicast RPF command, the router drops the unverifiable packet, and the counters are updated. You can use ACL logging to obtain the source address information. Just be sure to specify the **log** option in the ACL used with the **ip verify unicast reverse-path** command.

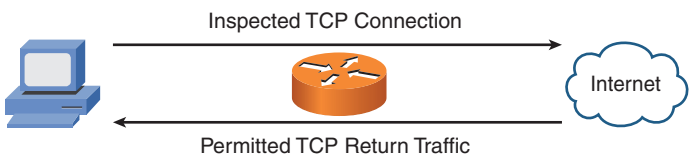
To implement Unicast RPF, ensure that CEF is enabled on the router, and use the **`ip verify unicast reverse-path list interface`** configuration command.

 Please see page 245 for more details.

Context-Based Access Control

Context-Based Access Control (CBAC) makes firewall-like stateful packet filtering a possibility on your Cisco IOS router (see Figure 9-2). This capability makes the Cisco IOS router act much like a Cisco PIX or Adaptive Security Appliance. Using CBAC, the router can permit TCP and User Datagram Protocol (UDP) connections from the "trusted" inside interface of the network to "untrusted" outside interfaces (for example, an Internet connection to an Internet service provider). The router then creates a stateful session table to monitor for the appropriate return traffic for these TCP and UDP sessions. Stateful packet filtering is much more powerful than traditional firewall packet filtering in that it can examine application layer information to ensure traffic is safe for entrance into the network. Traditional filtering was often limited to source address inspection, for example.

FIGURE 9-2
CBAC



CBAC functions on a router as follows:

- Control traffic is inspected by an administrator-configured CBAC rule (for example, **`ip inspect name MYCBACRULE tcp`**).
- CBAC creates a dynamic ACL enabling return traffic through the router.
- Inspection continues with dynamic ACLs being created and removed as needed; application-specific attacks are also monitored for.
- Application termination is detected, or timeouts occur and dynamic ACLs are removed.

 Please see page 245 for more details.

CBAC can be configured to support all TCP connections or all UDP sessions. You can also configure CBAC to inspect certain application layer protocols:

- FTP
- Simple Mail Transport Protocol (SMTP)
- HTTP
- ICMP
- Session Initiation Protocol (SIP)

Configuring CBAC on a router involves the following tasks:

- Determine whether CBAC will be configured on an internal or external interface.
- Ensure access lists configured for outbound traffic permit the CBAC-analyzed traffic, and ensure access lists configured for inbound traffic deny the CBAC-analyzed traffic.
- Configure global timeouts and thresholds:

  ```
  ip inspect tcp synwait-time seconds
  ip inspect tcp finwait-time seconds
  ```

- Define an inspection rule.

  For an application layer protocol:

  ```
  ip inspect name inspection-name protocol [alert {on | off}] [audit-trail {on | off}] [timeout seconds]
  ```

  For an RPC application layer protocol:

  ```
  ip inspect name inspection-name rpc program-number number [wait-time minutes] [alert {on | off}]
  [audit-trail {on | off}] [timeout seconds]
  ```

 Please see page 245 for more details.

For Java blocking:

```
ip inspect name inspection-name http [java-list access-list] [alert {on | off}] [audit-trail
{on | off}] [timeout seconds]
```

For TCP and UDP inspection:

```
ip inspect name inspection-name tcp [alert {on | off}] [audit-trail {on | off}] [timeout seconds]
ip inspect name inspection-name udp [alert {on | off}] [audit-trail {on | off}] [timeout seconds]
```

- Apply the inspection rule to an interface:

```
ip inspect inspection-name {in | out}
```

- Configure audit trail messages:

```
ip inspect audit-trail
```

LAN Security

Switch Port Security

You can use the port security feature to restrict input to an interface by limiting and identifying MAC addresses of the stations allowed to access the port.

You can configure these types of secure MAC addresses:

- **Static secure MAC addresses:** Manually configured by using the **switchport port-security mac-address** MAC address interface configuration command.
- **Dynamic secure MAC addresses:** Dynamically learned, stored only in the address table and removed when the switch restarts.
- **Sticky secure MAC addresses:** Dynamically learned or manually configured, stored in the address table, and added to the running configuration. These addresses can be saved in the configuration file.

 Please see page 245 for more details.

To enable sticky learning, enter the **switchport port-security mac-address sticky** interface configuration command.

You can configure the interface for one of three violation modes, based on the action to be taken if a violation occurs:

- **Protect:** Packets with unknown source addresses are dropped until you remove a sufficient number of secure MAC addresses or increase the number of maximum allowable addresses.
- **Restrict:** Packets with unknown source addresses are dropped until you remove a sufficient number of secure MAC addresses or increase the number of maximum allowable addresses; you are notified.
- **Shutdown:** Port security violation causes the interface to immediately become error-disabled and turns off the port LED; it also sends a Simple Network Management Protocol (SNMP) trap, logs a syslog message, and increments the violation counter.

The following interface configuration commands enable and configure port security:

```
switchport port-security
switchport port-security maximum value [vlan [vlan-list]]
switchport port-security violation {protect | restrict | shutdown}
switchport port-security mac-address mac-address [vlan vlan-id]
switchport port-security mac-address sticky
```

You can use port security aging to set the aging time for static and dynamic secure addresses on a port. Two types of aging are supported per port:

- **Absolute:** The secure addresses on the port are deleted after the specified aging time.
- **Inactivity:** The secure addresses on the port are deleted only if the secure addresses are inactive for the specified aging time.

```
switchport port-security aging {static | time time | type {absolute | inactivity}}
```

 Please see page 245 for more details.

IP Source Guard

IP Source Guard is a Catalyst security feature related to DHCP snooping. IP source guard helps prevent IP spoofing by enabling only the IP addresses obtained through DHCP snooping on a particular port. When a client receives an IP address from the authorized DHCP server, a port access control list (PACL) is installed on the port. This PACL enables traffic in the interface if it is sourced from the DHCP provided IP address.

The steps to configuring IP source guard follows:

Step 1. Configure DHCP snooping in global configuration mode:

```
ip dhcp snooping
```

Step 2. Enable DHCP snooping for the appropriate VLANs:

```
ip dhcp snooping vlan number
```

Step 3. Configure the trust state of the interface:

```
no ip dhcp snooping trust
```

Step 4. Configure the IP Source Guard feature:

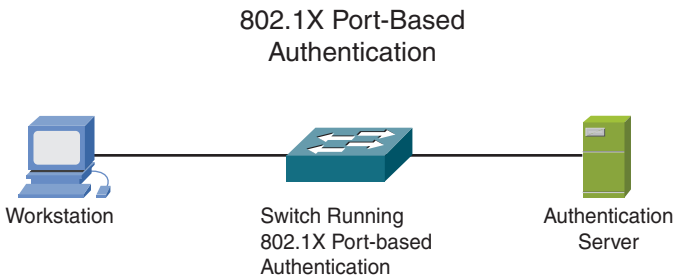
```
ip verify source vlan dhcp-snooping port-security
```

802.1X Port-Based Authentication

The IEEE 802.1X standard defines a client/server-based access control and authentication protocol that restricts unauthorized clients from connecting to a LAN through publicly accessible ports (see Figure 9-3). The authentication server authenticates each client connected to a switch port before making available any services offered by the switch or the LAN.

 Please see page 245 for more details.

FIGURE 9-3
802.1X Port-Based Authentication



You control the port authorization state by using the **dot1x port-control** interface configuration command and these keywords:

- **force-authorized:** Disables 802.1X authentication and causes the port to transition to the authorized state without any authentication exchange required
- **force-unauthorized:** Causes the port to remain in the unauthorized state, ignoring all attempts by the client to authenticate
- **auto:** Enables 802.1X authentication and causes the port to begin in the unauthorized state

802.1X port-based authentication is supported in two topologies:

- Point-to-point
- Wireless LAN

Use the following commands to enable 802.1X authentication. Note that dot1x authentication requires RADIUS as the method.

```
Switch(config)# aaa new-model
Switch(config)# aaa authentication dot1x {default} method1 [i...]
Switch(config)# dot1x system-auth-control
Switch(config-if)# dot1x port-control auto
```

 Please see page 245 for more details.

Device Security/Access

Remember, you will find no substitute for physical security of your Cisco devices. Not only can the devices be easily stolen, but also access to the console port allows passwords to be reset and security into the network to be breached. After ensuring your devices are physically secured, you should place passwords on the various operating modes of your device.

It is simple to set local passwords and security on your router or switch to help protect the operating modes and line access.

Use the following syntax to protect access to the console port with a local password:

```
CiscoDevice(config)# line console 0
CiscoDevice(config-line)# login
CiscoDevice(config-line)# password cisco
```

Note
The preceding passwords are stored in the configuration in plain text. To ensure that they are encrypted—along with all other plain-text passwords that might exist—use the **service password-encryption** command. This uses Cisco type 7 encryption, which is weak and easily crackable. For better security it is recommended that you use AAA with the username command for local users. These local user's passwords can be encrypted with MD5, which is not easily crackable.

Notice that the preceding command login permits the use of local password checking on the line. You can use the **no login** command to disable password checking.

The sample syntax protects the Telnet lines with a local password as follows:

```
CiscoDevice(config)# line vty 0 4
CiscoDevice(config-line)# login
CiscoDevice(config-line)# password cisco
```

For enacting local security, you can configure 16 different privilege levels, numbered 0 through 15. To configure a privilege level for users and associate commands with that privilege level, use the **privilege** command in global configuration mode. For example, to set the use of the **configure** command to level 14, use the following command:

```
privilege exec level 14 configure
```

To protect access to **privileged** mode, you can use the **enable password global configuration** command. You can specify a privilege level if you use various levels in your local security model. If no level is specified, the default level 15 is assumed. This privilege level provides full access to the privileged mode commands by default.

 Please see page 245 for more details.

For additional protection, use the **enable secret** command to set an encrypted privileged mode password. Again, you can use the **level** argument to assign the password to a particular privilege level.

It is a best practice to set both versions of the privileged mode password (enable password and enable secret), but you should set them to different values. If you attempt to set the passwords the same, you get a warning, but the password is still accepted. After you set a password using the **enable secret** command, a password set using the **enable password** command works only if the enable secret is disabled or an older version of Cisco IOS Software is used, such as when running an older rxboot image.

Also part of the local security model is the **username** command. It provides username and password authentication for login purposes only. Add a username entry for each remote system that the local router communicates with and requires authentication from (for example, Challenge Handshake Authentication Protocol [CHAP], used with PPP). The remote device must have a username entry for the local router. This entry must have the same password as the local router's entry for that remote device. You can also use this command to define usernames that get special treatment. For example, you can use this command to define a guest username that does not require a password but connects the user to a general-purpose information service.

Authentication, Authorization, and Accounting

You can also ensure security in the network through the use of AAA—authentication, authorization, and accounting.

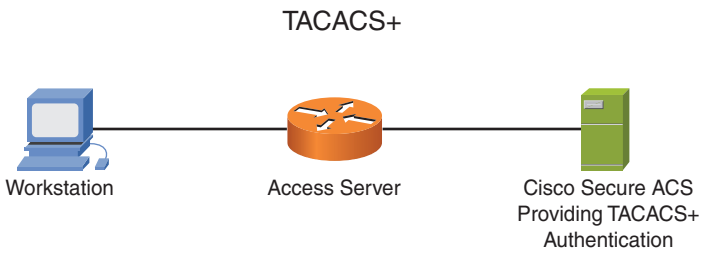
Authentication

Authentication can be accomplished using usernames and passwords configured locally on the switch, one or more RADIUS servers, or one or more TACACS+ servers.

You can and should configure multiple authentication sources. For example, if your TACACS+ servers are unavailable (an error is returned when access fails), you should have authentication seamlessly fail over to some other method—perhaps the local username and password database on the device.

 Please see page 245 for more details.

FIGURE 9-4
TACACS+



To configure authentication on a router or switch, complete the following steps:

Step 1. Enable AAA on the device using the **aaa new-model** command. This command permits the use of modern security protocols such as TACACS+, RADIUS, and Kerberos.

Step 2. Define the source of authentication. For example, you can use the **tacacs-server host** command to define the TACACS+ servers you are using for authentication. You can then use the **aaa group server tacacs+** command to group these servers.

Step 3. Define a list of authentication methods to try using the **aaa authentication login** command. If you specify TACACS+ servers first and you get no response from them (an error is returned), the next listed method is tried.

Step 4. Apply a method list to the router or switch line using the **login authentication** command.

Authorization

When authenticated, a user is placed in user EXEC mode by default. Configure authorization with the following steps:

Step 1. Configure the RADIUS or TACACS+ servers that contain the authorization database. These are typically already defined for you using Step 1 from the configuration of authentication.

Step 2. Define a method list of authorization methods to be tried in sequence using the **aaa authorization** command. In this command, you not only specify the authorization sources (for example, a group of TACACS+ servers), but you also specify the function or service needing authorization. This is done with one of the following keywords:

 Please see page 245 for more details.

Commands: The authorization server must return permission to use any command at any level.

config-commands: The server must return permission to use a configuration command.

Configuration: The server must return permission to enter configuration mode.

Exec: The server must return permission for the user to run an EXEC session.

Network: The server must return permission to use network-related services.

reverse-access: The server must return permission for a reverse Telnet session.

Step 3. Apply the authorization method list to a specific line on the device using the **authorization** command.

Accounting

The RADIUS and TACACS+ servers can also collect usage information for auditing or even billing purposes.

Step 1. Define the accounting servers; typically this is completed in Step 1 of the authentication process.

Step 2. Define a method list providing a sequence of accounting methods using the command **aaa accounting**. In this command, you specify functions that trigger accounting, for example:

System: Major events such as reload

Exec: User authentication into an EXEC session

Commands: Information about any executed commands

You can also specify that certain types of accounting records be sent:

start-stop: Events are recorded when they start and stop.

stop-only: Events are recorded when they stop.

None: No events are recorded.

Step 3. Apply the accounting method to a line on the device using the accounting command.

 Please see page 245 for more details.

RADIUS Versus TACACS

Understand the differences between these two security protocols, as outlined in Table 9-2.

Table 9-2 RADIUS Versus TACACS

	RADIUS	TACACS+
Transport protocol	User Datagram Protocol (UDP)	TCP
Encryption	Encrypts only the password	Encrypts entire body
AAA	Combines authentication and authorization	Separates AAA functions
Standards-based	Industry standard	Cisco proprietary

Control Plane Policing (CoPP)

The control plane of the router handles network aspects that are responsible for ensuring that the network is up and operational; it is the "how" a network is built. For example, routing protocol updates and ARP frames are handled by the control plane. Obviously, it is important the control plane is not compromised because without it, the network cannot run. Control Plane Policing (CoPP) helps to protect the network from flooding attacks against the control plane. CoPP uses a policy map to define which traffic is allowed to be permitted to the route process and at what levels. Not only can CoPP deny or allow based upon the type of traffic, it can rate limit the traffic so the route processor is not overwhelmed.

To configure CoPP, there are number of steps:

1. Define the ACLs that describe the control traffic on the network.
2. Create class-maps that group the ACLs together.
3. Create a policy map that permits, rate-limits, or denies the traffic.
4. Apply the policy map to the control plane.

The creation of the ACLs is highly dependent on the network. When created they can be placed in a class map:

```
Router(config)# class-map <CLASS MAP NAME>
```

 Please see page 245 for more details.

The policy map groups the class-maps and applies a policy to them:

```
Router(config)# policy-map <POLICY MAP NAME>
Router(config-pmap)# class <CLASS MAP>
Router(config-pmap)# drop|police
```

Apply the policy map to the control plane:

```
Router(config)# control-plane
Router(config-cp)# service-policy <POLICY MAP NAME>
```

IOS Intrusion Prevention System (IPS)

The Cisco IOS Intrusion Prevention System (IPS) is a feature that was built into IOS to complement the security features provided by the IOS Firewall. The IPS inspects that traffic as it flows through the router and matches it to a number of signatures in a file. If the IPS detects a match in the traffic to a signature, a number of actions can be performed by the IPS:

- The matching traffic can be allowed and an alert generated via syslog.
- The matching traffic flow can be reset.
- The matching packet can be dropped.
- The matching source IP of the traffic flow can be blocked for a period of time.
- The matching traffic flow can be blocked for a period of time.

The signature files are provided by Cisco and are updated periodically. These files can be uploaded into the router and applied on an interface, or the router can use the built-in signatures that are part of the IOS. To specify a location of the signature file (this is not needed if the default ruleset within IOS is used):

```
Router(config)# ip ips sdf location <LOCATION>
```

To create an IPS rule that is then applied to an interface

```
Router(config)# ip ips name <IPS NAME>
At this point the IPS can be applied to an interface:
```

 Please see page 245 for more details.

```
Router(config-if)# ip ips <IPS NAME> in
A single signature within the ruleset can be disabled by:
Router(config)# ip ips signature <#> disable
To view the IPS configuration within the CLI simply issue:
Router# show ip ips configuration
```

Secure Shell

When the infancy of the Internet, security was an afterthought. Secure Shell (SSH) is a replacement for telnet and should be used, without exception, on all Cisco devices. SSH strongly encrypts all management traffic between the device and the management workstation that keeps nefarious individuals from stealing passwords, configuration files, and running amok on the network. SSH comes in version 1 or version 2, both of which are supported in IOS. SSHv2 is the industry standard and should be used over SSHv1. SSH depends upon the creation of asymmetric keys within the device before it can be used.

To create the SSH keys within IOS, the crypto libraries must be accessed:

```
Router(config)# crypto key generate rsa general modulus <MODULUS>
```

Generally speaking, the larger the modulus, the more strong the keys. Typically, keys of 2048 bits are recommended.

After the keys have been generated, SSH will be enabled. To view which version of SSH is used

```
Router# show ip ssh
```

The version shows 1.5 if only version 1 is supported, 1.99 if version 1 and version 2 are supported, and version 2.0 if only version 2 is supported.

To change the version of the SSH server running on IOS

```
Router(config)# ip ssh version <VERSION>
```

Unfortunately, enabling SSH does not stop telnet from being available. To disable telnet, the vty lines must be accessed, and the **transport input** input command must be used to only enable **ssh**:

```
Router(config-line)# transport input ssh
```

Because telnet is not listed, it will no longer be supported.

 Please see page 245 for more details.

Chapter 10

MPLS

Multiprotocol Label Switching Overview

Multiprotocol Label Switching (MPLS) leverages the efficiency of Cisco Express Forwarding (CEF) and the intelligence provided by IP routing. CEF enables the creation of a copy of the Routing Information Base (RIB) in memory in the Cisco router or switch. This memory-based structure is called the Forwarding Information Base (FIB) and enables for remarkable packet forwarding times through what traditionally had been much slower devices.

MPLS appends a label to packets. This label can efficient forward decisions through an MPLS network. Using MPLS, the Layer 3 header information can be analyzed once as the packet enters the MPLS domain. After this single Layer 3 examination, a label can be appended that enables the subsequent MPLS devices to skip the traditional Layer 3 routing process. Although labels typically correspond to Layer 3 destination addresses, the labels can also correspond to QoS requirements, the source address, or a variety of other criteria.

Label Switch Router

A Label Switch Router (LSR) is the device that makes MPLS possible. These devices can be grouped into two categories:

- **Edge LSR:** Resides at the edge of the MPLS network and has many functions it is responsible for as a result. These functions include the following:

  Label distribution

  Packet forwarding based on labels

  Label imposition (insertion)

  Label disposition (removal)

- **LSR:** Does not reside at the edge of the MPLS network; therefore, it is only typically responsible for the following:

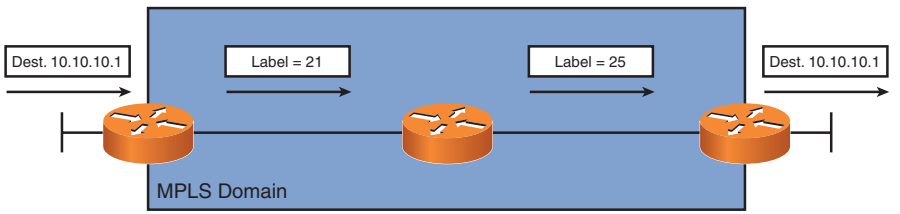
 Please see page 245 for more details.

Label distribution

Packet forwarding based on labels

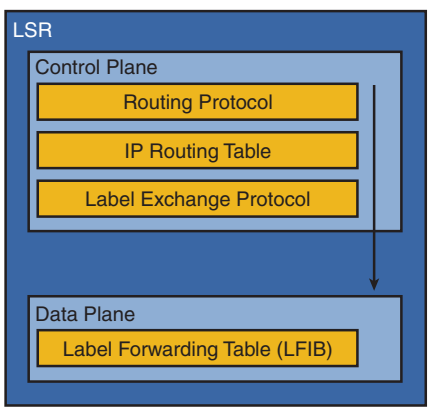
Service providers typically refer to an Edge LSR as a PE (Provider Edge) router and an LSR as a P (Provider) router, as shown in Figure 10-1.

FIGURE 10-1
LSRs



Examine the architecture of the LSR shown in Figure 10-2. Notice how the Label Forwarding Information Base (LFIB) in the data plane forwards labeled packets, and the control plane mechanisms build this LFIB.

FIGURE 10-2
Architecture of the LSR



Label Format

The MPLS header (often called the label) is 32 bits in length and contains a number of fields. The label identifies the destination and the services the packet will receive. This is called the Forwarding Equivalence Class (FEC). Labels are locally significant. Each LSR independently maps a label to a FEC. The LSRs then exchange these label bindings.

The 32-bit label field used by MPLS is shown in Figure 10-3.

 Please see page 245 for more details.

FIGURE 10-3
MPLS Label Format

Notice this label contains the following fields:

- **Twenty-bit label**
- **Three-bit experimental field:** Typically used to carry IP precedence or class of service
- **Bottom-of-Stack bit:** Determines whether the label is the last in the stack of labels
- **Eight-bit TTL field:** Prevents looping of packets

With Frame Mode MPLS, the label is inserted between the Layer 2 and Layer 3 header. With Cell Mode MPLS, the fields in the ATM header are used as the label.

Note the Bottom-of-Stack bit is required because some packets can have multiple labels. This can happen for the following reasons:

- **MPLS virtual private networks (VPN) (two labels):** One label is used to locate the egress router, and the second label is used to identify the VPN.
- **MPLS traffic engineering (two or more labels):** One label points to the endpoint of the tunnel, and the other label points to the destination.
- **MPLS VPNs used with MPLS traffic engineering (three or more labels)**.

Label Imposition/Disposition

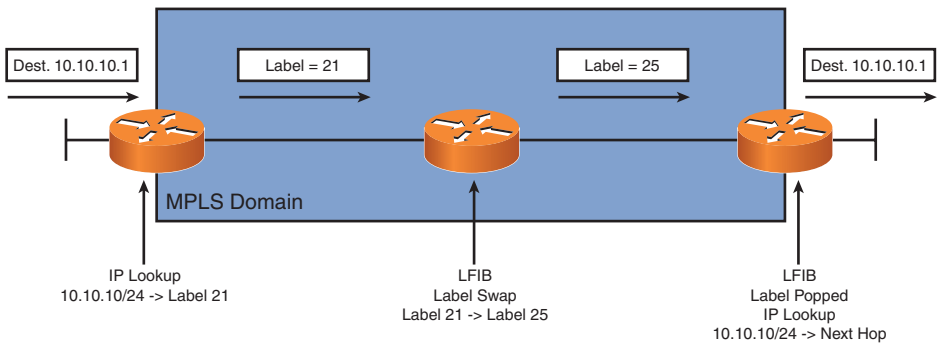
The LSR (or Edge LSR) performs one or more of the following functions:

 Please see page 245 for more details.

- **Ingress Edge LSR:** Inserts (imposes) the label or stack of labels.
- **Core (Interior) LSR:** Top label is swapped with the next-hop label or stack of labels.
- **Egress Edge LSR:** The label is removed (popped).

Figure 10-4 shows this process.

FIGURE 10-4
Label Imposition/ Disposition



Label Distribution

Label Distribution Protocol (LDP) exchanges labels between adjacent routers. LDP is session-based and has the following characteristics:

- Hello messages are sent periodically on MPLS-enabled interfaces; these messages initiate session establishment.
- Hello messages are UDP (port 646) sent to multicast 224.0.0.2 (all routers).
- TCP actually establishes the session (port 646).

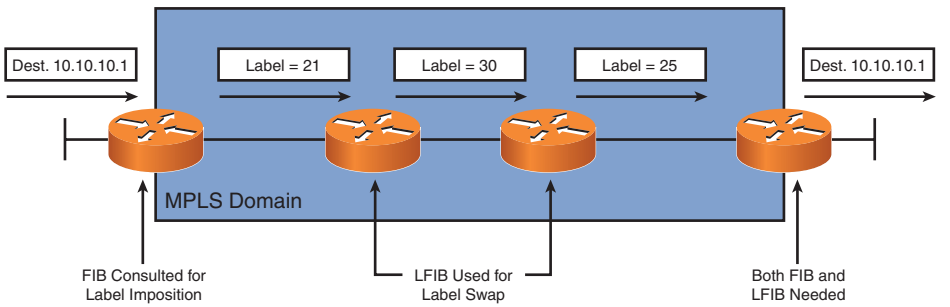
Label-Switched Path

The Label-Switched Path (LSP) is simply the sequence of LSRs that make up the Forwarding Equivalence Class (FEC) path. LSPs are unidirectional, which means that the return path might be different. However, routing protocols typically provide symmetric paths, so if MPLS is based on the routing table output, it is often symmetric, too.

 Please see page 245 for more details.

Penultimate Hop Popping (PHP) is used in the LSP to improve efficiency in the MPLS operations. Figure 10-5 shows the issue if PHP is not used in the MPLS network.

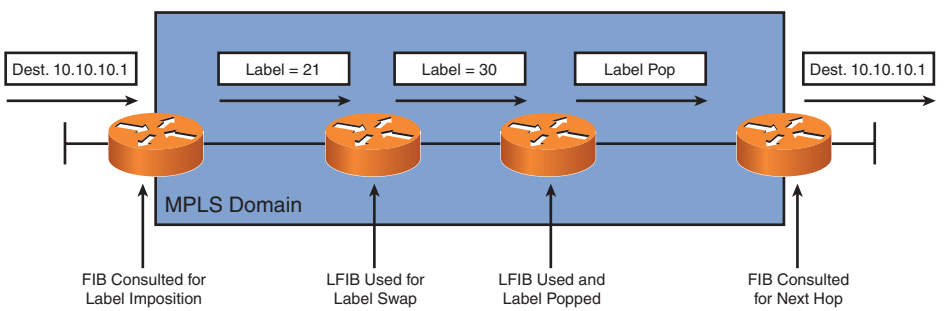
FIGURE 10-5
No PHP



Notice that both a FIB and an LFIB lookup are required on the egress router. This is because the LFIB must be consulted to learn that the label should be removed (popped), and the FIB needs to forward the packet to the next-hop IP address.

Figure 10-6 shows the use of PHP to increase efficiency.

FIGURE 10-6
PHP

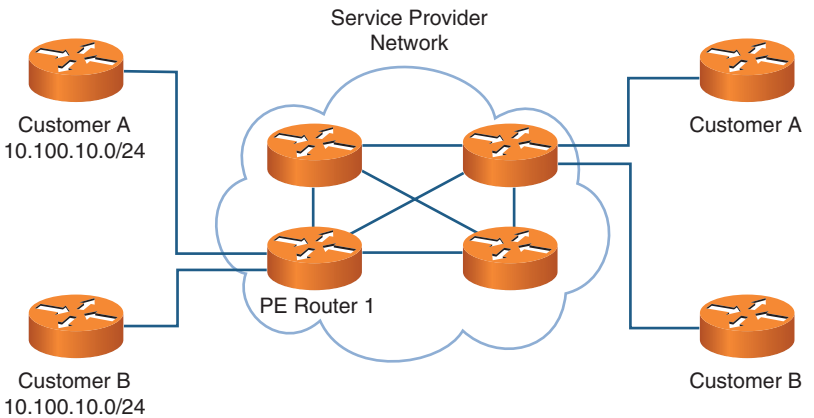


 Please see page 245 for more details.

Route Descriptor

The Route Descriptor (RD) is a key MPLS element that enables service providers to implement MPLS VPNs for customers. The following discussion is based on the sample MPLS VPN topology shown in Figure 10-7.

FIGURE 10-7
MPLS VPN Topology



On PE Router 1, a VPN Routing and Forwarding (VRF) table is configured for each customer (Customer A and Customer B). These VRF tables contain the routes advertised by each customer. What if each customer has the same prefix to advertise? In this example, notice that each customer wants to advertise the 10.100.10.0/24 prefix. The PE Router 1 handles this situation by prepending an RD to the IPv4 prefix to uniquely identify the prefixes as belonging to particular VPN customers. The VPN-IPv4 address is a combination of the IPv4 and RD.

The PE Router 1 device can propagate the prefix information to the other provider routers using Multiprotocol BGP, which is an option because it supports an extended community attribute field. This field carries a route target that enables VPNv4 addresses to be assigned to VRFs.

The VRF tables on PE Router 1 also contain an Export Target. This attribute determines which target PE routers receive the VPN-IPv4 address information. On the receiving PE routers, an Import Target value is set.

 Please see page 245 for more details.

Basic MPLS Configuration

Basic MPLS configuration consists of three mandatory and several optional tasks:

Mandatory:

Step 1. Use the appropriate command for your platform to enable CEF.

Examples include the global configuration command **ip cef** or the interface command **ip route-cache cef**.

Step 2. Enable label switching on a frame-mode interface:

```
mpls ip
```

Step 3. Start the appropriate label distribution protocol on the interface:

```
mpls label protocol [tdp | ldp | both]
```

Optional:

Step 1. Configure the MPLS ID on a router:

```
mpls ldp router-id interface
```

Step 2. Configure a label-switching MTU:

```
mpls mtu bytes
```

Step 3. Configure IP TTL propagation:

```
mpls ip propagate-ttl
```

Step 4. Configure conditional label distribution:

```
mpls ldp advertise-labels [for prefix-access-list [to peer-access-list]]
```

 Please see page 245 for more details.

Monitoring MPLS is possible because of the following commands:

- **show mpls ldp parameters:** Displays LDP parameters
- **show mpls interfaces:** Displays MPLS status on interfaces
- **show mpls ldp discovery:** Displays all discovered LDP neighbors
- **show mpls ldp neighbor:** Displays individual LDP neighbors
- **show mpls ldp neighbor detail:** Displays more details about an LDP neighbor
- **show mpls ldp bindings:** Displays the Label Information Base
- **show mpls forwarding-table:** Displays the contents of the LFIB
- **show ip cef detail:** Displays labels attached to a packet by the Edge LSR
- **debug mpls ldp:** Debugs LDP adjacencies
- **debug mpls lfib:** Debugs LFIB events
- **debug mpls packets:** Debugs labeled packets

Implementing MPLS Layer 3 VPNs

Implementing Multiprotocol Label Switching (MPLS) Label Switching Router

Implementing Layer 3 Virtual Private Networks (VPNs) on Provider Edge (PE) and Customer Edge (CE) Router

Layer 3 virtual private networks (VPN) are one of the most common uses of MPLS networks. Common VPN networks employ an overlay topology in which the virtual links are built on top the service provider network; the service provider is not involved with the customers internal routes. An MPLS VPN network uses a peer-to-peer topology in which the service provider learns the internal routes from the customer and builds a virtual network within the service provider core. MPLS takes advantage of a number of technologies to accomplish Layer 3 VPNs.

 Please see page 245 for more details.

To create an MPLS VPN infrastructure, an MPLS must be enabled. To enable MPLS, Cisco Express Forwarding (CEF) must be enabled:

```
Router(config)# ip cef
```

To enable MPLS on an interface, use the **mpls ip** command:

```
Router(config-int)# mpls ip
```

The customer edge (CE) routers peer with the provider edge (PE) routers using a routing protocol that supports Virtual Forwarding/Routing (VRF), such as RIPv2, EIGRP, or OSPF. (The following section provides details of implementation.) BGP can also be implemented by using MP-BGP extensions. The PE router can create VRF for each connected customer that enables the same subnet to exist if they are in different VRFs. Each MPLS edge router now contains routes from customers that need to connect across the service provider using MP-BGP. For example, Customer A has a local site that might want to share data with another remote, site across the service provider cloud. The remote site's CE router peers with the PE router. To create a VPN between Customer A's sites, the two PE routers must do the following:

Step 1. Share the routing information found in Customer A's VRF on the PE router.

Step 2. Keep the VRF information private so that another customer does not have access to Customer A's information.

Step 3. Ensure that Customer A's routes are not imported into another customer's VRF.

To accomplish these goals, BGP employs route targets within the VRFs to import and export routes from one router to another. To export the routes from MP-BGP within a VRF, use the **route-target export** command:

```
Router(config-vrf)# route-target export <ASN:nn or IP:nn>
```

The target that is specified can take the form of AS number:nn in which nn is a 32-bit number or IP address:nn in which nn is a 16-bit number. The routes are then exported out of the VRF and sent to all the iBGP peers using VPNv4 routes. To import the routes into a VRF on the other end of a VPN, use the **route-target import** command:

```
Router(config-vrf)# route-target import <ASN:nn or IP:nn>
```

To support the VPNv4 routes within BGP, the iBGP neighbors must be activated within the vpnv4 address family:

 Please see page 245 for more details.

```
Router(config-router)# address-family vpnv4
Router(config-router-af)# neighbor <IP> activate
```

Then the extended communities must be used:

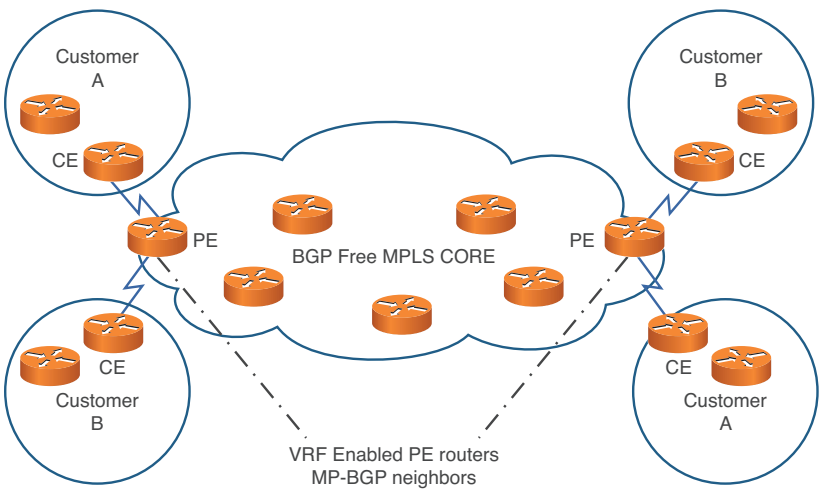
```
Router(config-router-af)# neighbor <IP> send-community both
```

The final configuration step in MPLS VPNs is to configure the IPv4 address family for each VRF:

```
Router(config-router)# address-family ipv4 vrf <VRF NAME>
```

The routing protocol used to learn routes from the customer network must be redistributed into BGP using the **redistribute** command.

Each BGP speaking router that advertises a network to its peers assigns a label to that destination. When a BGP router wants to forward a packet toward a destination that has been advertised, the associated label is used as an inner label through the MPLS network. The outer label then transports the packet from BGP router to BGP router. The inner label serves to indicate to the egress BGP router the VRF that the packet belongs to.



 Please see page 245 for more details.

Implement Virtual Routing and Forwarding (VRF) and Multi-VRF Customer Edge (VRF-Lite)

A routing table within a router contains a list of the networks known to be reachable through the router's interfaces. By default, the router places all the learned networks into one global routing table.

A virtual routing and forwarding (VRF) table is a virtual routing table that contains a configured subset of the interfaces on the router. The network reachability information is entered only in the routing table or VRF that the interface,through which the route was learned, belongs to. This enables an engineer to isolate networks by placing router interfaces in different virtual routing tables. When a VRF is created, interfaces must be assigned to a VRF or belong to the global (normal) routing table. Interfaces can't belong to more than one VRF or routing table. Additionally, under normal circumstances, information is not shared between VRFs; they are isolated from one another.

To create a VRF, it must be named, and interfaces must be associated with it.

```
Router(config)# ip vrf <name>
```

At this point, the VRF is alive, but no interfaces have been assigned to it. To assign an interface to an VRF, use the **`ip vrf forwarding <VRF>`** command within the interface configuration mode.

```
Router(config-int)# ip vrf forwarding <VRF>
```

Now the VRF is configured, and all routing information learned through that interface is populated in the VRF assigned. All interfaces not explicitly assigned to a VRF remain in the global VRF.

To view the routes in a VRF, the VRF must be specified in the **`show ip route`** command:

```
Router# show ip route vrf <VRF>
```

Or the global routing table is shown. Likewise many other commands such as `ping`, `traceroute`, or `ip route` assume the global routing table is used unless a VRF is specified within the command.

OSPF, BGP, EIGRP, and RIPv2 all support the use of VRFs and can send dynamic routing updates with the VRF framework. Each routing protocol has a specific syntax with which is supports VRFs.

 Please see page 245 for more details.

- **OSPF:** Runs an instance per VRF. All commands applied to the instance are valid for that VRF only.

```
Router(config)# router ospf <instance> vrf <VRF NAME>
Router(config-router)#
```

  All OSPF commands are configured as they would be in non-VRF topologies.

- **EIGRP:** In global configuration mode, the AS number entered is the global AS number. The IPv4 address family is entered for each VRF, and commands specific to the VRF are applied. Each VRF must be configured with a unique AS number:

```
Router(config)# router eigrp <Global AS #>
Router(config-router)# address-family ipv4 vrf <VRF NAME>
Router(config-router)# autonomous-system <AS #>
```

EIGRP configuration commands such as `network` are configured in the address family configuration mode:

- **RIPv2:** The IPv4 address family is entered for each VRF, and commands specific to the VRF are applied:

```
Router(config)# router rip
Router(config)# version 2
Router(config)# address-family ipv4 vrf <VRF NAME>
```

RIPv2 configuration commands such as **network** are configured in the address-family configuration mode.

Multi-VRF CE extends the use of VRFs to create VPNs over a service provider network by implementing VRFs in the customer edge device rather than on the provider edge router. Typical MPLS-VRF implementations rely upon the provider edge (PE) device to create the virtual routing and forwarding tables and the MPLS specific functions such as LDP, label operations, and MPLS specific table maintenance. When the CE router contains separate VRF instances and peers with the PE router to create a VPN over MPLS, a Multi-VRF CE is implemented.

 Please see page 245 for more details.

Some implementations provide VPN support without involving MPLS. Instead, the VRF-aware routers peer with each other and share routes using VRF-aware routing protocols. These routes stay within specific routing tables that create distinct networks or VPNs. Because VRFs and VRF-aware routing protocols are involved, but MPLS is not, these distinct category of networks are called VRF-lite VPNs rather than MPLS VPNs.

 Please see page 245 for more details.

Chapter 11

IPv6

Address Structure

An IPv6 address contains 128 bits, a much larger address space than the address space in IPv4. It can provide approximately 3.4 * 10 ^ 38 addresses.

IPv6 addresses are represented as a series of 16-bit fields presented as a hexadecimal number and separated by colons (:). The format used is *x*:*x*:*x*:*x*:*x*:*x*:*x*:*x*.

To shorten the writing of IPv6 addresses, you can use the following techniques:

- The leading 0s in a field are optional.
- You can use two colons (::) to compress successive hexadecimal fields of 0s at the beginning, middle, or end of an IPv6 address; this can be done one time in an address (see Figure 11-1).

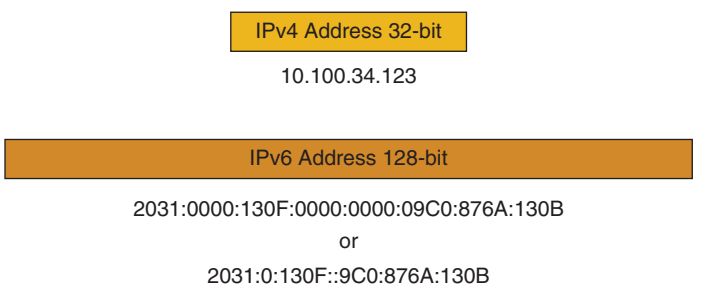


FIGURE 11-1
IP Addresses

 Please see page 245 for more details.

Benefits

The main benefits of IPv6 include the following:

- Has a larger IP address space
- Eliminates the need for Network Address Translation (NAT)
- Enables hosts to have multiple IPv6 addresses and networks to have multiple IPv6 prefixes (site multihoming)
- Includes a fixed header size that makes processing more efficient
- Includes optional security headers
- Has increased mobility and multicast capabilities
- Includes a new capability to enable packet labeling to belong to particular traffic "flows" so that the sender can request special handling

Datagram Structure

The header has eight fields:

- **Version:** A 4-bit field that indicates the IP version.
- **Traffic Class:** An 8-bit field that tags packets with a traffic class used in differentiated services.
- **Flow Label:** A 20-bit field that a source uses to label sequences of packets for which the source requests special handling by the IPv6 routers.
- **Payload Length:** A 16-bit field similar to the Total Length field in the IPv4 packet header.
- **Next Header:** An 8-bit field similar to the Protocol field in the IPv4 packet header. This field differs from IPv4 because there can be a stack of multiple headers within one IPv6 header, unlike IPv4.
- **Hop Limit:** This 8-bit field specifies the maximum number of hops an IP packet can traverse and is similar to the Time To Live (TTL) field in the IPv4 packet header.
- **Source Address:** A 128-bit (16-octet) field that contains the packet's source address.
- **Destination Address:** A 128-bit (16-octet) field that contains the destination address.

 Please see page 245 for more details.

Address Types

Scope types under version 6 include the following:

- **Unicast**
- **Anycast**: An identifier for a set of interfaces that typically belong to different nodes. A packet sent to an anycast address is delivered to the closest interface, as defined by the routing protocols in use, identified by the anycast address.
- **Multicast**

Address Scopes

- **Link-local address:** An IPv6 unicast address that you can manually configure or have automatically configured on an IPv6 interface. When configured automatically, the address uses the link-local prefix FE80::/10 (1111 111010) and the interface identifier. Link-local addresses are used in the neighbor discovery protocol, the stateless autoconfiguration process, and many other control operations such as routing protocols.
- **Site-local address:** IPv6 unicast addresses that use the prefix FEC0::/10 (1111 111011) and concatenate the subnet identifier (the 16-bit field) with the interface identifier. These addresses are similar to RFC 1918 private addresses in IPv4; they are not advertised beyond the local site. This feature has been deprecated in the standards.
- **Global aggregatable address:** Enable strict aggregation of routing prefixes that limit the number of routing table entries in the global routing table. These are the unique addresses assigned by service providers or regional registries for participation in the public network.

IPv6 Neighbor Discovery

IPv6 neighbor discovery enables for the following functions:

- Determine the link-layer address of a device on the same local link. This is similar to the function of ARP in IPv4.

 Please see page 245 for more details.

- Find neighbor routers.
- Track neighbor routers.

The IPv6 neighbor discovery process is the solicited-node multicast address. Any node must join the multicast group corresponding to each of its unicast and anycast addresses. The solicited-node address is composed of the FF02:0:0:0:0:1:FF/104 prefix concatenated with the right-most 24 bits of the corresponding unicast or anycast address. The solicited-node addresses are used for neighbor solicitation messages. The source node takes the right-most 24 bits of the IPv6 address of the destination node and sends a neighbor solicitation message to the multicast group on the link-local address. The corresponding node responds with its link-layer address.

IPv6 Multicast

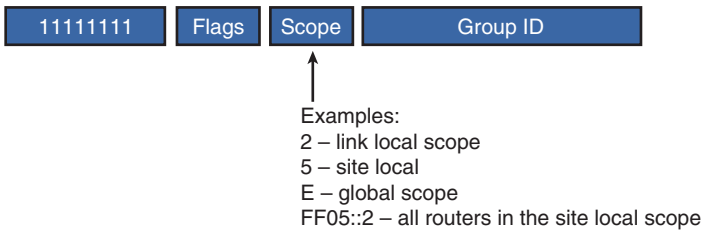
IPv6 multicast is based on the same basic principles as IPv4 multicast. One big difference, however, is that IPv6 relies on multicast for more functions. For example, neighbor discovery, node autoconfiguration, and Mobile IPv6 all rely heavily on IPv6 multicast for their operations. Internet Group Management Protocol (IGMP) is dropped in IPv6 multicast. Multicast Listener Discovery (MLD) now replaces IGMP.

You should immediately recognize a multicast address in IPv6. The address starts with FF, as shown in Figure 11-2.

FIGURE 11-2
IPv6 Multicast Addresses

The scope portion of the IPv6 multicast address controls how far the multicast traffic can flow through the network. Figure 11-3 provides some examples.

FIGURE 11-3
IPv6 Multicast Addresses—Scope



Deployment Strategies

Three primary mechanisms help with the transition from IPv4 to IPv6:

- **Dual stack:** Both the IPv4 and the IPv6 stacks run on a system that can communicate with both IPv6 and IPv4 devices.
- **Tunneling:** IPv6 packets are encapsulated to traverse IPv4 networks and vice versa.
- **Translation:** This mechanism translates one protocol to the other to facilitate communication between the two networks.

Open Shortest Path First Version 3

Open Shortest Path First Version 3 (OSPFv3) has more similarities to the previous version of the routing protocol than it does differences. You should leverage your existing knowledge of OSPFv2 when you study this protocol. Here are just some of the similarities between the two protocols:

- The two are so similar in nature that they can run concurrently in the network without problems.
- OSPFv3 uses the same basic packet types as the previous version; for example, a Database Description Packet still used checks for database synchronization.

 Please see page 245 for more details.

- The neighbor discovery process and the adjacency formation process between two OSPFv3 speakers is identical to that of the previous version.
- Both versions recognize the same network types, and both treat these network types in a similar manner.
- The link-state advertisement (LSA) flooding and aging mechanisms are identical in the two protocols and the timers.

Although there are many similarities, some differences do exist:

- Link-local addresses are used for the formation of adjacencies.
- Multiple IPv6 subnets can be assigned to a single link; OSPFv3 is per interface, not per network.
- Two nodes can communicate over a link even if they do not share a common subnet.

Here is a sample OSPFv3 configuration. This configuration is based on Figure 11-4.

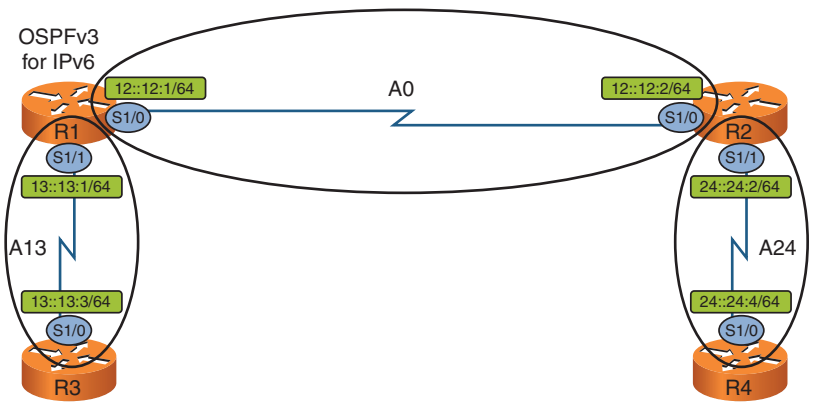


FIGURE 11-4
OSPFv3 Sample Configuration

 Please see page 245 for more details.

```
R1
R1(config)# ipv6 unicast-routing
R1(config)# ipv6 router ospf 1
R1(config-router)# router-id 0.0.0.1
R1(config-router)# interface serial 1/0
R1(config-if)# ipv6 ospf 1 area 0
R1(config-if)# interface serial 1/1
R1(config-if)# ipv6 ospf 1 area 13
R2
R2(config)# ipv6 unicast-routing
R2(config)# ipv6 router ospf 1
R2(config-router)# router-id 0.0.0.2
R2(config-router)# interface serial 1/0
R2(config-if)# ipv6 ospf 1 area 0
R2(config-if)# interface serial 1/1
R2(config-if)# ipv6 ospf 1 area 24
R3
R3(config)# ipv6 unicast-routing
R3(config)# ipv6 router ospf 1
R3(config-router)# router-id 0.0.0.3
R3(config-router)# interface serial 1/0
R3(config-if)# ipv6 ospf 1 area 13
R4
R4(config)# ipv6 unicast-routing
R4(config)# ipv6 router ospf 1
R4(config-router)# router-id 0.0.0.4
R4(config-router)# interface serial 1/0
R4(config-if)# ipv6 ospf 1 area 24
```

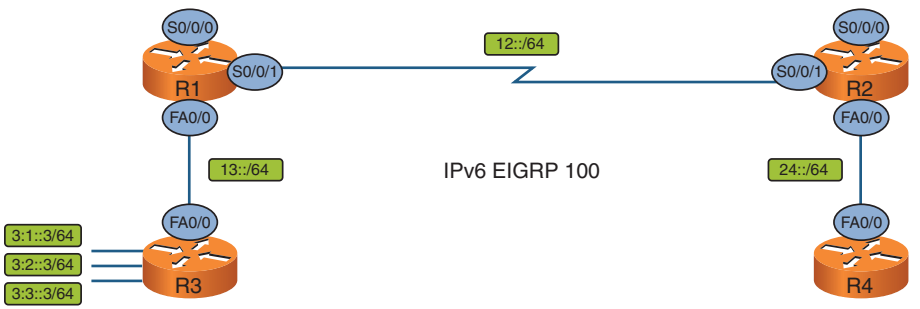
 Please see page 245 for more details.

Enhanced Interior Gateway Routing Protocol Version 6

Like OSPFv3, EIGRPv6 can coexist with the previous version of the protocol. Also, EIGRPv6 is configured using interface configuration commands rather than the **network** command. Link-local addressing is used for adjacencies. Like OSPFv3, a router ID value is required. This value is automatically taken from an interface with an IPv4 address. If no such interface exists, you must provide the router ID. Also, the routing process has a shutdown feature and defaults to the shutdown state. Finally, it is worth noting that there is no longer any auto-summarization behavior, as in the previous version of the protocol.

The following configuration is based on Figure 11-5.

FIGURE 11-5
Sample EIGRPv6 Configuration




```
R1
R1(config)# ipv6 unicast-routing
R1(config)# ipv6 router eigrp 100
R1(config-router)# router-id 0.0.0.1
R1(config-router)# no shutdown
R1(config-router)# interface serial 0/0/1
R1(config-if)# ipv6 eigrp 100
R1(config-if)# interface fastethernet 0/0
R1(config-if)# ipv6 eigrp 100
R2
R2(config)# ipv6 unicast-routing
```

 Please see page 245 for more details.

```
R2(config)# ipv6 router eigrp 100
R2(config-router)# router-id 0.0.0.2
R2(config-router)# no shutdown
R2(config-router)# interface serial 0/0/1
R2(config-if)# ipv6 eigrp 100
R2(config-if)# interface fastethernet 0/0
R2(config-if)# ipv6 eigrp 100
R3
R3(config)# ipv6 unicast-routing
R3(config)# ipv6 router eigrp 100
R3(config-router)# router-id 0.0.0.3
R3(config-router)# no shutdown
R3(config-router)# interface fastethernet 0/0
R3(config-if)# ipv6 eigrp 100
R3(config-if)# interface loopback 301
R3(config-if)# ipv6 eigrp 100
R3(config-if)# interface loopback 302
R3(config-if)# ipv6 eigrp 100
R3(config-if)# interface loopback 303
R4
R4(config)# ipv6 unicast-routing
R4(config)# ipv6 router eigrp 100
R4(config-router)# router-id 0.0.0.4
R4(config-router)# no shutdown
R4(config-if)# interface fastethernet 0/0
R4(config-if)# ipv6 eigrp 100
```

 Please see page 245 for more details.

Chapter 12

Implementing Layer 2 Technologies

Implementing VLAN and VLAN Trunking Protocol

DTP

Dynamic Trunking Protocol (DTP) is a Cisco proprietary protocol that negotiates the trunking status of a switchport. Connected switches exchange DTP messages that indicate their desirability to create a trunk. The DTP port state dictates its capability to create a trunk. Following are the possible states:

- **auto:** Enables the switch to create a trunk if initiated from the other switch. A switch programmed with **auto** cannot initiate a trunk but can form a trunk if the other side initiates. The trunk is formed with **desirable** and **on.**
- **desirable:** Actively tries to create a trunk link with the peer. The trunk is formed with **auto**, **desirable**, and on.
- **on:** DTP messages are sent, and a trunk is formed unless the peer explicitly forbids it. The trunk is formed with **auto**, **desirable**, and **on**.
- **off:** Trunking is not enabled on the switchport regardless of the DTP status of the peer.
- **nonegotiate:** Disables DTP and does not form a trunk link with a peer that requires trunk negotiation. Trunk is formed with **on** and **nonegotiate**.

 Please see page 245 for more details.

Implement SPAN and RSPAN

Remote Switched Port Analyzer (RSPAN)

A method of transporting source SPAN data to a remote destination. This is used when the SPAN source and the SPAN destination are located on two different switches. The data must be transported over a special purpose VLAN, which is configured on all switches in the transit path. The VLAN must be configured as a **remote-span** in VLAN configuration mode:

```
Router(config-vlan)# remote-span
```

On the switch in which the data is originated, the SPAN destination is the RSPAN VLAN. On the switch in which the data is ultimately destined, the SPAN source is specified as the RSPAN VLAN, and the destination is a physical port. All transit switches simply trunk the RSPAN between switches.

Implement Frame Relay

Discard Eligible (DE)

A Frame Relay frame contains a number of fields, one of which is the Discard Eligible bit. A frame with the Discard Eligible bit set indicates the preference of that frame to be dropped because of congestion over those without the bit set. In periods of congestion, the service provider drops frames above the CIR, and the DE bit provides a way to specify prioritization. This enables less important information to be dropped prior to critical packets. A router can be configured to set the DE bit of specific types of traffic using QoS mechanisms such as MQC and de-lists.

Full Mesh

Full mesh is an networking topology in which each router in the network is directly connected to every other router in the network. This means that a virtual circuit (VC) exists between every router, creating a total of n(n – 1) / 2 VCs. This topology is efficient because only a single hop is necessary to reach any other node on the network. Full-mesh topologies can tolerate a large number of link failures and maintain connectivity because the frames can traverse around the failure through other connected nodes. The costs involved in building and maintaining a full-mesh topology are large because of the number of links. Not only because of the cost of the links but also the cost to create and maintain those links. As a result, large full mesh topologies are often not practical.

 Please see page 245 for more details.

Hub and Spoke

Hub and spoke is a network topology in which one router, the hub, connects to every other router in the network. All other routers, and the spokes, connect only to the hub, not other spokes. Network traffic between spokes must travel from the source spoke, through the hub, toward the destination spoke. To create this topology, only one link is required for each spoke, bringing the total number of links to n—1, where n is the number of routers (assuming only one hub). This topology is scalable and less expensive than full mesh but it is not as fault-tolerant. A loss of the hub results in complete communication failure between all nodes. As a result, many networks employ more than one hub that can maintain connectivity to the spokes if the loss of one hub occurs.

Implement High-Level Data Link Control (HDLC) and PPP

HLDC

High-level Data Link Control (HDLC) is a common link layer technology in use today. There are two main functions of HDLC: link management and data transfer. As part of the data transfer, HDLC provides error detection and flow control.

Although HDLC is the default encapsulation on a Cisco serial interface, it can be manually configured using the following interface command:

```
Router(config-int)# encapsulation hdlc
```

The current encapsulation on a serial interface can be viewed with the **show interface** command:

```
Router# show interface s0/0

Serial0/0 is up, line protocol is UP
  Hardware is GT96K Serial
  Internet address is 192.168.20.1/24
  MTU 1500 bytes, BW 1544 Kbit/sec, DLY 2000 usec,
    reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  CRC checking enabled
<OUTPUT OMITTED>
```

 Please see page 245 for more details.

PPP

One of the most important networking protocols is the Point-to-Point protocol (PPP). Not only can it be used to transport many network protocols over point-to-point links, it provides capabilities to add authentication, link quality testing, allocate IP addresses, and provide error detection. The four major parts to PPP are network datagram encapsulation (such as HDLC), link setup, termination and maintenance, and network layer protocol establishment and configuration.

When a PPP link is established, the link setup and maintenance protocol, LCP, is used. LCP is responsible for PPP link establishment, termination, and maintenance. When the link is established, different parameters are exchanged, and the link can optionally be quality tested. After the link establishment phase, the Network Control Protocols (NCP) take over and facilitate the transfer of layer 3 protocols such as IP.

To configure a serial link to use PPP, simply set the encapsulation:

```
Router(config-if)# encapsulation ppp
```

One of the features of PPP encapsulation is the capability to authenticate the link. PPP can use both Password Authentication Protocol (PAP) or Challenge-Handshake Authentication Protocol (CHAP). CHAP authentication is considered more secure than PAP authentication because the username and password are not transferred in the clear as they are in PAP. Instead CHAP uses more secure hashs and frequent challenges that add security to the authentication process.

To configure PAP on a serial interface, the authentication credentials must be created first. This is typically done by created a username and password combination on each router. The interface then contains a command that specifies what is sent to the peer router. If the two match, authentication succeeds:

```
Configuration on R1 -
R1(config)# username Router2 password Cisco
R1(config)# int <INTERFACE>
R1(config-if)# encapsulation ppp
R1(config-if)# ppp authentication pap
R1(config-if)# ppp pap sent-username Router1 password Cisco
```

 Please see page 245 for more details.

```
Configuration on R2 -
R2(config)# username Router1 password Cisco
R2(config)# int <INTERFACE>
R2(config-if)# encapsulation ppp
R2(config-if)# ppp authentication pap
R2(config-if)# ppp pap sent-username Router2 password Cisco
```

CHAP authentication is configured slightly differently because the hostnames of the routers are involved. A username is created matching the hostname of the peer router and the password supplied must match:

```
Router(config)# hostname R1
R1(config)# username R2 password Cisco
```

This means that when R1 tries to authenticate to R2, it will use the password Cisco.

```
R1(config)# int <INTERFACE>
R1(config-if)# encapsulation ppp
R1(config-if)# ppp authentication chap
Router(config)# hostname R2
R2(config)# username R1 password Cisco
R2(config)# int <INTERFACE>
R2(config-if)# encapsulation ppp
R2(config-if)# ppp authentication chap
```

Unfortunately, configuring PAP and CHAP on a PPP over Frame Relay is not be as easy as enabling or disabling the authentication. When dealing with Frame-Relay interfaces, a virtual template must be configured and applied to the interface.

First, a virtual-template interface must be created, and its encapsulation must be PPP:

```
Router(config)# hostname R1
```

 Please see page 245 for more details.

```
R1(config)# int virtual-template <#>
R1(config-if)# encapsulation ppp
```

The IP address from the serial interface is moved to the virtual-template:

```
R1(config)# int <SERIAL INTERFACE>
R1(config-if)# no ip address
R1(config-if)# int virtual-template <#>
R1(config-if)# ip address <IP> <MASK>
The frame-relay interface-dlci is programmed to reference the virtual-template:
R1(config-if)# frame-relay interface-dlci <DLCI> ppp virtual-template <#>
Now authentication can be configured on the virtual-template as before:
R1(config-if)# ppp authentication chap
or
R1(config-if)# ppp authentication pap
```

The details of how to configure the authentication are the same using the new virtual-template interfaces.

 Please see page 245 for more details.

Chapter 13

Implementing IPv4

IPv4 Addressing

IPv4 addresses consist of 32 bits. These 32 bits are divided into four sections of 8 bits, each called an octet. Addresses are typically represented in dotted-decimal notation. For example

10.200.34.201

Subnet masks identify which portion of the address identifies a particular network and which portion identifies a host on the network.

The address classes defined for IP networks consist of the following subnet masks:

Class A 255.0.0.0 (8 bits)
Class B 255.255.0.0 (16 bits)
Class C 255.255.255.0 (24 bits)

Class A addresses begin with 0 and have a first octet in decimal of 1 to 127. Class B addresses begin with 10 and range from 128 to 191. Class C addresses begin with 110 and range from 192 to 223.

Class D and Class E addresses also are defined. The Class D address space has the first 4 bits set to 1110 and has a first octet of 224 to 247. These addresses are used for IP multicast.

Class E addresses have the first 4 bits set to 1111 and have a first octet of 248 to 255. These addresses are reserved for experimental use.

 Please see page 245 for more details.

Of the entire IPv4 address space, several blocks of IPs have been reserved for a specific use. The private IP space, meaning it should not be used outside of an administrative domain, has been allocated the following blocks:

10.0.0.0 to 10.255.255.255
172.16.0.0 to 172.31.255.255
192.168.0.0 to 192.168.255.255

Other allocated ranges include the multicast ranges (224.0.0.0 to 239.255.255.255), the loopback range (127.0.0.0 to 127.255.255.255), and the link local range (169.254.0.0 to 169.254.255.255).

Variable Length Subnet Masking

One of the fundamental concepts in networking is subnetting, that is, breaking one subnet into smaller pieces. With Variable Length Subnet Masking (VLSM), is a subnet can be broken into variable length pieces. To illustrate, a /24 network can be broken up into two /25 networks, four /26 networks, or eight /27 networks.

Before VLSM, only one of these options could be chosen. With VLSM, the same /24 network can be subnetted into one /25, one /26, and two /27s. That is, the new, smaller subnets can be of variable length; they don't need to be a single length (/25, /26, or /27).

Before VLSM, to properly address a series of point-to-point networks, a /30 subnet was required. Without variable length subnets, an entire network would need to be subnetted into /30 networks. If only a handful of /30s where required, many IPs would be wasted. VLSM enables a network administrator to choose subnetting boundaries based on the requirements of the network, rather than being forced to design around the constraints of IP addressing.

VLSM does not change other rules of IP addressing. If a /24 network is subnetted into one /25, one /26, and two /27s, the organization must follow the standard "breaks" between subnets. In other words, the order of the subnets matter. The /24 cannot be broken into a /25, then one /27, and then a /26, followed by the second /27. The subnetting must occur along natural breaks.

VLSM is often confused with classless networking and CIDR. They are related but refer to different IP addressing concepts. Classless networking refers to the delinking of Class A, B, C, and D networks from actual IP addresses. In a classless network, a subnet within the 10.x.x.x range does not need to be a /8. CIDR is a method in which subnets can be grouped together. It provides a way to refer a list of consecutive subnets without having to list each one individually. For example, the

 Please see page 245 for more details.

subnets of 192.168.0.0/24, 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24 can be aggregated together and referred to as 192.168.0.0/22. It is massively useful in large networks where large groups of IP address ranges can be aggregated together within a routing table or access lists.

Implementing IPv4 Tunneling and GRE

Typical network traffic is consistent with the TCP/IP model meaning it has a number of distinct headers. For example, a packet can have a layer 2 header, such as Ethernet; a layer 3 header (such as IP); and a layer 4 header, such as TCP. When tunneling is implemented, a header with data is encapsulated within a header at the same layer. In IPv4 tunneling, an IP packet typically contains another IP packet.

Tunneling is used for a number of reasons including connecting two disjointed networks that might not have IP communication between them. There are a number of IPv4 tunneling protocols including IPsec and Generic Router Encapsulation (GRE).

GRE can carry an arbitrary payload (such as IPv4, IPv6, or IPsec) using IP packets of protocol 47. It does not, however, encrypt any data that is tunneled. GRE tunnels can be used with OSPF to extend the backbone to a disconnected area. Care must be taken to ensure that the route to the destination address provided during the GRE configuration is not learned via OSPF. This can lead to a recursive route, causing the GRE tunnel to bounce.

To create a GRE tunnel, it is necessary to create the numbered tunnel interface:

```
Router(config)# int tunnel <#>
Configure an IP address on the interface:
Router(config-int)# ip address <IP> <MASK>
Specify the source interface or IP address:
Router(config-int)# tunnel source <int or IP>
Specify the destination address:
Router(config-int)# tunnel destination <IP>
```

 Please see page 245 for more details.

Implementing IPv4 Open Shortest Path First (OSPF)

Standard OSPF Areas

Any nonzero area must be connected to area 0 through an area border router (ABR) or virtual link. An ABR that connects to a standard area can advertise network summary (type 3), ASBR summary (type 4), and external summary(type 5) LSAs into the area. Autonomous System Border Routers (ASBR) might be present within a standard area.

To configure an OSPF router to be in a standard area, simply specify the area in the required **network** statement:

```
Router(config)# router ospf <process-id>
Router(config-router)# network <IP> <WILDCARD MASK> area <#>
or configure the area within the interface:
Router(config-int)# ip ospf <process-id> area <#>
```

A router configured to be part of area 0 and another area becomes an ABR. Remember that each area must connect to area 0.

A router that advertises external networks into OSPF becomes an ASBR.

To view the current configuration of areas within the router

```
Router# show ip ospf
```

Stub Area

A stub area is an area that does not permit the advertisement of type 5 (external) LSAs. Instead, these LSAs are replaced with a default route advertisement. Type 3 and Type 4 advertisements are sent into the area from the ABR. Stub areas are to be used when all traffic destined to an external network would travel through an ABR. A default route accomplishes this while saving resources.

For an OSPF adjacency to form, routers must agree on the area type. This means that all routers within a stub area must be configured as a stub:

```
Router(config)# router ospf <process-id>
Router(config-router)# area <#> stub
```

 Please see page 245 for more details.

Because stub areas do not allow the propagation of type 5 LSAs, an ASBR cannot be part of a stub area. A not-so-stubby area (NSSA) was created for this purpose. Additionally, a virtual link cannot transverse a stub area.

Totally Stubby Area

A totally stubby area is a Cisco proprietary feature that extends the concepts of a stub area a step further. In addition to the Type 5 (external summary) LSAs replaced by the ABR, Type 3 (network summary) and Type 4 (ASBR summary) LSAs are replaced with a default route as well.

To configure an area as totally stubby

```
Router(config)# router ospf <process-id>
Router(config-router)# area <#> stub no-summary
```

Unlike the configuration of a stub area, the `no-summary` command is required only on the ABR, not all routers within the area. All other routers (non-ABR) require only the '`area` <#> `stub`' command.

Like stub areas, ASBRs and virtual links are not allowed within totally stubby areas.

Not-So-Stubby-Area (NSSA)

One of the limitations of stub areas is that they do not enable ASBRs. Because ASBRs advertise Type 5 (external summary) LSAs into an area, they violate the objective of a stub area, namely to disallow such LSAs. There is a need, in some networks, to have an ASBR inject external routes into an area, while limiting external routes from ASBRs in other areas. To do this a Type 7, NSSA external, LSA was created. An ASBR can inject Type 7 LSAs into a stub area that are converted to Type 5 LSAs by the ABR connected to the backbone area.

To configure an area as NSSA:

```
Router(config)# router ospf <process-id>
Router(config-router)# area <#> nssa
```

Like the stub area, all routers within the area must agree that the area is an NSSA.

 Please see page 245 for more details.

When an NSSA area is created, the ABR does not create a default summary route. If a summary route is desired, a totally NSSA area can be used or **default-information-originate** can be added to the command:

```
Router(config-router)# area <#> nssa default-information originate
```

Totally NSSA

The Totally NSSA area is a Cisco proprietary enhancement to the NSSA concept that extends the NSSA concept to replace Type 3 and Type 4 LSAs with a default route. Like the NSSA area, it does enable Type 7 LSAs to be generated by an ASBR. Unlike the NSSA area, it converts the Type 5, Type 3, and Type 4 LSAs into a default route that is advertised in the area.

To configure an area as NSSA, the **no-summary** option simply needs to be added to the NSSA area command:

```
Router(config)# router ospf <process-id>
Router(config-router)# area <#> nssa no-summary
```

This command is required only on the ABR; all other routers require only the **nssa** option on the area.

Implementing IPv4 Enhanced Interior Gateway Routing Protocol (EIGRP)

EIGRP Queries

If EIGRP detects a change to the network topology, an input event, which requires a change to a route, must perform a check to determine the existence of a Feasible Successor (FS). If an FS is not found, the Query process must be initiated, which is going Active on a route.

When a route is active, an EIGRP router uses a multicast query to ask all its neighbors for a valid route to the subnet. Because a received query is considered an input event, EIGRP follows a similar process before responding. If the neighbor router receives a query for a subnet to which it does not have a route to, it sends a unicast reply stating that it has no route. If the neighbor router does have a route to the subnet, that route might be affected by the original Query. In this case, EIGRP goes Active on the route as well. If not, or if the router has a FS, it responds with a unicast EIGRP reply message with the route details. If the query causes the router to go Active on the route, it does not immediately respond but instead generates a Query to all its neighbors. If no router in the EIGRP domain contains a route to the subnet, the route is removed from all routing tables. Otherwise when an FS is found, it is propagated to all the querying routers.

 Please see page 245 for more details.

A route is considered stuck-in-active if no response to the query has been received for a configured amount of time (default 3 minutes). After this time, the EIGRP drops all neighbors that it has not received replies from.

Implementing IPv4 Border Gateway Protocol (BGP)

Internal BGP (IBGP) Versus External BGP (EBGP)

BGP operates by establishing peer relationships with other BGP routers in either an external (eBGP) or internal (iBGP) manner. Internal BGP peers are those that share the same autonomous system (AS) number. By contrast, external BGP peers are those that do not share the same AS number. Although these are minor configuration differences, they are handled in different ways:

Step 1. Packets sent to eBGP peers use a TTL of 1.

Step 2. The next-hop field is updated with the last eBGP peer. It is not updated when iBGP is used.

Step 3. eBGP neighbors do not advertise routes to eBGP neighbors in an AS that is contained within the AS_PATH.

Step 4. iBGP routes have an AD of 200; eBGP routes have an AD of 20.

Step 5. iBGP routes are subject to BGP synchronization (if enabled).

BGP synchronization is the major difference between eBGP and iBGP routes and is characterized by the BGP synchronization rule:

For an iBGP route to be added to the BGP table, the exact prefix must be in the routing table from an IGP.

The synchronization rule is a method that guarantees that a route is known to all routers within the AS even if they are not running BGP. If a route is advertised via iBGP, and a non-BGP router sits logically between the BGP peers, the non-BGP router will black hole the traffic because the destination is not known via IGP. The synchronization check can be turned off (and is by default as of IOS version 12.2(8)T) with the router configuration command:

```
Router(config-router)# no synchronization
```

If disabled, it must be guaranteed that a routing black hole within the AS by creating a full-mesh iBGP network or using a BGP tool such as route reflectors or confederations.

 Please see page 245 for more details.

Implement Performance Routing (PfR)

Performance Routing (PfR) expands upon the existing Optimized Edge Routing (OER) feature set within IOS. Currently, the two terms are relatively interchangeable.

In a network in which multiple exit points exist, OER can be configured to intelligently choose the exit point that maximizes performance based on metrics such as packet loss, response time, load distribution, or cost minimization. These metrics are not available in standard BGP path selection. OER uses two main architectural components: the Master Controller (MC) and the Border Router (BR). The MC controls all OER functions within a domain by providing the monitoring of the data flows and changes to flow policies through control of the border routers. The BR is the device on the edge of the network, through which the data flows. The BR reports on the link measurements to the MC and implements the policy changes that affect the traffic flow. The MC can also live on a BR.

Implement Route Redistribution

Often occasions exist in which two or more routing protocols are used within a domain. Because these two routing protocols might contain information about different networks, if full connectivity is to be created, there must be a way to feed information from one protocol into another. Route Redistribution takes information from one protocol and inserts it into another protocol. A common example of this is found in most modest-sized networks when BGP is run as EGP with a service provider and an IGP is run internally to create full network connectivity inside the domain. BGP advertises the internal network routes to the rest of the Internet and provides a path to the Internet. Because each routing protocol uses different methods for metric calculation, it is difficult to equate a metric in one protocol to a second protocol. Other problems arise, such as learning the same prefix from more than one routing protocol. A decision must be made as to which protocol is more trustworthy. You can create routing loops as routes are advertised from one protocol to another.

To advertise routes into a protocol, enter the routing protocol configuration mode and use the **redistribute** command. Each routing protocol requires specific information that calculates the metric for the redistributed routes. This information is supplied within the **redistribute** command and depends upon the routing protocols.

To redistribute routes into RIP:

```
redistribute protocol [process-id] [match route-type][metric metric-value] [route-map map-tag]
```

 Please see page 245 for more details.

RIP requires a metric to be specified within the normal valid limits of a RIP route. This must be specified after the **metric** option. A route-map can also be optionally applied that enables control over which routes will be redistributed into RIP. The metric type can also be matched that enables a specific type of route to be the only routes redistributed.

To redistribute routes into OSPF:

```
redistribute protocol [process-id] [metric metric-value][metric-type type-value] [route-map map-tag]
[subnets] [tag tag-value]
```

OSPF also requires a metric when importing routes. The subnets option is also almost always used when redistributing routes into OSPF; otherwise, only networks not subnetted will be added. Route maps can specify with granularity specific subnets to be added. The tag option can add a tag to the routes. This helps when determining where a route came from. This is often helpful when trying to stop routing loops because of redistribution.

To redistribute routes into EIGRP:

```
redistribute protocol [process-id] [match {internal | external 1 | external 2}] [metric metric-value]
[route-map map-tag]
```

EIGRP mirrors OSPF and RIP with some of the same options, but additionally adds the match option. This enables only specific route types from OSPF to be inserted into the EIGRP routing process. The metric also must be specified using the EIGRP metrics of bandwidth, delay, load, reliability, and MTU.

 Please see page 245 for more details.

Chapter 14

Implementing IPv6

IPv6 Addressing and Types

Implementing Tunneling Techniques

Tunneling is often used to bridge two disconnected networks. This is done in IPv6 to connect two IPv6 networks through an IPv4 network. There are number of tunneling protocols that can be implemented depending on factors such as topology, address space used, and application.

To configure a tunnel, the tunnel interface must first be created:

```
Router(config)# int tunnel <#>
```

Depending on the tunnel type, the configuration varies.

```
Manual IPv6 in IPv4 tunnels:
```

The tunnel between two IPv6 networks is manually created over an IPv4 backbone. The source IP/interface is specified along with a destination IP address. IPv6 packets are then encapsulated within an IPv4 packet. This type of tunneling is point-to-point and requires that the routers that terminate the tunnel run an IPv4 and an IPv6 simultaneously.

```
Router(config-int)# ipv6 address <IP>
Router(config-int)# tunnel source <int>
Router(config-int)# tunnel destination <IP>
Router(config-int)# tunnel mode ipv6ip
```

 Please see page 245 for more details.

GRE IPv6 in IPv6 Tunnels

A GRE tunnel is created that transports the IPv6 packets across the IPv4 backbone. Much like the manual tunnel, it creates a point-to-point connection between the two networks. Also like the manual tunnel, any address space can be used in the creation of the tunnel. One advantage over manual IPv6 in IPv4 tunnels is that GRE can support a number of different protocols in addition to IPv6. As a practical matter however, this is a limited benefit:

```
Router(config-int)# ipv6 address <IP>
Router(config-int)# tunnel source <int>
Router(config-int)# tunnel destination <IP>
Router(config-int)# tunnel mode gre ip
Automatic 6to4 tunnels:
```

A different approach in creating IPv6 connectivity is to use an automatic, point-to-multipoint tunneling technology. The 6to4 tunnels use the specific address space of 2002::/16, and each IPv6 network that connects to the 6to4 tunnel cloud is assigned a network range of

```
2002:IPv4 address of router::/48
```

Each router then is assigned an IPv4 address on the tunnel interface that must be reachable by the other router in the cloud. A route is then applied to each router:

```
ipv6 route 2002::/16 <tunnel interface>
```

All traffic destined for a subnet in the 2002::/16 range routes to the tunnel interface. The automatic 6to4 tunnel connects to the IP address of the router found in the 32 bits found after the 2002::/16 prefix.

To create an automatic 6to4 tunnel, apply an IPv6 address to the tunnel interface that contains the IPv4 address of the interface that is the source of the tunnel interface:

```
Router(config-int)# ipv6 address <2002:IPv4 address>
Router(config-int)# tunnel source <interface>
```

 Please see page 245 for more details.

The mode is changed to automatic 6to4:

```
Router(config)# tunnel mode ipv6ip 6to4
```

ISATAP Tunnels

Another automatic, point-to-multipoint IPv6 tunneling methodology is the Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). The IP addressing scheme differs from that of the automatic 6to4 tunnels. ISATAP tunnels use a 64-bit prefix followed by the 32-bit sequence of 0000:5EFE followed by the IPv4 address of the remote router that the ISATAP forms a link with.

The IPv6 address of the tunnel interface must also be learned via EUI-64 methods.

```
Router(config-int)# ipv6 address <64bit prefix> eui-64
Router(config-int)# tunnel source <int>
Router(config-int)# tunnel mode ipv6ip isatap
```

ISATAP tunnels also requires that IPv6 router advertisements be reenabled on the tunnel interface.

```
Router(config-int)# no ipv6 nd ra suppress
```

Implement Filtering and Route Redistribution

Traffic filters within IPv6 are not different from IPv4; they follow the same logic and format except that the address spaces are much larger. IPv6 access lists are always named, not numbered. To create an IPv6 access list, specify the name of the traffic filter in global configuration mode:

```
Router(config)# ipv6 access-list <ACL name>
```

Individual access-list entries can then be created just as they would within IPv4. Remember that IPv6 uses prefixes, so if the rule applies to a subnet, CIDR notation must be used.

```
Router(config-ipv6-acl)# permit | deny protocol {source-ipv6-prefix/prefix-length | any | host source-ipv6-
address | auth} [operator [port-number]] {destination-ipv6-prefix/prefix-length | any | host destination-
ipv6-address | auth} [operator [port-number]]
```

 Please see page 245 for more details.

To apply the ACL on an interface, use the **ipv6 traffic-filter** command:

```
Router(config-int)# ipv6 traffic-filter <ACL NAME> in|out
```

The ACL can also be applied to a VTY line to control management access. Like IPv4, use the **access-class** command:

```
Router(config)# line vty 0
Router(config-line)# ipv6 access-class <ACL name> in
```

Route redistribution within IPv6 functions the same as it does with IPv4. Within the routing protocol, the **redistribution** command is specified along with the details depending on the routing protocols involved. When redistributing prefixes into MP-BGP, the address family for Ipv6 must be entered.

Following is an example of redistributing RIP into OSPF:

```
Router(config-router)# redistribute rip <PROCESS NAME>
Example of redistributing OSPF into RIP
Router(config-router)# redistribute ospf <process-id> metric 3
Example of redistributing OSPF into BGP
Router(config-router)# address-family ipv6
Router(config-router-af)# redistribute ospf <process-id> metric 6 metric-type external
```

 Please see page 245 for more details.

CCIE Routing and Switching v4.0 Quick Reference

Brad Ellis
Jacob Uecker
Steven Means

Technical Editor: **Scott Morris**

Published by:

Cisco Press

800 East 96th Street

Indianapolis, IN 46240 USA

First Printing September 2011

ISBN-10: 1-58714-163-9

ISBN-13: 978-1-58714-163-8

Warning and Disclaimer

This book is designed to provide information about the CCIE Routing and Switching written exam. 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 ebook 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 ebook should not be regarded as affecting the validity of any trademark or service mark.

Feedback Information

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

Reader feedback is a natural continuation of this process. If you have any comments on how we could improve the quality of this ebook, or otherwise alter it to better suit your needs, you can contact us through email at feedback@ciscopress.com. Please be sure to include the ebook title and ISBN in your message.

We greatly appreciate your assistance.

Corporate and Government Sales

The publisher offers excellent discounts on this ebook when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact: **U.S. Corporate and Government Sales** 1-800-382-3419 corpsales@pearsontechgroup.com.

For sales outside the United States please contact: **International Sales** international@pearsoned.com

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

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.**