Exam Manual

LearnSmart

CISCO (642-902)

ROUTE

Cisco Certified Network Associate

Smarter Training

LearnSmart's CCNP ROUTE exam manual offers network professionals an in-depth survey of all the skill sets needed to successfully complete the 642-902 exam and become Cisco certified. Topics covered in this guide include:

- Implementation of an EIGRP-based Solution
- Implementation of a Multi-Area OSPF Network
- Border Gataway Protocol
- IPv6
- Path Control and Branch Offices
- And more!

Give yourself the competitive edge necessary to further your career as a network professional and purchase this exam manual today!

CCNP ROUTE (642-902)
LearnSmart Exam Manual

Copyright © 2011 by LearnSmart, LLC.
Product ID: 012376
Production Date: November 10, 2011

Warning and Disclaimer

Volume, Corporate, and Educational Sales

Favorable discounts are offered on all products when ordered in quantity. For more information, please contact us directly:

1-800-418-6789
solutions@learnsmartsystems.com

International Contact Information

International: +1 (813) 769-0920

United Kingdom: (0) 20 8816 8036

Table of Contents

Domain 1: Implementing an EIGRP-based solution

Introduction

EIGRP (Enhanced Interior Gateway Routing Protocol) is a feature-rich routing protocol with a unique vocabulary and set of concepts to master for passing the exam. To understand EIGRP, we must first introduce some concepts and describe how EIGRP works.

EIGRP shares features of both distance vector and link state protocols. EIGRP is a Cisco proprietary protocol. Internal EIGRP routes have an administrative distance of 90 and external EIGRP routes have an administrative distance of 170, while summary routes have an administrative distance of 5.

Note: Administrative Distance is a value determined by a protocol's trustworthiness on a scale from 1 to 255 with 1 being the most trustworthy. If a router has two routes to the same network from different protocols, the route with the lowest administrative distance will be the one used.

EIGRP is composed of three tables, which are shown in Figure 1. After the EIGRP process is started, the router begins to create a neighbor table, a list of state information for neighboring EIGRP routers formed by the exchange of EIGRP messages. Next, the topology table is created from the information contained in the neighbor table through the Diffuse Update Algorithm (DUAL), which calculates an optimal, loop-free routing topology and keeps those routes in memory in case a route fails. Then, routes (which are called successors in EIGRP) are chosen based on certain criteria and added to the routing table. Here the router will use them as it would any other route. EIGRP will route for IPv4 and IPv6 (as well as non-IP protocols) and keep separate neighbor and topology tables for each.

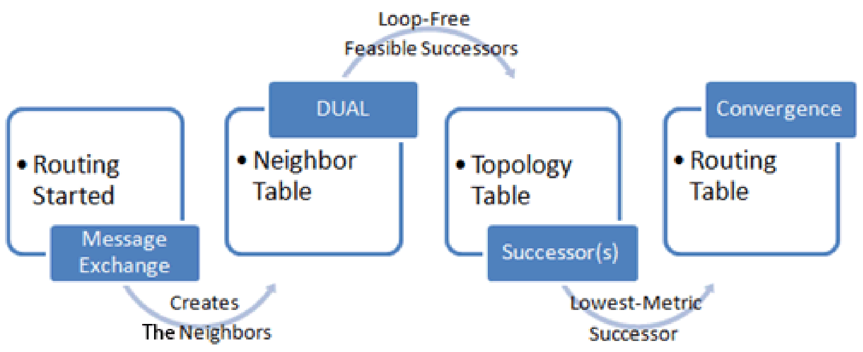

Figure 1: EIGRP Processes and Tables

EIGRP Messages and the Neighbor-Creation Process

The first component and process in EIGRP routing is setting up neighbor relationships. A router sets up relationships with other routers within its reach and with the same Autonomous System (defined by an Autonomous System Number). The ASN defines a domain of routers under a central authority's control and identifies an instance of EIGRP. Neighbor relationships are formed by a series of messages that the routers exchange with each other. Neighbor relationships are not transitive. This means that if R1 is neighbors with R2 and R2 is neighbors with R3, R1 is not automatically neighbors with R3.

EIGRP uses RTP (Reliable Transport Protocol) for message exchange between routers. This is neither TCP nor UDP, but rides on top of IP as protocol number 88. There are six types of messages that EIGRP routers exchange with the goal of setting up neighbor relationships. The six message types are discussed in this section. Note that the RTP protocol used by EIGRP is not the same as the protocol used in voice over IP. The RTP protocol for voice transports sampled audio from calls, while the RTP for EIGRP provides sequencing and acknowledgement between EIGRP packets. Since EIGRP doesn't use TCP or UDP, RTP is responsible for making sure EIGRP messages reliably sent.

The Hello packet is used to discover neighbors and keeps neighbor relationships alive. Hellos are sent to multicast address 224.0.0.10, and are therefore received by all EIGRP-enabled routers on a subnet with a single transmission. Routers reply with Acknowledge packets. An Acknowledgement is simply a Hello with no data. Hello and Acknowledge packets do not use RTP. They are simply IP transmissions with the protocol type field set to 88. This is because EIGRP doesn't need to know if the Hellos were received. A Hello packet is sent to 224.0.0.10 with the request "Are there any neighbors for my AS?" to any router willing to accept it. It then listens for replies in the form of an Acknowledgement with an Update containing the answering router's topology table.

The Hello packet has a timer that is customizable by using the ip **hello-interval eigrp *ASN seconds*** interface command. The default interval for a Hello packet is 5 seconds for links with bandwidth at T1 or greater speeds (T1, Ethernet, frame relay, ATM). It is 60 seconds for links slower than T1 (which can include multipoint frame relay and other forms of serial links). The Hello timer can be shortened to accomplish faster convergence, at the expensive of using more bandwidth. Although not a message type, it is worth mentioning that a Hold timer is the amount of time a router will wait without hearing a Hello message before it declares the link to be dead and begins to change the topology. The default Hold time is 3 times the Hello time. Therefore, the Hold time is 15 seconds for most links.

Update packets are sent to neighbors when changes occur. Most of the time, the Update message contains only route changes. The exception to this rule is when a new neighbor is brought up. First, think of the update as still taking place, except that it contains all the EIGRP routing information instead of just a single change. Second, an initial Update is a unicast response to a neighbor attempting to form a relationship instead of a multicast. This is because initially, Updates are destined as a reply to form specific neighbor relationships. Figure 2 shows this process. If R2 is the router that is responding to R1's initial Hello, R2 will send its entire table as an Update. R1 will then reply with an Acknowledgement and a unicast Update of its own table destined for R2. R2 will finalize this process with a final Acknowledgement. After this, the relationship is established. When a route is added, Updates packets are multicast to advertise the update to neighbors.

After neighbors are established, the multicast Query packet is sent when a route goes down and no backup route (known as a Feasible Successor in the topology table) is available. It is answered by a unicast Reply packet, which is returned by a neighboring router to provide a new route to replace the failed one. Feasible successors are part of the topology table and are discussed in the next section.

The final message type is the Goodbye message. This allows a router to notify its neighbors when it is going down.

Figure 2 shows how EIGRP messages are used when a relationship is formed and when routes are added or removed. After the messages are processed, DUAL (also discussed in the next section) may recalculate the topology with new neighbors. It also shows the results of a Query and Reply, as well as the addition of a route.

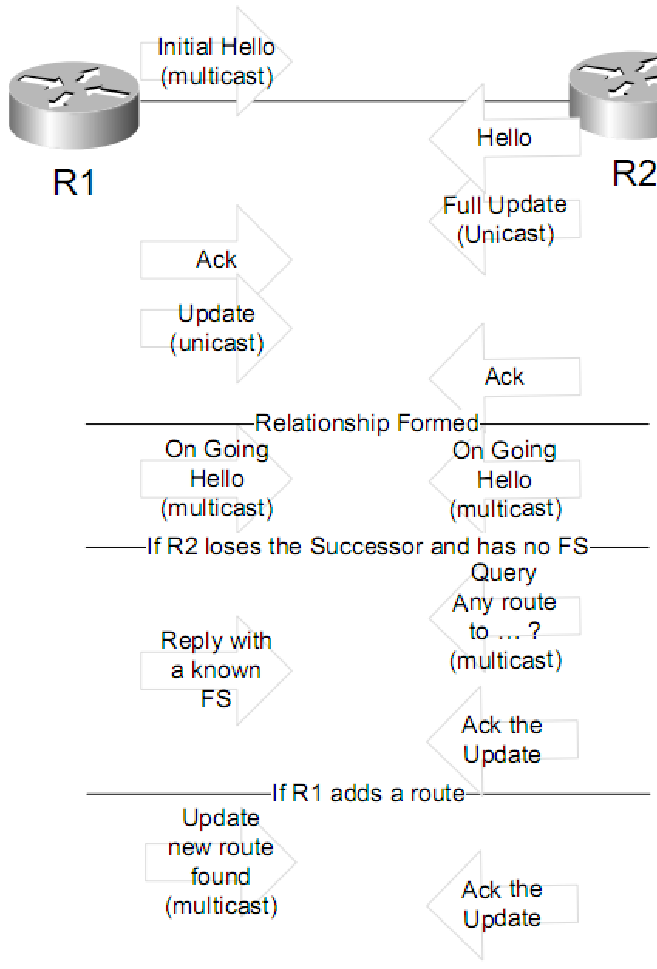
Initial Hello (multicast)
R1
R2
Hello
Full Update (Unicast)
Ack
Update (unicast)
Ack
Relationship Formed
On Going Hello (multicast)
On Going Hello (multicast)
If R2 loses the Successor and has no FS
Query Any route to … ? (multicast)
Reply with a known FS
Ack the Update
If R1 adds a route
Update new route found (multicast)
Ack the Update

Figure 2: EIGRP messages during different events

DUAL, Successors, Loop Avoidance, and EIGRP Topology Table

As stated in the introduction section, the second major component of EIGRP is the topology table. This table is kept in memory and recalculated as changes occur, such as route updates, route failures, or new neighbors coming online. Since the table is kept in memory, a failing route can be replaced almost immediately.

EIGRP's topology calculation algorithm is the Diffusing Update Algorithm (DUAL). The paths in the topology table that DUAL calculates are known as Feasible Successors (FS) and they are ready to be injected into the router's routing table when needed. The main purpose of DUAL is to identify the parameters of the topology and ensure that all successors are free of loops. DUAL calculates FS and keeps them in the topology table before they are needed if a primary route (simply known as the successor) goes down. DUAL ensures that successors and feasible successors are only valid if they are loop-free. Like all routing protocols, EIGRP avoids loops and has loop detection and avoidance mechanisms built in.

After the neighbor table is formed, DUAL begins to find FS. For each FS in a network, EIGRP calculates the path's metric or Feasible Distance (FD). The FS with the lowest FD becomes the Successor and by default is the only one that is put in the routing table. If you want to add more successors to the routing table, you can use the **variance** command. Variance is a multiplier you can set to allow a threshold of successors into the routing table, not just the best one. The EIGRP metric is calculated using these parameters:

Parameter	**K-value**	**Range**	**Measured in**
Bandwidth	K_1	0-10^7	10^7 / kilobits
Load	K_2	0-255	Estimation
Delay	K_3	0-2^{24}	Tens of ms
Reliability	K_4	0-255	Estimation
MTU	K_5	0-Any	kilobits

Figure 3: Calculating the EIGRP Metric

Each parameter is assigned a k value. The k value is used to assign a relative weight, or importance, to each parameter. Delay and bandwidth are the only two values used by default, and their k values are 1. The other k values are 0, indicating that these parameters have zero weight in path metric calculation. An administrator can tweak the k values so EIGRP's routes will meet their network requirements. For example, if a network's reliability is more important to an administrator than its delay, k4 can be assigned a relative value > 1. The "master equation" of EIGRP is:

$$metric = \left[K_1 \times bandwidth + \frac{K_2 \times bandwidth}{256\text{ - }load} + K_3 \times delay\right] \times \left[\frac{K_5}{reliability + K_4}\right]$$

Figure 4: The EIGRP "Master Equation"

You probably will not need to memorize this equation for the exam, but it shows how the parameters can be tweaked to optimize your network. Keep the fact that EIGRP can be optimized in mind for the exam, and when planning and designing your real-world networks. Cisco recommends that the k values not be changed because changing the k values can be problematic, as they must be changed on all EIGRP routers.

Because only bandwidth and delay have values of 1 by default, and other values have 0, the default and most common calculation of metric is:

((107/ minimum path bandwidth) + sum of delays)*256

107 is used as a normalizing constant and represents a 10 gigabit link. The bandwidth and delay in EIGRP are comprehensive. Bandwidth to a network is only as good as its slowest link, while delay to a network is the sum of all link delays.

It is important to note that the k values must be modified on either all routers in an AS or none. If metrics are calculated in different ways between routers, DUAL could create a looped topology. This is because DUAL uses the route with the smallest FD as the route in use, but at the same time is advertising the FD to other routers. Calculating metrics different ways would create unpredictable results as routers update their routing tables with incorrectly calculated FDs.

Reported Distances

Like other routing protocols, routers advertise the routes they know about to other routers. The distance to a network EIGRP receives from other routers are called Reported Distance (RD). Each router receives an advertisement with another router's calculated distance. EIGRP routers advertise successors to other routers in the same AS. To stop advertising a route, the router can filter the route with a **route-map**, which is discussed in more detail in the sections on BGP and Redistribution. The Reported Distance received is important in calculating a loop-free topology.

Feasibility Requirement

A route is considered feasible and may then be an FS if it meets the following rule:
A route is feasible if the Reported Distance of alternate routes is lower than the Feasible Distance of the existing route.

In other words, if a router has more than one route to a network, routes are only considered to be feasible if the reported distance it receives from another router is lower than the current feasible distance.
The Feasibility Requirement keeps a network loop free by setting the requirement that reported distances must be less than the router's own calculated distance for a given network before it is entered in to the routing table. If a RD is greater than the FD, then the competing routes must contain loops or be actually be longer. The calculation only works if all of the k values are the same across an AS.

Like other protocols EIGRP avoids creating loops by using Split Horizon and Poison Reverse. Split Horizon is the rule that a route isn't sent back out the interface it was received in. Split horizon is normally running when a router is not calculating a topology change that alters which interface a router should use to access a route. When a topology change like this occurs, Split Horizon turns off and Poison Reverse turns on. Poison Reverse poisons all of the old routes on a router's interfaces.

Study Suggestion
Remember: Split Horizon and Poison Reverse from the CCNA. These are general routing methods for avoiding looping and are used by several protocols. Be prepared to identify and eliminate routing loops on the exam.

EIGRP Topology

This section gives an example of how EIGRP calculates feasible distances and reported distances. Using Figure 5, assume that R1 needs to send data to a network connected to R4. It must decide on an optimal path to R4, either via R2 or R3. The costs on each link are given in numbers that are easy to work with. These costs can be thought of as bandwidth, delay, or any other measurement that a k value can represent.

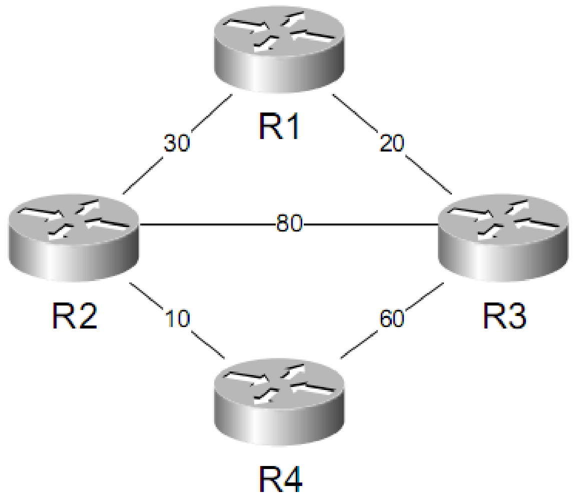

Figure 5: An Example Topology

We can calculate the costs as follows:

Route	R1's FD to this route	Reported Distance to R4
R1 – R2 – R4	40	R2 will report 10 to R1
R1 – R3 – R4	80	R3 will report 60 to R1
R1 – R2 – R3 – R4	170	R2 will report 140 to R1
R1 – R3 – R2 – R4	110	R3 will report 90 to R1

Figure 6: Calculating Route Cost

EIGRP will receive these reported distances in EIGRP messages. R1 will calculate the FDs for each route and by default, inject the lowest FD route in to the routing table. This is the path R1 – R2 – R4, which will be the successor.

R1 then must calculate feasible successors (if any), so it has those ready in case its successor path from R1 – R2 – R4 goes down. Which of these values will be feasible successors?

Because the successor's FD is 40, none of the other routes will be added as feasible successors.
The advertised distances are all greater than 40. This means that if the path from R1 – R2 – R4 goes down, R1 will have to Query and wait for a Reply with a new successor. In this case, our backup routes are not placed in to the topology table because their costs are too high. They will become part of the topology table if our successor goes down.

The EIGRP Routing Table

An FS for a given network will be sent to the routing table if the FD is less than the variance multiplied by the router's lowest FD to a network. The variance is 1 by default, which means that only routes equal to the lowest FD to a network will be inserted. The variance can be used as a multiplier that sets a threshold of acceptable FDs to the insert in the routing table.

Cisco uses 'D' to note EIGRP routes in the routing table. An example is given below.

```
R3#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B – BGP, D -
EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area, N1 - OSPF
NSSA external type 1, N2 - OSPF NSSA external type 2, E1 - OSPF external
type 1, E2 - OSPF external type 2, i - IS-IS, su - IS-IS summary, L1 -
IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area, * - candidate
default, U - per-user static route, o - ODR, P - periodic downloaded
static route

Gateway of last resort is not set
     1.0.0.0/24 is subnetted, 1 subnets
D    1.1.1.0 [90/2297856] via 10.7.0.5, 02:41:23, Serial0/0
     2.0.0.0/24 is subnetted, 1 subnets
D   2.2.2.0 [90/2297856] via 10.7.0.10, 02:41:23, Serial0/1
     3.0.0.0/24 is subnetted, 1 subnets
D   4.4.4.0 [90/2297856] via 10.7.0.18, 02:44:17, Serial0/3
```

Determining Network Resources Needed for Implementing EIGRP

Figure 7 shows how EIGRP fits in your network design.

My network requires...	EIGRP features…	Which provides…
Low CPU usage	The DUAL algorithm, triggered updates	Reduced overhead
Fast Convergence	Customizable Hello and Hold timers	Customizable route failover times
Bandwidth Conservation	Customizable Hello timers and partial table updates, unequal-cost interface load balancing, metric optimization, multicast messages with unicast message retransmission	Efficient link utilization over a variety of links
Ease of Configuration	Autosummarization, robust default values for timers and metrics, lack of "areas" as found in OSPF	Reduced complexity for smaller networks
Customizable Routes	Static routes, manual route summarization, passive interfaces, configurable k-values, unequal-cost load balancing	Powerful features for larger networks
IPv6	Multiprotocol routing	Scalable address space, next-generation IP support
Security against some DoS attacks	MD5 with Pre-Shared Keys	Neighbor Authentication
Exclusive use of Cisco Routers	EIGRP is Cisco proprietary	A single-vendor solution

Figure 7: EIGRP requirements analysis and features

Study Suggestion: Table Figure 7 shows how EIGRP meets your network design requirements. Being able to narrate these properties from memory will help you in the network design and validation section of the exam.

Creating an EIGRP Implementation Plan

An example network topology is given in Figure 8 with a set of design requirements below.

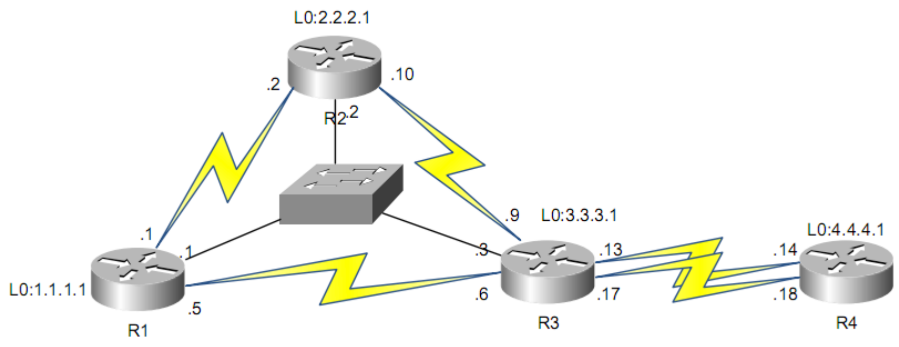

Figure 8: EIGRP Implementation

Example. All serial interfaces begin with 10.7.0. All FastEthernet interfaces begin with 192.168.0. The ASN is 99.

In this example network, we have determined our EIGRP configuration must meet the following requirements:

1. Create a converged network with ASN 99 so that each subnet on each router has a path to each other subnet.
2. Use MD5 authentication to increase security.
3. Use unequal-cost load balancing over the links between R3 and R4. On R4, S0/0 has 2000 kbit bandwidth, S0/1 has 128 kbit bandwidth.
4. Assume the 192.168.0.0/24 network is for management and ensure EIGRP does not route traffic through this network.
5. Reduce the convergence time by 60% of the defaults in the event of a successor failure.
6. Limit EIGRP traffic to 10% of bandwidth available on R4's serial 0/0 interface.
7. Summarize R4's 4.4.0.0 networks to downstream routers using an efficient summarization method.
8. Do not advertise the 1.1.1.0 /24 network from R1.

Creating an EIGRP Verification Plan

Design Requirement Number	Design Requirement Description	Verification Method	Passing Criteria
1	Create a converged network for ASN 99 so that each subnet on each router has a path to each other subnet	ping	Pings should success between networks if the network is converged.
2	Use MD5	No authentication errors occur in our converged network	MD5 keychains are applied
3	Use unequal-cost load balancing over the links between R3 and R4	show ip route	Both routes should be seen in the routing tables of R3 and R4.
4	Ensure EIGRP does not route traffic through the 192.168.0.0 network.	show ip route	Be sure the paths to the 192.168.0.0 network are not tagged as 'D' entries
5	Reduce the convergence time by 60% of the defaults in the event of a successor failure.	show ip eigrp neighbors	Verify the hold timers have been optimized
6	Limit EIGRP traffic to 10% of bandwidth available on R4's serial 0/0 interface.	show ip eigrp interfaces	Verify that the pacing time is reduced appropriately
7	Summarize R4's 4.4.0.0 networks to downstream routers using an efficient summarization method.	show ip route on R3	Verify that 4.4.0.0 is being routed with a /20 mask instead of multiple 4.4.0.0 networks with /24 masks
8	Do not advertise the route 1.1.1.0 on R1	show ip route on R2, R3, R4	The 1.1.1.0 network should not exist

Figure 9: EIGRP Design Requirements

Configuring EIGRP Routing

Beginning to configure EIGRP consists of two components:

1. The **router eigrp ASN** command defines an instance of EIGRP for a specific AS from 1-65,535.
2. The **network IP INVERSEMASK** command tells the router to begin using EIGRP for the network specified by **IP** for a given inverse mask. An inverse mask is also known as a wildcard mask. As a CCNA, the inverse mask should be familiar to you through your use of Access Control Lists (ACLs). Figure 10 illustrates enabling EIGRP on a router.

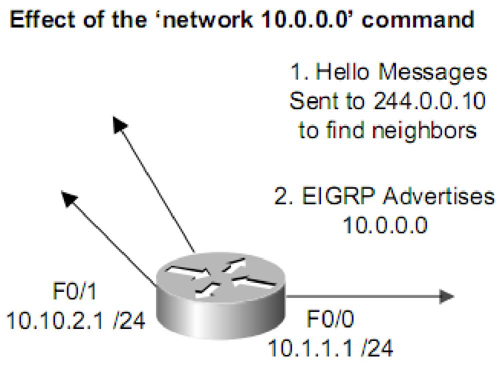
Effect of the 'network 10.0.0.0' command
1. Hello Messages Sent to 244.0.0.10 to find neighbors
2. EIGRP Advertises 10.0.0.0
F0/1 10.10.2.1 /24
F0/0 10.1.1.1 /24

Figure 10: Effect of the network 10.0.0.0 Command.
Also note that split horizon ensures routes aren't advertised out of their initiating interfaces.

You may also specify the IP without a mask. The router will automatically default to the IP's classful mask. In this case, the **network 10.1.0.0 command would be equal to network 10.0.0.0 0.255.255.255**.

Tweaking EIGRP Parameters

The **bandwidth** and **delay** interface commands are the preferred method of modifying a route's metric. This is much easier than modifying the k-values because they must be modified on all routers in the network. The **bandwidth** and **delay** subcommands tell EIGRP parameters of your choice, and can be used for tuning EIGRP's topology table and consequently the route table. Changing the bandwidth or delay has no effect on the actual capabilities of an interface.

Our design for this example calls for reducing the convergence time by 60% from the defaults in the event of a link failure. This is done by adjusting the Hello and Hold timers. In our example network Hello messages are sent every 5 seconds because the links are T1 or faster speed. A router will wait for 3 unacknowledged Hellos before the Hold timer expires and the successor is declared invalid and an FS is used.

```
R1(config-if)#ip hello-interval eigrp 99 2

R1(config-if)#ip hold-time eigrp 99 6
```

Applying this command to each serial interface in our network will meet the design requirement of reducing convergence time by 60% if a successor fails.

Limiting EIGRP Traffic

Design requirement #6 specifies limiting the amount of traffic EIGRP can consume. To do this, use the **ip bandwidth-percent eigrp ASN PERCENT** interface command. In EIGRP terminology, this technique is known as bandwidth pacing. It specifies the amount of bandwidth allowed for EIGRP itself, not protocols routed by EIGRP. It has no effect on metric calculation. As links have gotten faster, the importance of this command has diminished. Our design specification calls for limiting EIGRP bandwidth to 10% on R4's serial 0/0 link. Applying this to the interface will meet that design requirement.

Load Balancing

If two routes to a network have the same minimum FD, they will both be entered into the routing table. Up to 16 paths can be used. This is adjustable with the **maximum-paths** command in the EIGRP context.

Although not automatically balanced like equal-cost paths, load balancing for unequal cost paths can be easily configured in EIGRP. There are two ways to do this. First, an administrator can tweak the bandwidth and delay or k-weights for an interface until the FDs of both links are equal. This method is time consuming and error-prone. To make this easier, Cisco recommends using the **variance** command. Variance provides a multiplier that EIGRP can use to raise the threshold of FDs when it decides which routes to put in the routing table.

Variance is a whole number from 1-128. It is set to 1 by default, indicating that only the smallest and equal FD successors will become routes. Changing this value to 2 allows FDs within 2 times the smallest FD to go to the routing table. Changing this value to 3 allows FDs within 3 times the smallest FD to go to the routing table, and so on.

Our design plan also calls for unequal cost load balancing between routers R3 and R4. Before using variance, only the best route is added to the routing table. On router R3, the routing table is as follows:

```
R3# show ip route
...
D     4.0.0.0/24 is subnetted, 1 subnets
       4.4.4.0 [90/659968] via 10.7.0.14, 00:00:02, Serial0/2
...
```

Show the topology table to find out the FDs of routes. From the topology table, we see our 2 routes and their FDs. Because the FD of the best route is 3.4 times smaller than the route we want to balance over, we can use variance 4 to tell the router to include the other route in the routing table.

After applying **variance 4** and **clear ip route** *, the routing table is regenerated with load balancing to the 4.4.4.0 network. Remember to use **clear ip route** * after changing variance to reinitialize the route table.

```
R3(config)#router eigrp 99
R3(config-router)#variance 4
R3# clear ip route *
R3# show ip route
...
D       4.4.4.0 [90/2297856] via 10.7.0.18, 00:00:01, Serial0/3
                [90/659968] via 10.7.0.14, 00:00:01, Serial0/2
...
```

Even with variance, if a route doesn't meet the feasibility condition it won't be used for unequal cost LB. If this question appears on the exam, be sure to verify that EIGRP has actually added the route to the topology table. If it hasn't, it is because the AD is greater than the FD and EIGRP is assuming there is a loop.

Excluding EIGRP on an Interface

Excluding EIGRP on an interface has two meanings. First, we can stop messages from being sent and received on an interface by using the **passive-interface** command. This command prevents relationships from being formed on that interface. Secondly, route filtering, discussed later in this chapter, keeps a network from being advertised.

In our design requirements, we have specified that we do not want the routers' FastEthernet interfaces to form relationships. To this end, we can prevent this by not issuing the **network** command for 192.168.0.0 However, we want to make sure another administrator does not make this mistake. After issuing this command on our routers, we can be sure that no neighbor relationships will take place through the fa0/0 network.

```
R(config)#router eigrp 99

R(config-router)#passive-interface FastEthernet 0/0
```

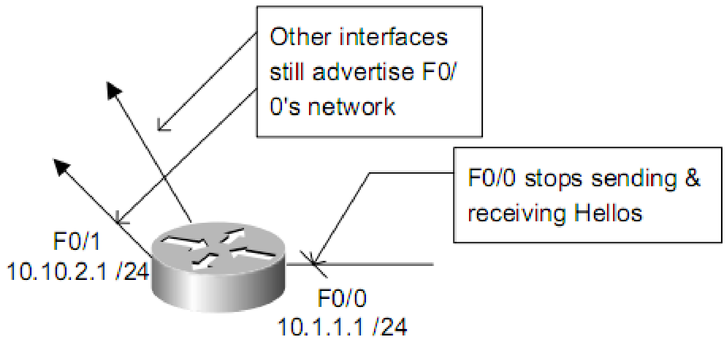
Other interfaces still advertise F0/0's network
F0/0 stops sending & receiving Hellos
F0/1
10.10.2.1 /24
F0/0
10.1.1.1 /24

Figure 11: Using passive-interface on F0/0

EIGRP Authentication

In our design requirements, we have specified that the network should not be susceptible to man-in-the-middle attacks or Denial of Service (DoS) attacks. EIGRP can be configured to authenticate peer routers before trusting neighbors and their routing advertisements. Without this, a rogue router can become a neighbor and poison the routing table, causing service disruption. EIGRP uses a **key chain** to store Pre-Shared Keys (PSK) which are exchanged using an MD5 hash. These are the same key chains used in RIPv2 authentication. Valid keys in the key chains of neighboring routers permit the routers to maintain neighbor relationships or create new ones, while invalid keys do not. Multiple keys can be concurrently valid.

In this example, we first create a **key chain** called **ourkeys**. The first interface command sets the EIGRP instance for AS 99 to require authentication. The second command establishes what keys are valid, which are the keys listed in **ourkeys**. Note that these are interface subcommands, and therefore must be used for pairs of interfaces.

Command	Description
R1(config)#key chain ourkeys	Creates a keychain
R1(config-keychain)#key 1	Specifies a key
R1(config-keychain-key)#key-string my_psk	Uses "my_psk" as our key's string
R1(config)#int serial 0/0	
R1(config-if)#ip authentication mode eigrp 99 md5	Tells EIGRP ASN 99 to enable authentication
R1(config-if)#ip authentication key-chain eigrp 99 ourkeys	Tells EIGRP ASN 99 which key chain to use

Figure 12: EIGRP Authentication Commands

Applying authentication to each interface in our example topology will meet the criteria for requirement #2, which was to authenticate neighbors to enhance security.

Stub Routers

EIGRP routers can be stub routers. This is common in a hub-and-spoke topology where the only route back to the network from a stub is the default route. If a router does not pass traffic for other networks, it is a stub router. The advantage of configuring a router as a stub router is that a hub won't Query EIGRP stub routers for successors, reducing the amount of processing and network traffic. Because stub routers in a spoke are only connected back to the hub, they will not have backup routes to offer to the hub.

The only networks that stub routers advertise by default are summary routes and directly connected networks.

```
R1(config)#router eigrp 99
R1(config-router)#eigrp stub ?
  connected      Do advertise connected routes
  leak-map       Allow dynamic prefixes based on the leak-map
  receive-only   Set IP-EIGRP as receive only neighbor
  redistributed  Do advertise redistributed routes
  static         Do advertise static routes
  summary        Do advertise summary routes
  <cr>
```

The **eigrp stub** command can be issued with defaults, or it can be tweaked to advertise different route types. Since our design document does not call for stub routers to be specified, we will not configure them in our example.

No Auto-Summary

Like RIPv2, EIGRP will automatically summarize networks by default using the classful boundary. If your network has discontiguous network space, this will result in suboptimal network routing. A discontiguous network space is when major (classful) networks are spread across different boundaries, making it difficult or impossible to effectively summarize. In Figure 13, note that router 3 receives two routes for the 10.0.0.0 class-A network. Since it now has summary routes to 10.0.0.0, it may route traffic to R1 that is actually destined for R2, and vice versa. This will result in slow performance as traffic must be re-sent on average 50% of the time. Use the **no auto-summary** EIGRP command to disable route summarization. This is also important for discontiguous network spaces with other routing protocols routes, not just EIGRP.

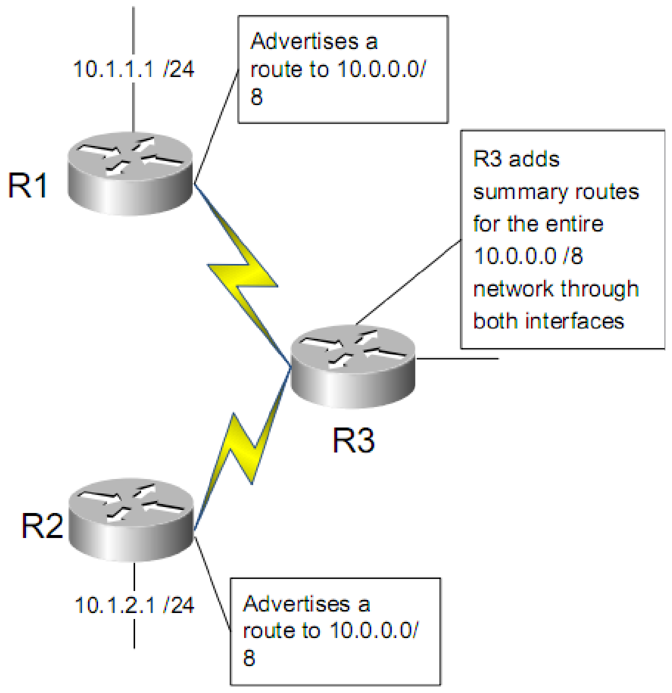
10.1.1.1 /24
Advertises a route to 10.0.0.0/ 8
R1
R3 adds summary routes for the entire 10.0.0.0 /8 network through both interfaces
R3
R2
10.1.2.1 /24
Advertises a route to 10.0.0.0/ 8

Figure 13: Auto-Summary Causes Suboptimal Routing.
Note that auto-summary is enabled by default on EIGRP.

If we didn't use **no auto-summary** in this chapter's lab topology, R3 would receive a summary route for the 10.0.0.0/8 network from each of the other routers. Packets destined for these networks could go to the wrong router. Because a summary EIGRP route has an administrative distance of 5 and an internal EIGRP route has an administrative distance of 90 the summary route would be used. Using **no auto-summary** will result in better network performance since the routes sent to R3 will be more granular.

Summarizing Routes Manually

Route summary can provide the following:

1. Smaller routing tables and smaller updates
2. Reduced CPU and memory overhead
3. Minimized effects of interface flapping: if an interface is flapping and the interface is part of a summary route, updates won't propagate through the entire network.
4. Prevented Stuck-In-Active situations
5. Hidden internal network structure from downstream routers

Assume that you need to add 10 networks to the infrastructure behind router R4. These networks will be numbered 4.4.5.0 - 4.4.15.0, each with a /24 bit mask. You can add the networks and EIGRP will automatically begin routing them, but our design document calls for R4 to send summary routes. To this end, we must tell EIGRP to manually summarize the routes for these networks.

Before summarization, R3's routing table from R4 looks like this:

```
R3#show ip route eigrp
     4.0.0.0/24 is subnetted, 12 subnets
D       4.4.4.0  [90/659968] via 10.7.0.14, 02:27:42, Serial0/2
D       4.4.5.0  [90/659968] via 10.7.0.14, 00:03:21, Serial0/2
D       4.4.6.0  [90/659968] via 10.7.0.14, 00:03:11, Serial0/2
D       4.4.7.0  [90/659968] via 10.7.0.14, 00:02:50, Serial0/2
D       4.4.8.0  [90/659968] via 10.7.0.14, 00:02:31, Serial0/2
D       4.4.9.0  [90/659968] via 10.7.0.14, 00:02:10, Serial0/2
D       4.4.10.0 [90/659968] via 10.7.0.14, 00:01:59, Serial0/2
D       4.4.11.0 [90/659968] via 10.7.0.14, 00:01:48, Serial0/2
D       4.4.12.0 [90/659968] via 10.7.0.14, 00:01:39, Serial0/2
D       4.4.13.0 [90/659968] via 10.7.0.14, 00:01:25, Serial0/2
D       4.4.14.0 [90/659968] via 10.7.0.14, 00:01:15, Serial0/2
D       4.4.15.0 [90/659968] via 10.7.0.14, 00:01:06, Serial0/2
```

The first step in determining an accurate summary is to find the mask that will encompass all of the routes as efficiently as possible. Because our networks are numbered 4.4.4.0 - 4.4.15.0, the smallest subnet mask we can use is 255.255.240.0. The IP address range for this subnet mask would be 4.4.0.0 to 4.4.15.254. Note that summary routes use a normal subnet mask and not a wildcard mask.

The second step in manual summarization is to use the **summary-address** interface command. The interface used should be the one connected to the downstream router.

```
R4(config-if)# ip summary-address eigrp 99 4.4.0.0 255.255.240.0
```

The following messages are logged on R3 and R4, respectively:

```
*Mar  2 06:48:05.100: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 99: Neighbor 10.7.0.18
(Serial0/3) is resync: peer graceful-restart
*Mar  2 06:46:27.388: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 99: Neighbor 10.7.0.13
(Serial0/0) is resync: summary configured
```

R3's routing table now looks like this, indicating that all of our routes have been summarized into one routing table entry:

```
          4.0.0.0/20 is subnetted, 1 subnets
D            4.4.0.0 [90/659968] via 10.7.0.14, 00:00:43, Serial0/2
```

Route Filtering

Although route filtering is relevant to all routing protocols, we will first introduce it in EIGRP. A route should be filtered when advertising it to other routers would violate the design requirements. This may be for reasons of security or router topology optimization. To prevent a route from being advertised, we can filter the route with the Access Control List (ACL), the prefix-list or the route-map.

The Access Control List (ACL) is given first as it is probably the most familiar from your CCNA studies. ACLs are a common method that IOS uses to categorize and apply rules to traffic. In route filtering, ACLs that **permit** traffic allow routes to be sent or received (this is the default) while ACLs that **deny** traffic filter routes from being processed. Routes can be filtered either on the incoming or outgoing direction using the **distribute-list** command and an interface, if desired.

In our design requirement, we have indicated that we do not want the other routers in our topology to have a route to the 1.1.1.0 network. The configuration below shows how to configure and verify this.

```
R2#show ip route
(lines omitted)
     1.0.0.0/24 is subnetted, 1 subnets
D       1.1.1.0 [90/2809856] via 10.7.0.9, 00:00:01, Serial0/1
(lines omitted)

R1(config)#access-list 1 deny 1.1.1.0 0.0.0.255
R1(config)#access-list 1 permit any
R1(config)#router eigrp 99
R1(config-router)#distribute-list 1 out

R2#show ip route
(lines omitted)
     2.0.0.0/24 is subnetted, 1 subnets
C       2.2.2.0 is directly connected, Loopback0
     3.0.0.0/24 is subnetted, 1 subnets
D       3.3.3.0 [90/2297856] via 10.7.0.9, 00:02:29, Serial0/1
     4.0.0.0/20 is subnetted, 1 subnets
D       4.4.0.0 [90/2809856] via 10.7.0.9, 00:02:29, Serial0/1
(lines omitted)
```

The second method for filtering routes is using the **ip prefix-list** command. The basic usage of the **ip prefix-list** router subcommand is as follows:

```
R1(config)#ip prefix-list ourlist deny 1.1.1.0/24
R1(config)#router eigrp 99
R1(config-router)#distribute-list prefix ourlist out

R2#show ip route
(lines omitted)
     2.0.0.0/24 is subnetted, 1 subnets
C       2.2.2.0 is directly connected, Loopback0
     3.0.0.0/24 is subnetted, 1 subnets
D       3.3.3.0 [90/2297856] via 10.7.0.9, 00:41:08, Serial0/1
     4.0.0.0/20 is subnetted, 1 subnets
D       4.4.0.0 [90/2809856] via 10.7.0.9, 00:41:08, Serial0/1
(lines omitted)
R2#
```

An important design note is that using prefix-list instead of ACL allows more flexibility. With prefix-list, you can match multiple networks based on the length of the prefix. For example, if you wanted to stop advertisement of 1.1.1.0 /24 but allow advertisement of 1.1.1.0 /28, you could use the expanded form of the **prefix-list** command, which includes ge (greater than or equal) and **le** (less than or equal) clauses. With these, you can specify prefix limits on permits or denies.

The final method for route filtering is through the **route-map** command. Route maps allow you to match prefixes to permit or deny, similar to ACLs. Unlike ACLs, they allow you to have aggregate ACL and prefix-lists together to form compound entries. They are numbered like ACLs, but allow for multiple lines within a single **permit** or **deny** entry. A route-map allows you to embed ACL and prefix-list logic inside a statement. The route-map's **permit** or **deny** will decide if a route gets filtered or not.

Consider the scenario where we have 2 ACLs with complex entries. We can separate all of our allow lines into one access-list and our deny lines into the other. Our route-map then merges these two ACLs and applies them sequentially based on the sequence we give them. Think of the route-map as a compound set of ACLs and prefix-lists. The route-map takes precedence over permitting or denying traffic compared to the ACL, while the ACL is simply used to catch the traffic using its permit or deny statements.

Command	**Description**
`route-map our_map deny 5`	Defines a route-map named "our_map" that will deny matched entries. It is given sequence number 5.
`match ip address my_bad_acl`	All addresses matched by the ACL named "my_bad_acl" will be denied.
`route-map our_map permit 10`	Adds a permit clause to the route-map called "our_map". It is given sequence number 10 in this case.
`match ip address my_good_acl`	All addresses matched by the ACL named "my_good_acl" will be permitted.
`router eigrp 99`	Our EIGRP instance.
`distribute-list route-map our_map out`	Apply the route-map in the outbound direction.

Figure 14: EIGRP Route Filtering Commands

You should be familiar with ACLs, prefix-lists, and route-maps for all protocols on the exam, not just EIGRP. Be prepared not only to create and apply a route-map, but also to design and verify them.

Verifying Proper EIGRP Implementation

Compared to previous exams, this revision of the CCNP exams places greater emphasis on verification that a system is running as intended. A network runs "as intended" if it effectively meets the requirements specified in the network design. To verify this basic EIGRP network, start at the routing topology and work backwards to the neighbor relationships. Ask yourself the following questions:

1. Has EIGRP discovered routes to all of my networks? In other words, has the network converged as expected?
2. Is my topology correct? Are my FSs present as expected? Are my FDs correct?
3. Did my neighbor relationships form as I desired them to?
4. Does my EIGRP network meet all of the requirements specified in my design document?

If your EIGRP network fails these tests, troubleshooting should be done in a top-down fashion, starting with verification of the neighbor relationships.

Verifying Neighbor Relationships

For the exam, be ready to identify reasons why neighbor relationships are not forming as expected. Possible reasons include authentication issues, a firewall, a link issue, excessive CPU or memory usage causing the upstream router not to reply, or an EIGRP configuration issue such as mismatched timers or weights. Also, keep in mind that a serial link may not be configured to send broadcasts by default, therefore causing a link error since Hellos are not sent. Figure 15 illustrates these common issues. Be prepared to diagnose these and other neighbor relationship problems for the exam.

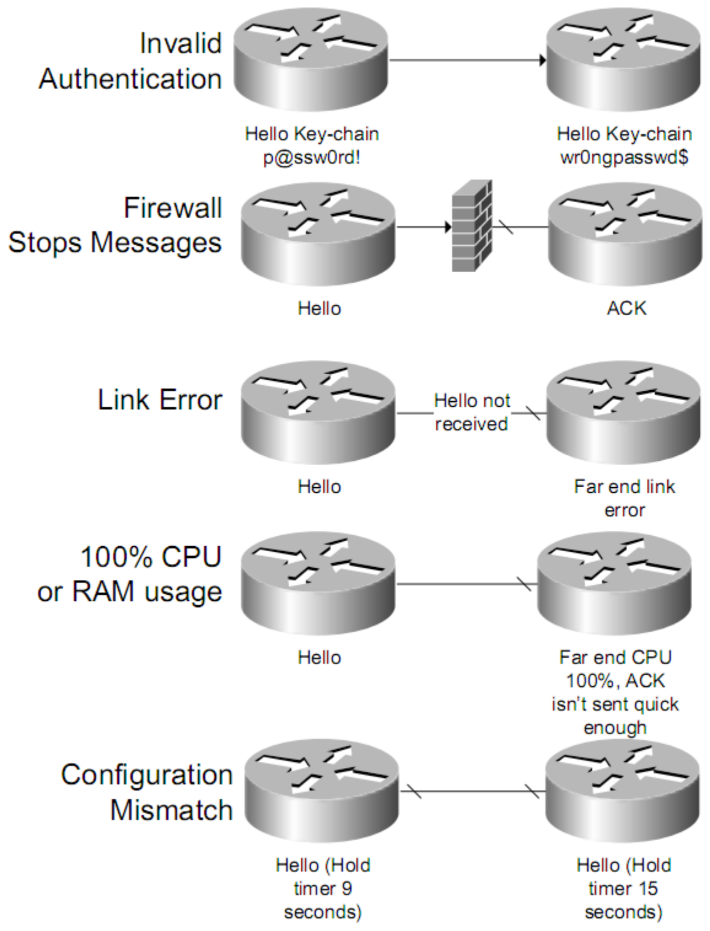
Invalid Authentication
Hello Key-chain p@ssw0rd!
Hello Key-chain wr0ngpasswd$
Firewall Stops Messages
Hello
ACK
Link Error
Hello not received
Hello
Far end link error
100% CPU or RAM usage
Hello
Far end CPU 100%, ACK isn't sent quick enough
Configuration Mismatch
Hello (Hold timer 9 seconds)
Hello (Hold timer 15 seconds)

Figure 15: Reasons why EIGRP neighbors may not form correctly.

Interpreting the neighbor table is the best way to troubleshoot neighbor issues. Before doing this, verify that EIGRP is enabled on the interfaces you want using the **show ip eigrp interfaces** command. An interface must be up/up to appear in the interfaces table.

Occasionally, a router may become Stuck In Active (SIA). An SIA error looks like this:

```
MAR 10 12:11:06: %DUAL-5-NBRCHANGE: IP-EIGRP 99: Neighbor 10.7.0.6
(S0/1) is down: stuck in active
```

SIA can have a number of causes and may occur in the following situation:

1. The router has a valid relationship with a neighbor router.
2. The router loses a route, either because the Hold time has expired or due to another issue.
3. The router has no known FS to the route it has lost, so it sends a Query message asking for a new route.
4. No neighboring router has responded with a route. The topology table shows an 'A' for Actively searching.
5. After a given period of time (3 minutes by default) with no Replies received, the router delivers %DUAL-3-SIA: Route network mask stuck-in-active state in IP-EIGRP 99. Cleaning up and the neighbor relationship is terminated.

```
MAR 10 12:14:23: %DUAL-3-SIA:Route 2.2.2.0 /24 stuck-in-active state
in IP-EIGRP 99. Cleaning up.
```

```
R3#show ip eigrp interfaces
IP-EIGRP interfaces for process 99

                        Xmit Queue   Mean   Pacing Time   Multicast    Pending
Interface        Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Se0/0              1        0/0       110       0/15         487          0
Se0/1              1        0/0        88       0/15         391          0
Se0/2              1        0/0       102       0/15         447          0
Se0/3              1        0/0        94       0/15         407          0
Lo0                0        0/0         0       0/1            0          0
```

If all interfaces are correct, verify neighbor relationships with the **show ip eigrp neighbors** command.

```
R3#show ip eigrp neighbors
IP-EIGRP neighbors for process 99
H   Address              Interface       Hold Uptime    SRTT   RTO  Q  Seq
                                         (sec)          (ms)       Cnt Num
3   10.7.0.10            Se0/1             14 00:01:23   88   528  0  14
2   10.7.0.5             Se0/0             12 00:03:47  110   660  0  15
1   10.7.0.18            Se0/3             14 00:04:21   94   564  0  24
0   10.7.0.14            Se0/2             14 00:04:21  102   612  0  23
```

In this example, we see that all relationships have formed as expected. The H value is the order in which the relationships were established. We can also see the Hold time in seconds, and how long the relationship has been established. The SRTT is the Smooth Round Trip Time and tells us the round-trip number of milliseconds an EIGRP message Hello and Acknowledge message takes. The Seq Num is the sequence number of the last update, query, or reply message received from this neighbor.

To verify that messages are being sent and received by an EIGRP process, use the router command **show ip eigrp traffic**. This command yields the following output:

```
R3#show ip eigrp traffic 99
IP-EIGRP Traffic Statistics for AS 99
  Hellos sent/received: 52595/52569
  Updates sent/received: 90/83
  Queries sent/received: 26/16
  Replies sent/received: 18/29
  Acks sent/received: 118/113
  SIA-Queries sent/received: 0/0
  SIA-Replies sent/received: 0/0
  Hello Process ID: 246
  PDM Process ID: 245
  IP Socket queue:   0/2000/5/0 (current/max/highest/drops)
  Eigrp input queue: 0/2000/5/0 (current/max/highest/drops)
```

Debugging EIGRP messages is useful when neighbor relationships aren't forming as expected. Use the command **debug eigrp packets TYPE** where **TYPE** is the message type to analyze.

Note: Debugging commands take top priority in IOS and can cause your router to become unstable. Turn off debugging using the **no debug all**, **undebug all** , or **u all** command.

The **debug eigrp packets** output below shows three issues that could be causing neighbor relationships to form. First, we see authentication is not being sent by a far-end router. Secondly, authentication is present but the keys do not match. Third is a k value mismatch.

```
*Mar  1 14:56:00.768: EIGRP: Serial0/0: ignored packet from 10.7.0.5,
opcode = 5 (missing authentication)

*Mar  2 00:22:38.525: EIGRP: Serial0/0: ignored packet from 10.7.0.1,
opcode = 5 (invalid authentication)

*Mar  1 15:11:19.788: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 99: Neighbor
10.7.0.2 (Serial0/0) is down: K-value mismatch
```

Hello and Hold timer mismatches will be more obvious and result in the link coming up and going back down repeatedly. This is known as flapping. While one router determines that a relationship should be dropped due to expiration of the Hold timer, the other router will determine that the link should stay up. This can also occur if the Hold timer is accidentally set smaller than the Hello timer. Hello and Hold timers should be the same on all routers, with the Hold timer representative of the desired convergence speed and the acceptability of lost Hello messages.

Verifying the Topology

The main command for verifying the topology is the **show ip eigrp topology** command.

```
R1#show ip eigrp topology
IP-EIGRP Topology Table for AS(99)/ID(1.1.1.1)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.7.0.12/30, 1 successors, FD is 2681856 via 10.7.0.6 (2681856/2169856),
Serial0/1
P 1.1.1.0/24, 1 successors, FD is 128256 via Connected, Loopback0
P 2.2.2.0/24, 1 successors, FD is 2297856 via 10.7.0.2 (2297856/128256),
Serial0/0
P 3.3.3.0/24, 1 successors, FD is 2297856 via 10.7.0.6 (2297856/128256),
Serial0/1
P 4.4.4.0/24, 1 successors, FD is 2809856 via 10.7.0.6 (2809856/2297856),
Serial0/1
P 10.7.0.8/30, 2 successors, FD is 2681856
    via 10.7.0.2 (2681856/2169856), Serial0/0
    via 10.7.0.6 (2681856/2169856), Serial0/1
P 10.7.0.4/30, 1 successors, FD is 2169856 via Connected, Serial0/1
P 10.7.0.0/30, 1 successors, FD is 2169856 via Connected, Serial0/0
P 10.7.0.16/30, 1 successors, FD is 2681856 via 10.7.0.6 (2681856/2169856),
Serial0/1
```

In the first column, we see that each feasible successor is listed as P for Passive. This is actually good. A P route means that this route is ready to be sent to the routing table, if needed. An A status means the route is being actively calculated. In other words, the router is Querying neighbors trying to find a neighbor that has a valid route. It is waiting for a Reply.

Let us break down a single entry from the topology table:

P	2.2.2.0/24	1 successors	FD is 2297856	via 10.7.0.2	(**2297856**/128256)	Serial0/0
P means routes are ready if needed	This is the destination network	The number of successors to this network	The calculated distance to this network	The next-hop destination	The feasible distance (in bold) and the reported distance.	The exit interface

Figure 16: A Topology Table Entry

Because the FD of 2297856 is the lowest FD calculated, this route will go to the routing table for the 2.2.2.0 /24 network.

Verifying the Routing Table

Our design criteria specified several routing table results. Unequal cost load balancing, route filtering, and convergence were given specific requirements.

The main command used for all routing protocols to examine their performance is the show ip route command. For R1, the output is given below.

```
R1(config-router)#do sir
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-
IS level-2, ia - IS-IS inter area, * - candidate default, U - per-user
static route, o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
     2.0.0.0/24 is subnetted, 1 subnets
D       2.2.2.0 [90/2297856] via 10.7.0.2, 01:44:01, Serial0/0
     3.0.0.0/24 is subnetted, 1 subnets
D       3.3.3.0 [90/2297856] via 10.7.0.6, 01:44:01, Serial0/1
     4.0.0.0/20 is subnetted, 1 subnets
D       4.4.0.0 [90/2809856] via 10.7.0.6, 01:44:01, Serial0/1
     10.0.0.0/30 is subnetted, 5 subnets
D       10.7.0.12 [90/2681856] via 10.7.0.6, 01:44:01, Serial0/1
D       10.7.0.8 [90/2681856] via 10.7.0.6, 01:44:01, Serial0/1
                 [90/2681856] via 10.7.0.2, 01:44:01, Serial0/0
C       10.7.0.4 is directly connected, Serial0/1
C       10.7.0.0 is directly connected, Serial0/0
D       10.7.0.16 [90/2681856] via 10.7.0.6, 01:44:01, Serial0/1
C    192.168.0.0/24 is directly connected, FastEthernet0/0
R1(config-router)#
```

The show ip protocols command is useful for debugging EIGRP routes. The highlighted output below shows useful EIGRP debugging data.

```
R1#show ip protocols
Routing Protocol is "eigrp 99"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 99
  EIGRP NSF-aware route hold timer is 240s
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    1.0.0.0
    10.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    (this router)         90      1d02h
    10.7.0.6              90      1d02h
    10.7.0.2              90      1d02h
  Distance: internal 90 external 170
```

Through the use of these two commands, we should be confident that our routing table is as we would expect it.

Documenting an EIGRP Implementation

Documentation is the final step in network engineering. Documentation of any network, regardless of the complexity, size and protocols used, is important. The documentation will serve as the starting point for the next iteration of network design. It will also be the starting point for maintenance of the network. Documentation is very important for troubleshooting and should be kept up to date if a design change is made while in the maintenance phase of a network's lifetime.

In reality, a network is rarely designed ideally the first time. Network design is an iterative process. If we were unable to implement any part of the design we had previously specified, the documentation process should explain why.

A documentation plan should contain the following things:

1. Output of the debug and show commands useful in documenting the network design and implementation: This should include highlights that confirm the network is operating as intended.
2. An explanation of why a particular design requirement was met or not met, with sample output
3. Data to look for if the network does not behave as expected

Domain 2: Implementing a Multi-Area OSPF Network

Introduction

Open Shortest Path First (OSPF) is a open standard link-state Interior Gateway Routing Protocol. The latest version of OSPF is Version 3, which includes IPv6 support and is defined in RFC 5340. Cisco's implementation adds some proprietary features which will be identified as they are introduced.

OSPF is an attractive routing protocol, particularly in multi-vendor networks. It is quick to converge and naturally supports a hierarchical design which makes summarization and route filtering easy. Bandwidth is conserved by using multicast and incremental updates.

In general, the task of finding the shortest path between adjacent routers is a combinatorial problem with no perfect real-world solution. As the number of routers grows, it becomes exponentially harder to find the optimal route. Therefore, OSPF designs strive to keep routers grouped into logical arrangements called *areas*. The smaller we can keep the number of potential routes, the more efficient OSPF will run. OSPF, and an OSPF design, must compensate for finite router CPU and RAM.

The main component of OSPF is the Shortest Path First algorithm. SPF finds optimal or near-optimal routes in a network. Because the Shortest Path First algorithm is CPU-intensive, OSPF uses the concept of *areas* to divide a network and control which routes are available to routers in that area. Areas limit message propogation and help reduce SPF calculations. An area is a logical hierarchy where network routes can be summarized before being passed to the next area for processing. This divide-and-conquer strategy requires a more in-depth network design than other protocols such as EIGRP, but allows for greater scalability by breaking networks down. The SPF algorithm will do its best to find routes, but a poorly designed network hierarchy can inhibit router performance and OSPF operation.

Because of the complexity and vocabulary of OSPF, mastering it for the exam and in the real world can be overwhelming. However, OSPF is similar to EIGRP in that it has 3 major components: the message exchange, the link-state database (LSDB), and the routes. These are parallel to EIGRP's message exchange, the topology table, and finally the routes. Since OSPF itself uses a divide-and-conquer strategy, this chapter will attempt to divide and conquer the complex task of designing, implementing and verifying an OSPF internetwork. First we will break OSPF down in to its components and explain how they interact. Then we will show how OSPF is configured and verified in an internetwork.

OSPF Areas

Before describing the specific types of router roles and messages, we must first formally introduce an area. An area is a logical division of a network composed of contiguous routers running OSPF. The purpose of the area is to break the routers into groups so they can efficiently run SPF. However, they also contain message flooding to a reasonable amount. Additionally, areas are designed to limit the scope of network changes. Through route summary, a route change in one area will not affect the summary route in another area. If R1 is in Area 1 and R2 is in Area 2, a link coming up or going down on R1 will not affect R2, because R2 has no knowledge of Area 1's internal routes.

Every OSPF network must contain an Area 0. This network is called the backbone network. All areas must contain at least one router with an interface (physical or logical) in Area 0. If an area has no physical connection to Area 0, a virtual (logical) link can be configured as an interface between areas. Inter-area traffic should cross the backbone. A backbone router is any router with an interface in Area 0.

All routers in an area contain the same routing table, but do not contain information about routes in other areas. Routers in other areas will have inter-area routes. OSPF routers know that to send data to another area, they should forward it to the Area Border Router (ABR), which has interfaces in both areas (or at least is a hop closer to a router with interfaces in the destination area). If summarizing or using stub areas, the only routes to external networks that internal routers know are summary routes with the ABR given as the next hop.

Figure 17 shows how areas interact in a network. We will expand on this diagram several times as more concepts are introduced.

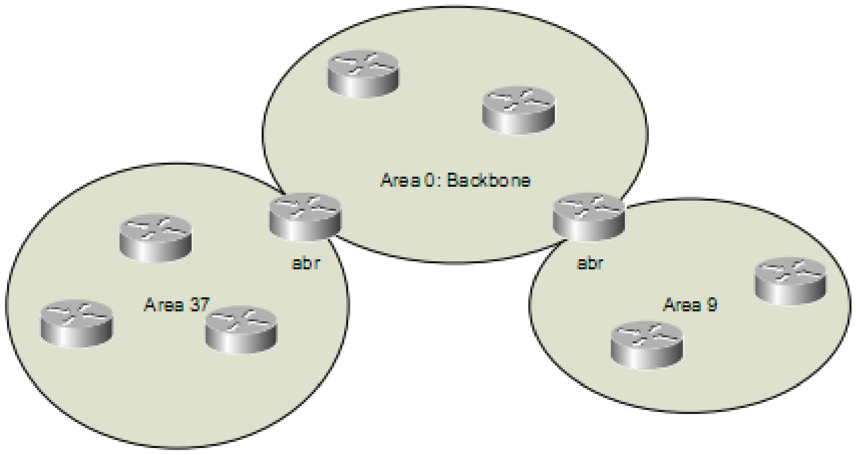

Figure 17: OSPF Areas

Inside an Area

An area is a way of dividing up larger networks: it is an abstraction to improve manageability and scalability. Inside an area is where adjacencies are created and messages are exchanged. To go inside an area, this section describes:

- 6 router types
- 2 multicast IP addresses
- 6 LSA message types
- 8 router states
- 4 interface types

Study Suggestion: You will need to be familiar with all the components of OSPF. Rather than attempting to memorize them directly, make a set of flash cards and quiz yourself for better retention.

Router Types

Table $ lists the different router types. Some routers can be multiple types at the same time. It is not uncommon for a router to be an ASBR and a DR. However, a router can't be a DR and a BDR at the same time, or a DR-Other and DR/BDR at the same time. By definition ABRs have interfaces in different areas, implying that an ABR can't be an internal router. More design constraints will become clear as we explore the different types. In Figure 18, we use the term "by design" or "by election" to describe the most common behavior. Each of these settings can be tweaked so that our routers take the roles we want them to, and in fact must be for good network design.

Determined by design	
ASBR	Area System Boundary Router. A router which interfaces with another routing protocol such as RIP or BGP. It can also be an edge device with static routes.
ABR	Area Border Router. An ABR has an interface in two or more areas; it is considered an entry or exit point to the OSPF area.
Internal	An internal router only has interfaces in one area.
Determined by election	
DR	Designated Router. In an area, one router will be elected to be the DR. The task of the DR is to gather reported routing information about subnets and update the link-state database on multi-access networks.
BDR	Backup Designated Router. This router has the same function as the DR. One router per area is elected to be the BDR on multi-access networks.
DR-Other	A DR-Other router is a router in an area that is neither a DR nor a BDR in a multi-access network.

Figure 18: Router Types and Descriptions

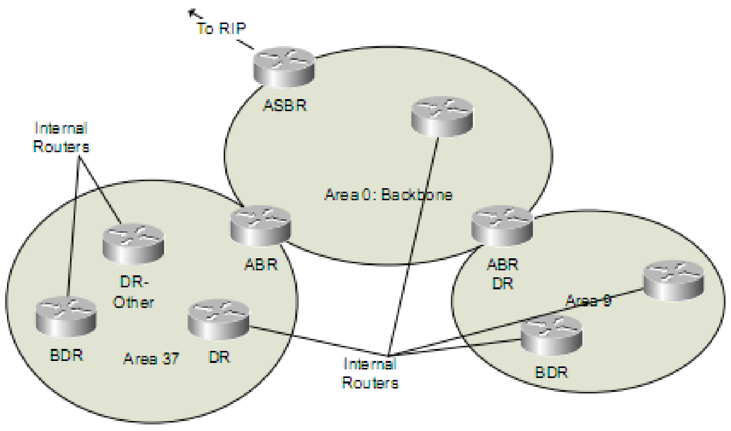

Figure 19: Router types

Initially, a network engineer will select a router to be the ASBR, which is a router interfacing with another protocol such as RIP. The ASBR summarizes OSPF routes and forwards them to the rest of the network, which is not running OSPF. An ASBR doesn't necessarily have to interface with another routing *protocol*; it can sit on the edge of your network facing the Internet with a static route. Regardless of the upstream network, the OSPF domain of control stops at the ASBR. The ASBR's job is to filter, summarize and redistribute routes with routers running other routing protocols on the network. The ASBR also summarizes the routes it receives from these routers and passes them in to the OSPF system. The only thing that other routers inside an area need to know is what to send to the ASBR and how to get it there. The ASBR will take care of the rest.

Area Border Routers (ABRs) are routers with interfaces in two or more areas. The purpose of the ABR is to act as a proxy for routes entering or exiting an area. ABRs can summarize the area's subnets and advertise this to adjacent areas. Likewise, the ABR can be configured to receive summary routes from an adjacent area. The ABRs have topology data and calculate potential routes inside each area. ABRs provide the input and output to an area, thus being quintessential to OSPF's divide-and-conquer nature.

Internal routers sit with all interfaces inside a single area. They are not ASBRs or ABRs, so they do not automatically have extended responsibilities of these types. However, an Internal router can be a DR, BDR, or DR-Other under certain conditions. We discuss DR, BDR, and DR-Other in the next few paragraphs.

The Designated Router (DR) and Backup Designated Router (BDR) are roles that also improve the efficiency of the SPF algorithm and limit network traffic. The job of the DR/BDR is to coordinate updates in an area. Instead of updates flooding to all routers and causing them to suddenly recalculate their routing tables, updates in an area are only multicast to the DR and BDR.

Imagine that a router gets an update; it processes the update and floods the update to its neighbors. If this were the case without DRs, network traffic wouldn't scale as well as it does. To illustrate the effect of having a DR, consider a broadcast subnet (like Ethernet) with n routers in it. It would require each router to know about n-1 other routers (all routers except itself) for a total of n2 - n adjacencies (a full mesh). By introducing a DR, each router only has to know about 1 adjacency (the adjacency to the DR). This reduces the number of adjacencies to n. For example, a subnet of 32 routers has a total of 992 adjacencies without DRs. By introducing a DRs, each router only needs to know how to contact the DR, so it is reduced to 32. Therefore, an update won't require 992 messages to be passed; it only requires 32 to messages to be passed because only one router (the DR) is doing the updating. On shared Ethernet, only 2 full neighbor relationships will emerge.

This special role means the DR/BDR will listen on 224.0.0.6 and receive the updates from other routers in the area. The DR and BDR are the managers of an OSPF area; when an update occurs, they will be the ones to receive it and process it. If the update is beneficial, they will update the other routers by multicasting the update to 224.0.0.5. The IP addresses 224.0.0.6 and 224.0.0.5 are the multicast addresses used by OSPF.

An area does not have to have a DR or BDR. In a point to point link, for example, there are only 2 routers. One router always updates the other. Therefore, the DR and BDR are not elected.

DR-Others are "other" routers in an area. They neither have the DR nor BDR role. However, if the DR or BDR goes down are eligible to become a DR/BDR.

As we have seen, ASBRs, ABRs, and Internal routers are determined by the way their interfaces are allocated, but how does a router become a DR or BDR?

In the DR/BDR election process, routers exchange OSPF priorities and compare them. The router with highest priority becomes the DR, and the router with the second highest priority becomes the BDR. If all priorities are equal (this is the default), then OSPF will look at the Router ID (RID). When OSPF is started, the router selects a RID, which is an IP address. The Router-ID will be determined automatically if it is not configured manually. If not explicitly defined, the RID will be the largest IP address of the router's interfaces, with loopback interfaces taking priority over physical ones. The router with the greatest RID becomes the DR. In summary, if the priority is set higher than the other routers' priorities, it will be used, otherwise the RID will be used.

This means we have 3 ways of manipulating which routers win. First, we can configure the priority with the **ip ospf priority n** command, where n is an integer from 1-255. If we use the command **ip ospf priority 0**, the router will not participate in the election process. Secondly, we can specify a Router-ID with the **router-id** router subcommand. Also, we can add a loopback interface and give it a high IP address. Finally, we could technically configure our physical interfaces on the intended DR with an IP address greater than the rest, but this would alter our IP addressing scheme, so it is not recommended. It is much easier to either configure the priority, router-id, or create a loopback interface.

The RID only changes when a router starts OSPF or is rebooted. Interfaces must be up/up for their IP addresses to be considered in RID determination. This means if you accidentally start OSPF before all your interfaces are up, you may have to reload it to get the desired RID.

The DR and BDR should be chosen manually in a network. We want to choose routers with lots of reliability, CPU and RAM, and bandwidth. In a hub-and-spoke topology, make sure that the DR and BDR are on a hub and not a spoke. Do not let the election process assign a DR to a slow, unreliable router when designing an OSPF network. Assigning your RIDs or loopback interfaces manually will also help you in troubleshooting. For instance, it is much easier to interpret the output of a **debug** or **show** command when you can quickly identify router 2.2.2.2 instead of router 68.15.148.143.

For an example, assume a router has the following values:

Priority: 1 Loopback 0: 8.8.8.8 /24 FastEthernet0: 10.10.81.8 /24 in Area 1 FastEthernet1: 10.10.82.8 /24 in Area 2 Serial 0/0: 141.10.81.8 to the internet	
Questions:	Answers:
What will the RID be?	The RID, not manually assigned with the **route-id** subcommand, will be 8.8.8.8. Although the FastEthernet 0 has a higher IP, loopbacks are given first priority.
Is this router eligible to be a DR/BDR?	Yes, assuming the 10.10.81.8 network is included in OSPF and not limited by the **passive-interface** command. We know this because Ethernet networks are multi-access "shared" segments and the priority is not 0.
Will the router be the DR?	If the other routers in the 10.10.81.0 /24 network have lower priority, or if they have lower IP addresses than 8.8.8.8, this router will win.
Is the router an ABR?	Assuming we run OSPF on both fast Ethernet interfaces, it will be an ABR because it has interfaces in both areas.
Is the router an ASBR?	Yes. Because the router has an interface to the Internet, this interface leaves our domain of control.

Figure 20: Planning an OSPF Solution

Study Suggestion: Be able to draw an OSPF network and identify the areas, routers, and messages they pass between them from a description. Identify the RIDs and how they will determine the DR/BDRs.

OSPF Messages

Like EIGRP, OSPF uses message exchange to learn the topology of the network and maintain relationships between neighboring routers. OSPF messages are sent to the 224.0.0.5 multicast address, which is known as the "All OSPF Routers" address. Messages destined for the DR/BDR are sent to 225.0.0.6, which is known as the "All DR/BDR Routers" address.

Messages in OSPF are reliably transmitted: they are acknowledged by downstream routers. Messages are sent via IP protocol 89. This is OSPF's own layer 4 protocol. OSPF messages are in the table below.

Message	**Description**
Hello	Similar to EIGRP's Hello. Multicast to 224.0.0.5
Database Description	Unicast database topology description packet
Link-State Request	LSRs request a part of the database so a router can update.
Link-State Update	LSUs are multicasts made up of LSA messages. LSUs are acknowledged by unicast.
Link-State Acknowledgement	Acknowledges an update. Note: this is not the same "LSA" referred to in the next section.

Figure 21: OSPF Messages

The OSPF Hello is similar to the EIGRP hello, but contains more data. The hello contains the RID, the timer values, and a list of adjacencies. The hello and dead timers must be the same between routers, otherwise an adjacency will cycle between INIT and DOWN, which are two of the 8 router states defined later.

Hello messages are sent every 10 seconds by default on broadcast and point-to-point interfaces. Hello messages are sent every 30 seconds on NBMA (defined in a later section). The dead-timer in OSPF is the amount of time that passes before an adjacency is declared lost. By default, it is four times the Hello timer. This means that for NBMA networks, 120 seconds will pass before we declare an adjacency to be dead. The hello and dead timers can and should be tweaked and should be for faster convergence when a route fails. For most networks, this is 10 seconds for hello and 40 seconds for dead.

Hello messages do not propagate between areas. The internal topology of an area is always hidden to routers outside of the area. This provides a layer of abstraction between SPF domains.

Like EIGRP, OSPF uses its own layer 4 protocol for communication. This is protocol number 89 and is neither TCP nor UDP. However, almost all communication with OSPF is reliably sent. Each packet except the (hello and ack) is followed by an acknowledgement that the packet was successfully received.

A Database Descriptor (DBD) packet is sent from one router to another by unicast when the router needs an update or first comes online. The DBD contains a summary of the LSDB and sequence numbers. It is used for sequence number comparison. If LSDBs need updating, the router will then send an LSR.

In OSPF, updates are sent with these rules:

1. When a router first comes online, it will receive a full copy of the LSDB.
2. As routes change, they will be sent as partial updates.
3. Every 30 minutes, OSPF "refreshes" by sending a full update to each router, just in case any routes are missing.

The Link-State Request is a packet equivalent to EIGRP's Query message. If a router is missing a route, it will send an LSR to a neighbor. The Link-State Update is the reply to the request. A Link-State Update contains one or more Link-State Advertisements. The LSA is a special type of message that routers use to advertise what resources they provide.

Link-State Advertisements

So how does a router become one or more of the router types from the last section? In the case of defining boundaries and areas, this is done by the placement and assignment of interfaces and the exchange of Router ID (RID) values. Since the main structure of OSPF is the Link-State Database (LSDB), it must be built by adjacencies. OSPF adjacencies are formed in an area through the exchange of messages called Link State Updates (LSUs). Each LSU contains multiple Link State Advertisements (LSAs).

Think of the LSA as a more detailed version of EIGRP's message schema. Both processes have the same goal, which is to discover adjacent routers and advertise routes. The main difference between EIGRP messages and OSPF messages is that OSPF messages advertise the router's state. Being a link-state protocol, OSPF keeps track of neighbors and their states in order to calculate the shortest path and paths into and out of the area. Each router in an OSPF network represents a Finite State Machine, and the states of the routers are transmitted via LSAs.

For example, ABRs recognize that they are multi-area and configure themselves as such. They will then send LSA Type 3 messages, which are specific to ABRs for advertising their role. Other routers receive the LSA and place it in their LSDB. After convergence, routers in an area know they can send data to subnets outside the area via the ABR.

Figure 22 lists the OSPF LSA types. Although other types exist, these are the most commonly used ones and are most important for the ROUTE exam.

LSA	Common Name	Purpose
LSA Type 1	Router LSA	Each router sends Type 1 LSAs to 224.0.0.5. They advertise the routers' interfaces, its adjacent routers, and their costs.
LSA Type 2	Network LSA	DRs list adjacent routers with Type 2 LSAs broadcast to 224.0.0.5. These are for broadcast-only subnets such as Ethernet.
LSA Type 3	Summary LSA	ABRs list prefixes in each area. Type 3 LSAs hold summary information of the routers using Type 1 and Type 2 LSAs. Exchanges area prefix and cost information but no topology data leaves the area.
LSA Type 4	External Summary LSA	Similar to Type 3 LSA. ASBRs send these so hosts can find them.
LSA Type 5	External LSA	ASBRs send these to advertise external routes and default routes. Not sent to stub areas.
LSA Type 7	NSSA LSA	ASBRs in an NSSA (discussed later) tunnel Type 5 LSAs inside an NSSA and convert them back to Type 5 LSAs at the ABR.

Figure 22: OSF LSA Types

LSA Subtypes

Two of the LSAs mentioned above have subtypes.

Type 5 LSAs contain routes from another routing system from an ASBR. They can be either E1 or E2 type, with E2 being the default. The metric of E1 increases at each router in the network, while E2 routes do not. The NSSA's ASBR generates Type 7 to represent the other routing domain (since Type 5 is not allowed in stubs). Likewise, Type 7 LSAs can be either N1 or N2 subtype. In N1 the metric increases with each hop, in N2 it does not.

Router States

A link-state protocol is quite literally a database of links and a database of states. The links can affect the states and the states can affect the links.

The OSPF LSDB, equivalent to the EIGRP topology table, is created and updated by LSAs and kept in memory. In link-state routing, all routers have a copy of their area's database. Each router also has a state that determines how it interacts with other routers. When a router begins to run OSPF, it goes through a series of states, given in Figure 23.

State	Description
Down	Before OSPF has received any *hello* messages, it is in this state.
Attempt	This state is for manually configured neighbors. In this state, OSPF has not received any *hello* messages.
Init	A router has received a hello message.
2-Way	DR/BDR election begins.
Exstart	Master-Slave relationships are established.
Exchange	DBDs are sent to neighbors and the database is updated.
Loading	Link-State data is being exchanged via LSR and LSU packets.
Full	An adjacency has been created and the database is synchronized.

Figure 23: OSPF Router States

When R1 joins an OSPF network, it will send a multicast hello to the upstream router. The upstream router, R2, will receive the hello and go in to an Init state. R2 will send R1 a hello in response. The routers will check the timers, authentication, area, and mask to be sure they are allowed to form an adjacency. If all of these tests pass, the routers go to the 2-Way state.

The 2-Way state is where routers elect a DR and BDR. DR-Other routers will stay in the 2-Way state. This is because in a network with a DR/BDR, DR-Other routers don't need to form adjacencies with other DR-Other routers. On a shared Ethernet segment, you will only see 2 Full adjacencies. The other adjacencies in the area will stop at the 2-Way point.

In the Exstart state, master-slave relationships are established. In a master-slave relationship, database updates can be sent and received. The master is simply the router who sends first. The master is determined by OSPF priority.

In the Loading state, the LSDB is built. The master and slave routers exchange routes via LSR and LSU packets until both routers have synchronized copies of the LSDB. When both routers have a copy of the database, they enter the Full state. This is where the routers will stay. When the routers have reached this point, they are free to run the SPF algorithm and determine the best paths through the area.

OSPF Interface Types

One issue that every OSPF network implementation plan must consider is an inherent problem in OSPF: OSPF assumes that every interface in your network can send and receive multicasts. The network cannot exchange messages without the ability to transfer multicasts or a workaround to overcome this limitation.

Study Suggestion: Rather than trying to remember which network types have a DR, remember that Point-to-Anything networks do not. Also, all types work on the same subnet except Point-to-Point.

Interface Type	Examples	Description
Broadcast	Default for Ethernet	Has a DR, Neighbors automatically discovered on same subnet
NBMA	Default on Frame Relay & ATM	Has a DR, Neighbors statically configured on same subnet
Point-to-Multipoint	Can be any interface	No DR, all routers on same subnet, neighbors automatically configured
Point-to-Point	Default on non-FR serial	No DR, each router pair forms an adjacency, each link on different subnet, neighbors automatically configured

On broadcast segments like Ethernet, configuring OSPF is easy. Ethernet networks will automatically detect the DR and BDR and discover neighbors.

Point-to-multipoint and Point-to-Point networks configure neighbors automatically, but due to the nature of multicast, no DR will be assigned because the links are not capable of having multiple interfaces participating in the election process.

NBMA networks will assign DRs, but neighbors must be statically programmed. Because multicast is a type of broadcast, routers have difficulty exchanging hello messages and forming adjacencies. This type of network, which includes frame-relay configurations, can be intimidating to a ROUTE candidate. But the main thing to know is that when designing an NBMA network, you want to make sure DR and BDRs are elected on hubs and not spokes. The DR and BDR should have a complete mesh if possible.

To statically assign neighbors, use the **neighbor *ip c*** router process subcommand where **ip** is the neighbor's address and **c** is the optional cost.

Study Suggestion: In your lab, use a topology that is large enough to utilize the different interface types to compare and contrast their features. Troubleshooting interface configurations in the lab will also help you troubleshoot on the exam.

OSPF Metric

OSPF uses bandwidth as a function of the metric. The metric is known as the *cost* in OSPF terminology. It is determined by the following:

$$Cost = \frac{100{,}000{,}000}{LinkSpeed}$$

Figure 24: OSPF Metric Equation

The value 100,000,000 is used because this represents a 100 megabit link. A 100 megabit link would have a cost of 1, while a T1 would have a cost of 100,000,000/1,544,000 = 64.7. Because metrics are always integers, this is rounded up to 65.

The cost metric in OSPF is cumulative. That is, it is the sum of all costs between the source and destination routers. In our example later in this chapter, our data must traverse several T1 links (a cost of 65) and a fast Ethernet segment (a cost of 1). We can easily add up these values to arrive at the cost that OSPF uses when calculating the metric.

The cost can also be explicitly defined with the **ip ospf cost *c*** where ***c*** is 1-65,535. If the interface subcommand ***ip ospf cost c*** command is given, it will override the calculation.

Consider the network in Figure 25. To calculate the OSPF cost, we first find the cost of each link.

Ethernet = 10,000,000	= 10	
2 megabit = 2,000,000	=50	10 + 50 + 1786 = 1846
56k = 56,000	= 1785.714	

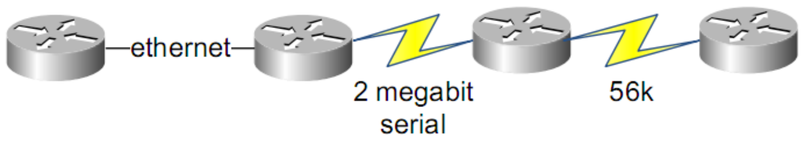

Figure 25: Cost Calculation Example

Multi-Area OSPF

Now that we have covered what happens inside an area, we can cover multi-area OSPF and describe how different areas interact. OSPF requires that there be at least one area present, and that area is Area 0. It also requires that all areas have an interface in Area 0. OSPF has a total of 4 area types that are implemented by Cisco.

Standard Area	An area with no special restrictions.
Stub Area	Type 5 LSAs, which advertise external routes, are not sent. This is because in a stub, no adjacent areas need to receive external routes. The external routes from Type 5 LSAs are distributed by a default route from all ABRs. Stub areas don't receive external routes.
Totally Stubby Area	The TSA is a Cisco proprietary type that not only disallows Type 5 LSAs, but also Types 3 and 4, which are summary LSAs. A TSA receives a default route.
Not So Stubby Area	A NSSA is a stub that has an ASBR. Internal routers receive the redistributed routes from another routing process through the ASBR.

Figure 26: OSPF Area Types

Stub areas are generally used when there is only 1 way in and out of an area. Think of a router on the edge of a network. A common practice is to use a default gateway to send all traffic to the next upstream router, since it is the only path available. Usually this is done with a static route. Since OSPF routers (with the exception of the ABR and ASBR) have no need to know about routers outside of their area, a stub area can be created to further optimize network traffic. When an area is a TSA, an ABR sends a Type 3 LSA advertising a route to 0.0.0.0 with mask 0.0.0.0 into the TSA. This is the only Type 3 LSA that the ABR will send. Normal stub areas still receive inter-area routes. The output from **show ip route** shows the E1, E2 and N1, N2 types in the legend.

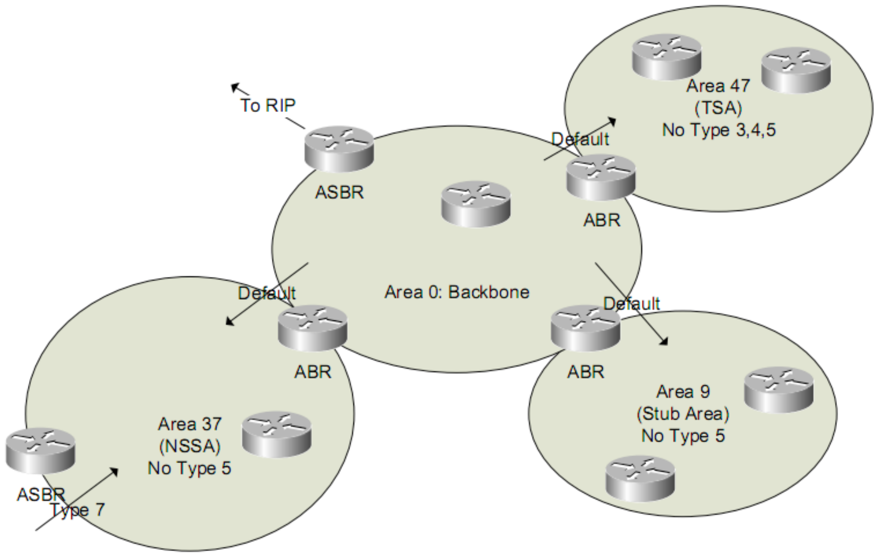

Figure 27: Multi-Area OSPF

Determining Network Resources Needed for Implementing OSPF

OSPF requires you to analyze what types of routers you have installed. Specifically, you must look at the interfaces that will be participating in the OSPF areas. Page $ gives the work-around and caveats for using OSPF on different network types. You need to be familiar, both for the exam and the real world, with the limitations of NBMA networks. You should know which networks assign DR/BDR routers and which networks do not. Also, you should know how to statically configure a neighbor.

Designing an OSPF Network

Engineers with the CCDA certification will notice that the hierarchy imposed by the use of areas follows naturally with Cisco's Access-Distribution-Core model. The access layer routers are often OSPF stub areas. The distribution layer is Cisco's recommended route summarization point. Since the ABR performs summarization, this fact should support your area and summary design. OSPF is also good for the Core layer, since it has a quick convergence time.

In addition to these general design rules, requirements for OSPF must be met:

- Every router must connect to Area 0.
- Route summarization can only be performed on an ABR or ASBR.
- Multicast interfaces must be used.
- Cisco routers do not support certain LSA message types, such as LSA Type 6. Check with your IOS documentation for more information.

Cisco gives some broad recommendations specific to OSPF design that may or may not be applicable to your network:

- No router should be in more than 3 areas.
- No area should have more than 50 routers.
- No router should have more than 60 neighbors.
- A router can be a DR or BDR for more than one net segment if necessary.
- Only 1 instance of OSPF should run on an ABR.
- In a hub-and-spoke topology, the DR/BDR should be at hubs.

Creating an OSPF Implementation Plan

An OSPF implementation plan should be similar to other implementation plans. Note that when implementing OSPF, you may have to restart the router or the routing process after applying OSPF. This may cause a network disruption. You must take this special consideration in to account when implementing your configuration.

Creating an OSPF Verification Plan

The SPF algorithm is CPU intensive. While validating your OSPF design, make sure the ABRs are not showing excessive CPU usage, since they are the ones running the SPF algorithm. This may be the sign of (1) a router in the area flapping or (2) an overloaded ABR.

Configuring OSPF Routing

We will begin configuring OSPF by implementing a single-area network with 4 routers (Figure 28). As the chapter progresses, we will divide the network in to areas.

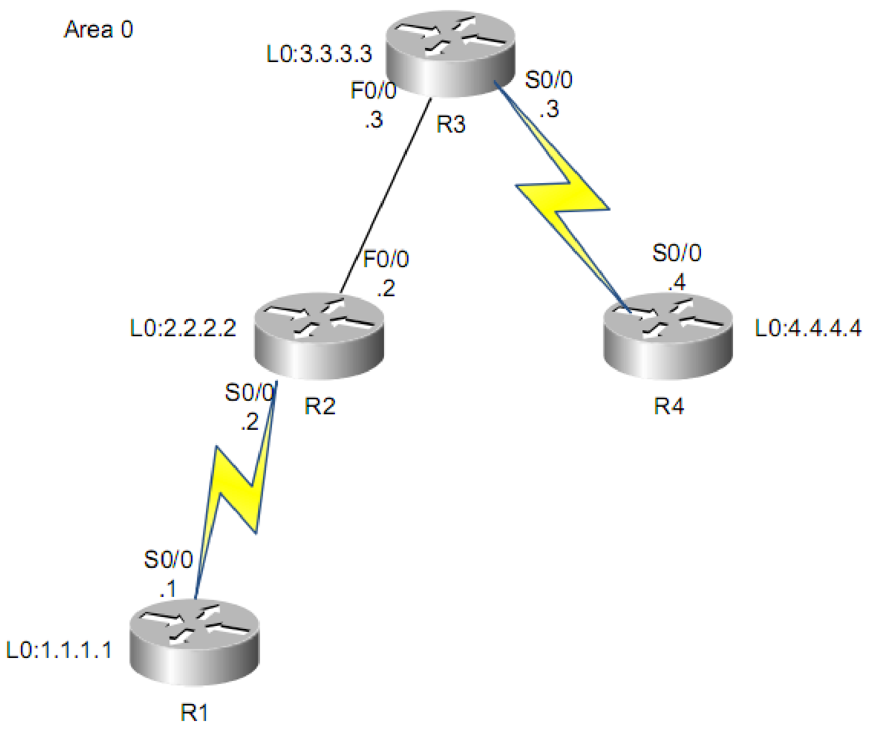

Figure 28: OSPF Topology. *All subnets are /24.*

Our serial links in this case are frame-relay point-to-point implementations using subinterfaces. For review, here is the config on R1's serial 0/0 interface:

```
R1(config)#int serial 0/0

R1(config-if)#encap frame-relay

R1(config-if)#!no frame-relay switch so no keepalive

R1(config-if)#no keepalive

R1(config-if)#no shut

R1(config-if)#end

R1(config)#int serial 0/0.102 point-to-point

R1(config-subif)#ip address 192.168.12.1 255.255.255.0

R1(config-subif)#!dlci must match on both sides

R1(config-subif)#frame-relay interface-dlci 100
```

The first step in configuring OSPF is to start an OSPF process on the router and tell OSPF which interfaces belong to which area. This can be done in one of 2 ways:

1. Use the **network *ip wcmask* area *n*** router subcommand. Where *n* is the area.
2. Use the **ip ospf *p* area *n*** interface subcommand. Where *p* is the local process-id and *n* is the area.

For example, to the following commands use method 1 to enable OSPF on the R1:

```
R1(config)#router ospf 1

R1(config-router)#network 192.168.12.0 0.0.0.255 area 0

R1(config-router)#network 1.1.1.1 0.0.0.0 area 0
```

Notice that if we use the interface's IP with wildcard mask 0.0.0.0, IOS is smart enough to enable it using the interfaces mask. For R2, we will use the interface subcommand method:

```
R2(config)#int serial 0/0.102

R2(config-subif)#ip ospf 1 area 0

R2(config-subif)#int fast 0/0

R2(config-if)#ip ospf 1 area 0

R2(config-if)#int loopback 0

R2(config-if)#ip ospf 1 area 0
```

The first method, using the **network *ip wcmask* area *n*** router subcommand, has the benefit of matching multiple networks at the same time and thus enabling OSPF for several interfaces in the same area with one command. The second method, using the **ip ospf *p* area *n*** subcommand, has the advantage that it doesn't require any subnet logic on the administrator's part. Be familiar with both methods for the exam.

As R1 and R2 come online, they exchange Hellos, LSUs, DBDs, and LSA messages. Understanding this output requires you to remember how adjacencies form and how routes are exchanged from the LSA section. Using **debug ip ospf events**, we can watch the process. The output below contains comments describing the process of starting OSPF on R1.

```
! OSPF becomes enabled and the first Hello is multicast


*Mar 1 00:23:19.951: OSPF: Interface Serial0/0.102 going Up


*Mar 1 00:23:19.951: OSPF: Send hello to 224.0.0.5 area 0 on Serial0/0.102
from 192.168.12.1

*Mar 1 00:23:19.967: OSPF: Interface Loopback0 going Up


! OSPF receives a hello message from R2

*Mar 1 00:23:20.047: OSPF: Rcv hello from 2.2.2.2 area 0 from Seri-
```

```
al0/0.102 192.168.12.2

! OSPF sees the Hello's areas matches and enters the two-way state.

*Mar 1 00:23:20.051: OSPF: 2 Way Communication to 2.2.2.2 on Seri-
al0/0.102, state 2WAY


! OSPF sends a Database Description packet to Router 2 (which it now ref-
erences by its RID)

*Mar 1 00:23:20.055: OSPF: Send DBD to 2.2.2.2 on Serial0/0.102 seq 0x226A
opt 0x52 flag 0x7 len 32

*Mar 1 00:23:20.059: OSPF: Send immediate hello to nbr 2.2.2.2, src ad-
dress 192.168.12.2, on Serial0/0.102

*Mar 1 00:23:20.059: OSPF: Send hello to 224.0.0.5 area 0 on Serial0/0.102
from 192.168.12.1


! OSPF Hellos are finished

*Mar 1 00:23:20.063: OSPF: End of hello processing


! Exstart begins and the neighbor Master-Slave relationship is determined

*Mar 1 00:23:20.155: OSPF: Rcv DBD from 2.2.2.2 on Serial0/0.102 seq
0x1104 opt 0x52 flag 0x7 len 32 mtu 1500 state EXSTART

*Mar 1 00:23:20.159: OSPF: NBR Negotiation Done. We are the SLAVE


! Since R1 is the slave, it $ from R2

*Mar 1 00:23:20.159: OSPF: Send DBD to 2.2.2.2 on Serial0/0.102 seq 0x1104
opt 0x52 flag 0x0 len 32

*Mar 1 00:23:20.251: OSPF: Rcv DBD from 2.2.2.2 on Serial0/0.102 seq
0x1105 opt 0x52 flag 0x3 len 52 mtu 1500 state EXCHANGE

*Mar 1 00:23:20.251: OSPF: Send DBD to 2.2.2.2 on Serial0/0.102 seq 0x1105
opt 0x52 flag 0x0 len 32

*Mar 1 00:23:20.327: OSPF: Rcv DBD from 2.2.2.2 on Serial0/0.102 seq
0x1106 opt 0x52 flag 0x1 len 32 mtu 1500 state EXCHANGE

*Mar 1 00:23:20.327: OSPF: Exchange Done with 2.2.2.2 on Serial0/0.102


! Sending the first LSA, which is a Link-State Request

*Mar 1 00:23:20.331: OSPF: Send LS REQ to 2.2.2.2 length 12 LSA count 1

*Mar 1 00:23:20.335: OSPF: Send DBD to 2.2.2.2 on Serial0/0.102 seq 0x1106
opt 0x52 flag 0x0 len 32


! Receiving a reply
```

```
*Mar 1 00:23:20.459: OSPF: Rcv LS UPD from 2.2.2.2 on Serial0/0.102 length
88 LSA count 1


! Enter the full state

*Mar 1 00:23:20.463: OSPF: Synchronized with 2.2.2.2 on Serial0/0.102,
state FULL


! Syslog a message that our neighbor is ready

*Mar 1 00:23:20.467: %OSPF-5-ADJCHG: Process 1, Nbr 2.2.2.2 on Seri-
al0/0.102 from LOADING to FULL, Loading Done
```

After enabling OSPF on R3 and R4, our single-area OSPF network is ready. We can confirm this by looking at the routing table:

```
R4#show ip route

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP

  D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

  N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

  E1 - OSPF external type 1, E2 - OSPF external type 2

  ia - IS-IS inter area, * - candidate default, U - per-user stat
ic route, o - ODR, P - periodic downloaded static route


Gateway of last resort is not set


O 192.168.12.0/24 [110/138] via 192.168.34.3, 00:00:06, Serial0/0.102

     1.0.0.0/32 is subnetted, 1 subnets

O 1.1.1.1 [110/139] via 192.168.34.3, 00:00:06, Serial0/0.102

     2.0.0.0/32 is subnetted, 1 subnets

O 2.2.2.2 [110/75] via 192.168.34.3, 00:00:06, Serial0/0.102

     3.0.0.0/32 is subnetted, 1 subnets

O 3.3.3.3 [110/65] via 192.168.34.3, 00:00:06, Serial0/0.102

     4.0.0.0/24 is subnetted, 1 subnets

C 4.4.4.0 is directly connected, Loopback0

O 192.168.23.0/24 [110/74] via 192.168.34.3, 00:00:06, Serial0/0.102

C    192.168.34.0/24 is directly connected, Serial0/0.102
```

Reducing Convergence Time

Before proceeding, we can tweak the OSPF Hello and dead timers to give us faster convergence. This is done on a per-interface basis and must be configured between pairs of routers. In this example, the **minimal** keyword sets the base interval to OSPF's lower limit of 1. This is built into IOS. The multiplier tells OSPF how many Hellos to send in that 1 second. In this case, we used 4, so we will send a hello every 250 ms.

```
R1(config-subif)#ip ospf dead-interval minimal hello-multiplier 4
```

Configure Authentication

Like other routing protocols, OSPF can authenticate neighboring routers before trusting their routes. Authentication can and should be configured between routers when the design calls for it. OSPF supports plain text authentication and MD5 authentication. First, we will configure plain text authentication between R1 and R2:

```
! Dead timer expires when authentication is configured

! For one side of a link but not the other

!

*Mar  1 17:15:26.575: %OSPF-5-ADJCHG: Process 1, Nbr 1.1.1.1 on
Serial0/0.102 from FULL to DOWN, Neighbor Down: Dead timer expired


R2(config-if)#int serial 0/0.102

R2(config-subif)#ip ospf authentication-key mypassw0rd!

R2(config-subif)#ip ospf authentication

R2(config-subif)#router ospf

R2(config-subif)#router ospf 1

R2(config-router)#area 0 authentication

*Mar  1 17:16:21.999: %OSPF-5-ADJCHG: Process 1, Nbr 1.1.1.1 on
Serial0/0.102 from LOADING to FULL, Loading Done
```

Likewise, we can repeat the process using the more secure MD5 hash. Since the authentication-key is sent in clear text, it has limited effectiveness. If intercepted in transit, an MD5 hash is a one-way function; it is difficult to reverse the encryption to get the source key. Reversing the MD5 hash requires a brute-force attack.

```
R3(config)#int serial 0/0.102

R3(config-subif)#ip ospf message-digest-key 1 md5 myk3y!

R3(config-subif)#ip ospf authentication message-digest

R3(config-subif)#router ospf 1

R3(config-router)#area 0 authentication message-digest
```

The adjacency will do the same thing as the adjacency between R1 and R2 in the first example.

Configuring Multi-Area OSPF

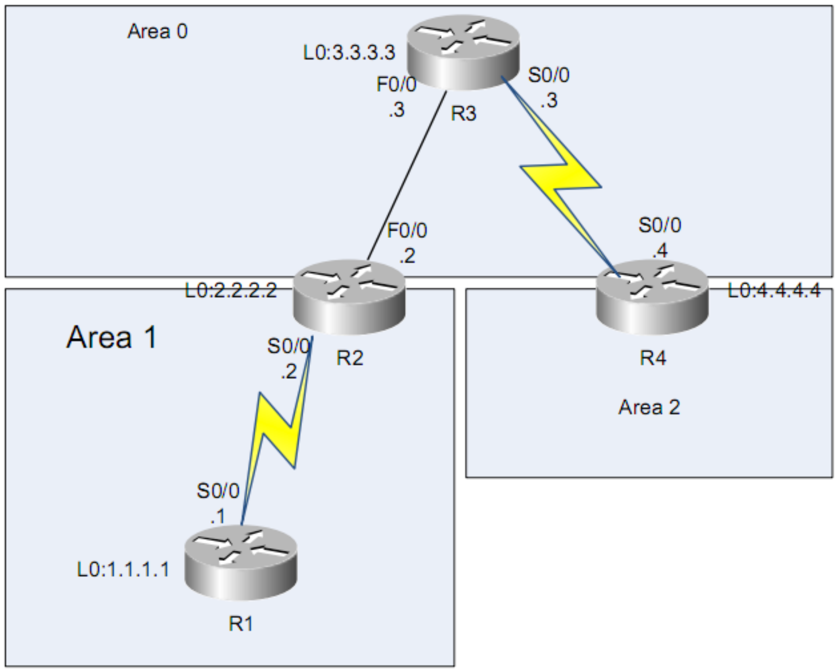

Figure 29: Multi-Area OSPF

In this example, we add two areas to our single-area design. This causes R1 and R4 to become ABRs and our routes to become Inter-Area routes. Behind the scenes, the neighbor topology has changed and so has the LSDB. Now, R1 only is exchanging messages with R2. R2, R3, and R4 are exchanging messages because they each have an interface in area 0. R4 is exchanging messages with no one because there are no other interfaces to talk to in Area 2.

```
R1#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address
Interface

2.2.2.2           0   FULL/  -        00:00:35    192.168.12.2
Serial0/0.102

R1# show ip route

C    192.168.12.0/24 is directly connected, Serial0/0.102

     1.0.0.0/24 is subnetted, 1 subnets

C       1.1.1.0 is directly connected, Loopback0

     2.0.0.0/32 is subnetted, 1 subnets

O IA    2.2.2.2 [110/65] via 192.168.12.2, 00:19:46, Serial0/0.102

     3.0.0.0/32 is subnetted, 1 subnets

O IA    3.3.3.3 [110/75] via 192.168.12.2, 00:10:12, Serial0/0.102

     4.0.0.0/32 is subnetted, 1 subnets

O IA    4.4.4.4 [110/139] via 192.168.12.2, 00:06:29, Serial0/0.102

O IA 192.168.23.0/24 [110/74] via 192.168.12.2, 00:19:46, Serial0/0.102

O IA 192.168.34.0/24 [110/138] via 192.168.12.2, 00:10:11, Serial0/0.102
```

Configure Summarization

OSPF routes should be summarized at the ABR and ASBRs. Technically these are the only places OSPF will allow you to summarize. The purpose of summarizing routes is to provide abstraction to the upstream router. By hiding the details of an internal area, processing is reduced for upstream routers and the router is protected from changes inside the area: summary routes will not change when an internal router changes. This is analogous to the idea of abstraction in a computer program. Computer programmers write functions that accept input and provide output, but the internal details are not divulged. Doing this allows programs to be simpler and makes them more modular. A change of code inside a function usually won't change the format of the values that are sent out.

OSPF can summarize routes in an area in two ways. The Inter-Area (ABR) route summarization, based on the Type 3 LSA, is configured using the **area range** command. The summary metric will be the lowest metric inside the range. On the ABR, an interface called Null0 is created and the routes for Null0 are sent out. The Null0 route is put in the routing table as a summary route. The purpose of this route isn't to send data to Null0, it is to provide a summarized route that can be advertised or redistributed to other routers.

The second summary is the ASBR summary, which summarizes the entire OSPF domain using a Type 5 LSA. This process is roughly the same as the Inter-Area summary except that it has a different scope. If the summary happens in a NSSA, the Type 5 LSA will be converted to a Type 7 LSA, since Type 5 LSAs are not allowed in stubs.

To illustrate this, we add 3 new networks to R1 using the following command:

```
int loopback 1

ip address 11.1.1.1 255.255.255.0

int loopback 2

ip address 11.2.1.1 255.255.255.0

int loopback 2

ip address 11.3.1.1 255.255.255.0
```

The routing table on R2 accepts these networks and enters them in to the routing table:

```
C    192.168.12.0/24 is directly connected, Serial0/0.102

     1.0.0.0/32 is subnetted, 1 subnets

O       1.1.1.1 [110/65] via 192.168.12.1, 00:07:36, Serial0/0.102

     2.0.0.0/24 is subnetted, 1 subnets

C       2.2.2.0 is directly connected, Loopback0

     3.0.0.0/32 is subnetted, 1 subnets

O       3.3.3.3 [110/11] via 192.168.23.3, 00:07:36, FastEthernet0/0

     4.0.0.0/32 is subnetted, 1 subnets

O IA    4.4.4.4 [110/75] via 192.168.23.3, 00:07:36, FastEthernet0/0

C    192.168.23.0/24 is directly connected, FastEthernet0/0

     11.0.0.0/32 is subnetted, 3 subnets

O       11.3.1.1 [110/65] via 192.168.12.1, 00:02:48, Serial0/0.102

O       11.2.1.1 [110/65] via 192.168.12.1, 00:00:46, Serial0/0.102

O       11.1.1.1 [110/65] via 192.168.12.1, 00:04:37, Serial0/0.102

O    192.168.34.0/24 [110/74] via 192.168.23.3, 00:07:36,
FastEthernet0/0
```

Router 3 also receives similar entries. Since R2 is the ABR for Area 1, we can summarize on R2 as follows:

```
R2#(config-router)# area 1 range 11.0.0.0 255.252.0.0

R2# show ip route

…

O       11.0.0.0/14 is a summary, 00:00:01, Null0
```

```
R3# show ip route

     11.0.0.0/14 is subnetted, 1 subnets

O IA    11.0.0.0 [110/75] via 192.168.23.2, 00:01:33, FastEthernet0/0
```

Note that R3 doesn't know that the 11.0.0.0 /14 network is a summary, but the summary route has been sent anyway.

Using Virtual-Links to Join Discontiguous Areas

One requirement of OSPF is that all areas must somehow touch the backbone. Assume that we wanted to connect a router to R1 and create a new area, Area 3. No physical way exists to connect R5 directly to an Area 0 router. OSPF overcomes this by using a logical interface called a virtual-link. A virtual link is a logical way of connecting areas to the backbone. They are needed to meet the requirement that all areas must connect to Area 0. The virtual link transits an area to reach Area 0. The virtual link is applied on the ABR and targets a router in Area 0. Figure 30 shows an updated diagram.

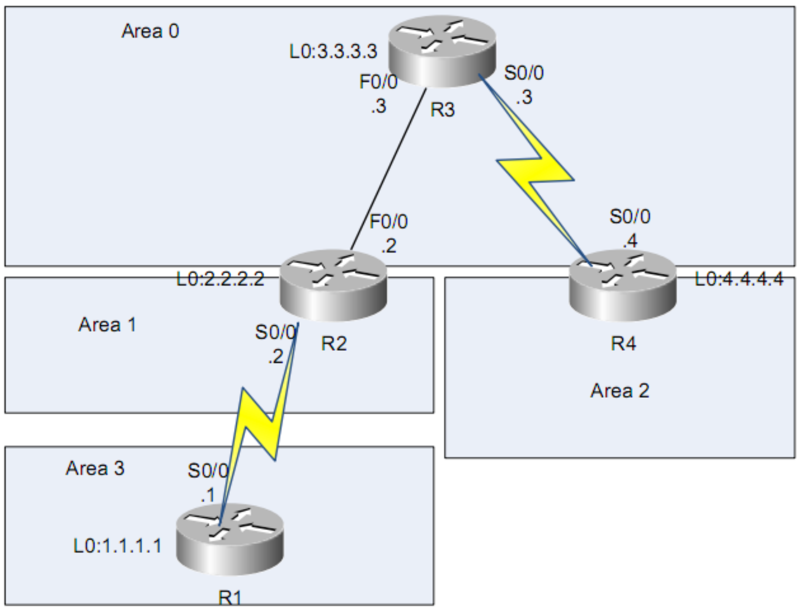

Figure 30: Area 3 has no backbone connection

To give Area 3 a virtual-link, we use these commands:

```
R3(config-router)#area 3 virtual-link 5.5.5.5

R5(config-router)#area 3 virtual-link 3.3.3.3
```

Now Area 3 will appear "bridged" to Area 0, as show in Figure 31.

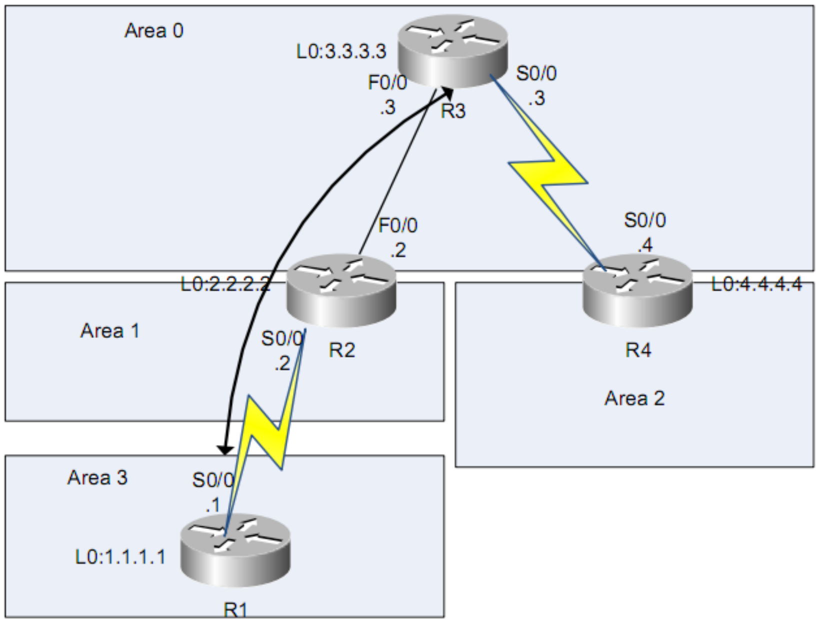

Figure 31: Virtual-Link

Verifying an OSPF Solution

```
R3#show ip protocols
Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 3.3.3.3
  Number of areas in this router is 2. 2 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    192.168.23.3 0.0.0.0 area 0
    0.0.0.0 255.255.255.255 area 0
 Reference bandwidth unit is 100 mbps
  Routing Information Sources:
    Gateway         Distance      Last Update
    2.2.2.2              110      00:47:22
    4.4.4.4              110      00:47:56
  Distance: (default is 110)
```

With the **show ip protocols** command, we can see that OSPF has been enabled and our RID has been established. This RID is correct, because R3 is using its highest-address loopback interface. If it was something other, it would be possible for the interface to have not existed when the OSPF process was started. In this case, we would have to restart the router for it to learn its new RID.

Show ip protocols also gives us a briefing on the number of areas, including stubs. It tells us which networks belong to which interfaces. The reference bandwidth is used to calculate the cost. If the interface subcommand **ip ospf cost c** is given, it will override the calculation.

```
R3#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
4.4.4.4           0   FULL/  -        00:00:35    192.168.34.4    Serial0/0.102
2.2.2.2           1   FULL/BDR        00:00:34    192.168.23.2    FastEthernet0/0
R3#
```

Show ip ospf neighbor can be useful when troubleshooting adjacencies and will tell us if there has been a DR/BDR assigned. In the case of R3, its only non-NMBA segment is the fast Ethernet to R2. Therefore, R3 has become the DR since it has a higher RID, and R2 has become the BDR.

You might see the following output from **show ip ospf neighbors**:

```
R9# show ip ospf neighbor

Neighbor ID     Pri       State           Dead Time   Address      Interface
10.11.3.4       1       2WAY/DROTHER    00:00:34    10.11.3.4     Ethernet0
10.11.3.3       1       2WAY/DROTHER    00:00:34    10.11.3.3     Ethernet0
10.11.3.8       1       FULL/DR         00:00:32    10.11.3.8     Ethernet0
10.11.3.2       1       FULL/BDR        00:00:39    10.11.3.2     Ethernet0
```

This output is deceiving for two reasons. First, recall that routers will get stuck in the 2WAY state if they are DR-Other routers and this command is run from a DR-Other. This is because the only adjacencies after a DR/BDR is elected are between the routers and the DR/BDR. Secondly, the Dead Time appears to be recent, indicating a problem. This is also deceiving because the Dead Time isn't the amount of time the neighbor has been down; it is the amount of time before the dead timer expires. Because this is an Ethernet interface which sends out Hello messages every 10 seconds by default, it will reset to 40 seconds every time a Hello is received. Therefore, the output generated by **show ip ospf neighbors** is correct.

The **debug** output listed on page $ gives the primary method of debugging OSPF issues, which is debugging the neighbor relationships and the message exchange process. Refer to the output on page $ for an explanation of what a neighbor relationship formation should look like while running **debug ip ospf events**.

To debug the OSPF database, use the command **show ip ospf database**. Sample output is given below:

```
R3#show ip ospf database


            OSPF Router with ID (3.3.3.3) (Process ID 1)

                Router Link States (Area 0)

Link ID         ADV Router      Age     Seq#       Checksum Link

2.2.2.2         2.2.2.2         47      0x80000005 0x0014E1 2

3.3.3.3         3.3.3.3         91      0x80000005 0x00D24E 4

4.4.4.4         4.4.4.4         54      0x80000005 0x00B14E 2


                Net Link States (Area 0)


Link ID         ADV Router      Age     Seq#       Checksum

192.168.23.3    3.3.3.3         91      0x80000003 0x003E56



                Summary Net Link States (Area 0)

Link ID         ADV Router      Age     Seq#       Checksum

1.1.1.1         2.2.2.2         47      0x80000003 0x00A746

4.4.4.4         4.4.4.4         54      0x80000003 0x005EBB

192.168.12.0    2.2.2.2         47      0x80000003 0x0095E7
```

Here is the output of the **show ip ospf interface** command. This command is also useful in troubleshooting. In the example below, we have 3 interfaces participating in OSPF (Loopback 0, FastEthernet 0/0, and Serial 0/0).

The costs of each network are displayed. Remember that cost is a function of bandwidth, so that it shows OSPF come up with these numbers. It also gives us information such as the Hello and Dead timers, the OSPF process and areas assigned, and if the link is a DR or BDR.

```
R2#show ip ospf interface
Loopback0 is up, line protocol is up
  Internet Address 2.2.2.2/24, Area 0
  Process ID 1, Router ID 2.2.2.2, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
FastEthernet0/0 is up, line protocol is up
  Internet Address 192.168.23.2/24, Area 0
  Process ID 1, Router ID 2.2.2.2, Network Type BROADCAST, Cost: 10
  Enabled by interface config, including secondary ip addresses
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 3.3.3.3, Interface address 192.168.23.3
  Backup Designated router (ID) 2.2.2.2, Interface address 192.168.23.2
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:00
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 1/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 3.3.3.3  (Designated Router)
  Suppress hello for 0 neighbor(s)
Serial0/0 is up, line protocol is up
  Internet Address 192.168.12.2/24, Area 1
  Process ID 1, Router ID 2.2.2.2, Network Type POINT_TO_POINT, Cost: 64
  Enabled by interface config, including secondary ip addresses
  Transmit Delay is 1 sec, State POINT_TO_POINT
```

```
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  oob-resync timeout 40
  Hello due in 00:00:06
Supports Link-local Signaling (LLS)
Cisco NSF helper support enabled
IETF NSF helper support enabled
Index 1/1, flood queue length 0
Next 0x0(0)/0x0(0)
Last flood scan length is 0, maximum is 0
Last flood scan time is 0 msec, maximum is 0 msec
Neighbor Count is 0, Adjacent neighbor count is 0
Suppress hello for 0 neighbor(s)
```

Documenting the Results of an OSPF Implementation

As stated in chapter one, a routing protocol documentation plan should be used to create concise and accurate documentation of the network, including data on any unforeseen issues. See chapter 1 for information on this topic.

Domain 3: Border Gateway Protocol

Introduction

As a CCNA, you might not have any experience with BGP. The ROUTE exam (642-902) introduces you to BGP from the perspective of an enterprise looking to hook their systems to an ISP. The BGP exam (642-661) is part of the CCIP curriculum and teaches BGP from the perspective of an ISP routing data to and from other ISPs.

BGP is a Layer 7 protocol that communicates on TCP port 179, making the communications reliable. It is called a "path vector" protocol because it uses paths, which are defined by a series of path attributes, to route traffic. Path Vector is similar to Distance Vector, but it uses Autonomous Systems as hops instead of routers. An Autonomous System (AS) is an entity under the control of an organization. In the context of BGP, an AS may be an ISP or enterprise.

BGP differs from internal routing protocols because it is used between Autonomous Systems and was designed to be used with the Internet in mind. In fact, the latest version of BGP (BGP v4) was ratified in 1994 to help the Internet move away from NSFNet, which was a research network whose operation was inhibiting Internet growth and eventually became part of the Internet backbone. BGP is unique in that it is the only routing protocol currently allowed on the Internet; hence it is the only EGP in operation today. BGP is very scalable and can be used in the largest of networks: there are currently some 320,000 BGP prefixes being advertised from 50,000 ASes worldwide. Each AS is usually composed of many BGP routers, which are hidden from eBGP's view of the network.

There are two subprotocols in BGP: iBGP and eBGP. iBGP (Internal BGP) runs inside an autonomous system, and eBGP (External BGP) routes traffic between autonomous systems. You do not have to specify which way BGP runs: if you configure two routers as neighbors with the same AS number, they will be iBGP neighbors. If you configure neighbors with different AS numbers, they will be eBGP neighbors.

The main component of BGP is the AS. With BGP being an inter-AS protocol, it is mostly concerned with limiting the number of ASes it must traverse to send data from its source AS to the destination AS. Like OSPF and EIGRP, BGP is composed of 3 components: the neighbor relationship, the path database, and the routing table.

eBGP is not concerned with the internal working of an AS, as those details are abstracted by the administrators of those ASs. Figure 32 shows an example of how a group of BGP neighbors (also known as peers) may look. This configuration might be how a group of ISPs, each with its own AS domain, may exchange data.

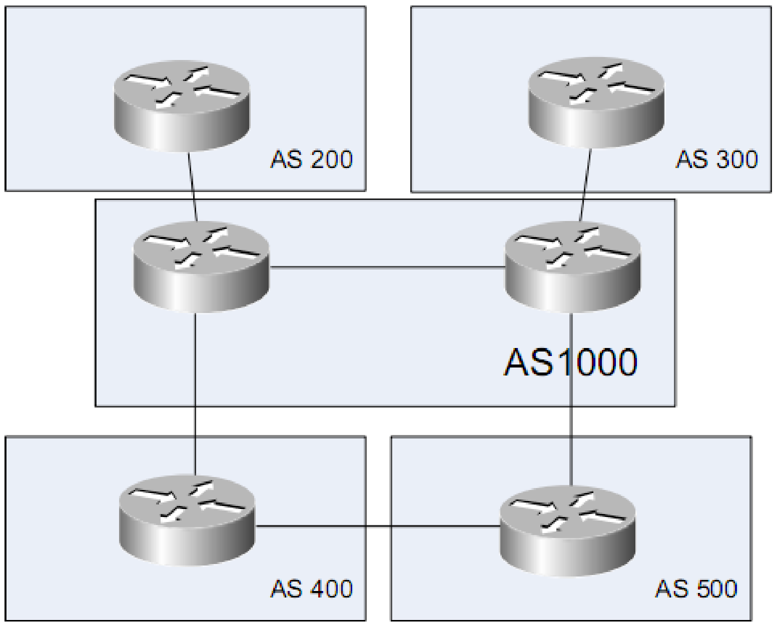

Figure 32: BGP between ASs.

Like IP addresses, there are private and public AS numbers. AS 1-64511 are public numbers and are assigned by the Internet Assigned Numbers Authority (IANA), the same entity that allocates IP addresses. AS 64512-65534 are private AS numbers and are not routable on the public Internet.

BGP convergence is slow, and its updates are triggered and incremental. This is because of the nature of the Internet, where routers come up and down somewhere all the time. If BGP converged quickly, routers would be working on updates non-stop. BGP was designed for stability and not convergence speed.

There are 3 designs for eBGP in an enterprise: single-homed, multi-homed, or transit. A *single-homed network* connects only to one other AS. In this case, a static default route is probably all that is needed for connection between ASs. A single-homed network is a stub. Multiple links to the same AS are considered single-homed.

A *multi-homed network* is one where an AS connects to two or more other ASs and all connections are active at the same time. This configuration provides redundancy and possibly better speed for certain routes, it may be more efficient to send certain kinds of traffic through one AS as opposed to the other. This configuration is common when an enterprise needs to link servers in its DMZ to several different ISPs, but use the same set of addresses on its servers. It will link to 2 or more upstream ISPs, each their own AS, and advertise its routes to the rest of the network. This is illustrated in Figure 33.

A *transit* AS is one where traffic crosses the AS to reach another AS. This is the common configuration for ISPs. ISPs peer with other ISPs and trade routing information using BGP. Traffic is then routed around the Internet as it *transits* ISPs. Consider Figure 33. If we wanted to, AS 100 could peer with AS 200 and exchange routes through our routers. If we were an ISP, this would be a good configuration. However, since we are an enterprise, we do not want traffic crossing our network if it isn't destined for our servers. Therefore, we can use route filtering to prevent these routes from entering our AS.

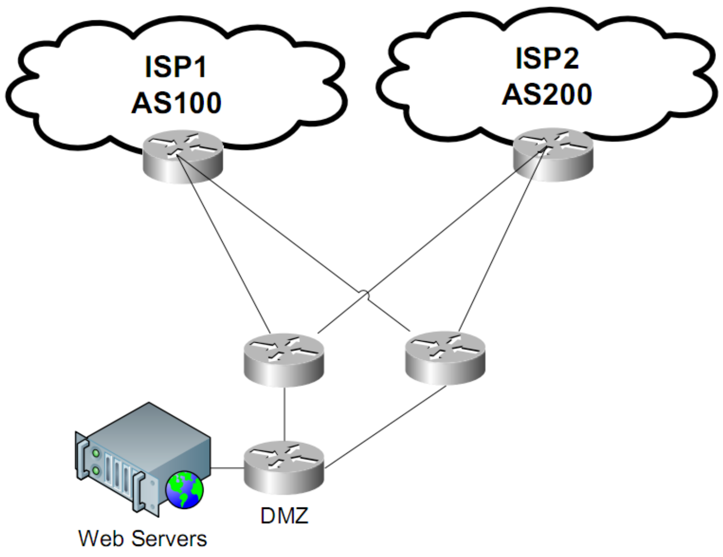

Figure 33: A Multi-Homed Web Server

How would one go about making sure that traffic is balanced across both ISPs? We need to make sure we have a balance to our servers. We could use DNS round robin, but DNS round-robin has no knowledge of the network structure or the underlying link costs. Plus, if a link goes down, DNS RR will not detect this. A better solution for true redundancy is to use BGP.

In this design, we could potentially download the entire Internet routing table to both of our routers from the perspective of both AS 100 and AS 200. We probably don't want to do this just to advertise routes for our web servers. We should filter out all of the routes we don't need.

Transit AS and Synchronization

iBGP is used to route BGP traffic (potentially from other ASs) through our AS. In this configuration, our AS is known as a *transit AS*. Figure 32 shows the topology of a transit AS. The BGP process in AS 300 has peered with the routers in AS 1000 and AS 200. AS 1000 has advertised a path to AS 200 that AS 300 has chosen to use.

To illustrate this, assume we were an ISP (AS 1000) who had connections to AS 200 and AS 300, who are other ISPs. When AS 200 wants to send data to AS 300, it will cross our network. Rarely would an ISP have all of the resources needed to join the three ASes in a single device. In fact, our design should use the Enterprise Campus and Access-Distribution-Core model from Cisco. In this case, traffic would have to cross multiple iBGP routers in our AS. Our internal routers don't necessarily have to run BGP, but creating a full-mesh of BGP routers is recommended because of the inability of other IGPs to handle the number of BGP routes.

To this end, a special rule applies exclusively to iBGP. The rule states that iBGP will not advertise traffic for a network unless the network is also routed by an underlying IGP. This is known as the *synchronization rule* and is enabled by default in IOS earlier than 12.2(8). Synchronization must be enabled when a router in the path between two BGP neighbors isn't running BGP. Figure 34 shows how the IGP, iBGP, and eBGP logically interact.

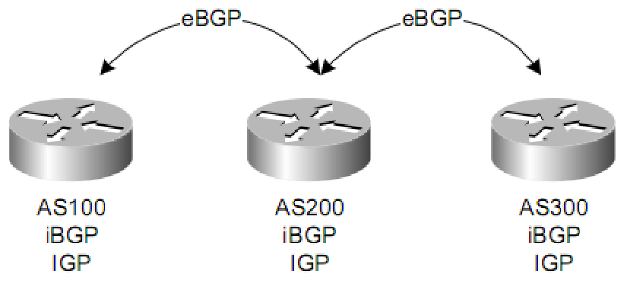

Figure 34: iBGP Interactions. *Note that iBGP rides on top of an IGP and helps eBGP.*

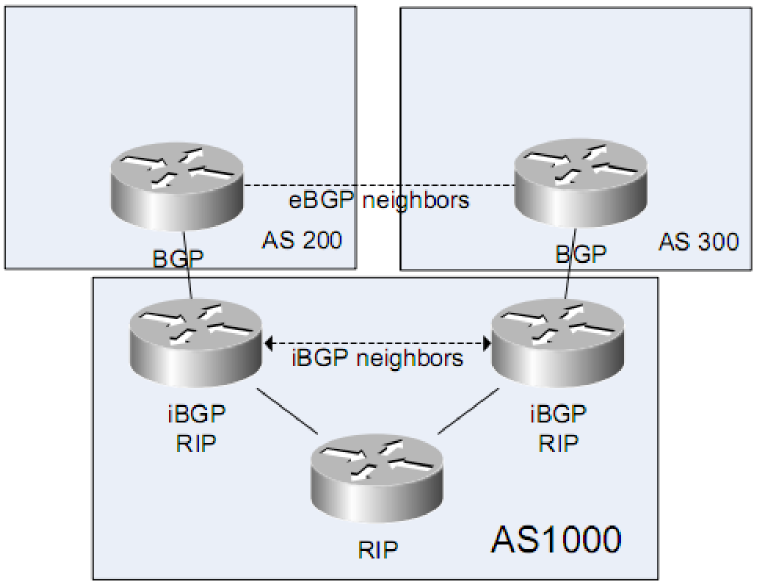

Figure 35: iBGP Creating Neighbor Relationships.
iBGP can create neighbors between non-connected routers. However, transit conditions must be accounted for both internally and externally.

Two facts emerge from the configuration in Figure 35. First, since BGP must traverse a router that is not running BGP in AS 1000, the routers must use the synchronization rule. Secondly, consider traffic moving through this network from AS 200 to AS 300. When the traffic hits the RIP router, the router may not know where the traffic should go, as it isn't processing the routes it receives from BGP. This is why iBGP is necessary on each router that may receive a BGP route.

Although not always possible, it is a good idea to run BGP on all of the routers in the path. Doing this also prevents the RIP router from not knowing routes that BGP knows, and prevents you from having to inject a large number of BGP routes into RIP, which could possibly overload the routers running RIP and crash the network. You should configure your iBGP routers in a fully-meshed topology so routes will not be lost (which is known as a *blackhole*) as they transit your AS.

Loop Avoidance

eBGP avoids loops by checking the AS_PATH variable. The AS_PATH variable stores a list of all the autonomous systems that the route has taken. If eBGP sees its own AS in the AS_PATH, it knows that it has seen the route before.

iBGP doesn't advertise any routes it learns from its peers to other iBGP peers. It does this as a loop-avoidance mechanism. The ASN in eBGP gives a mechanism to avoid eBGP loops, but all ASNs are the same with iBGP. Therefore, an iBGP network should be configured in a full mesh.

BGP Neighbors

Like other routing protocols, BGP uses neighbor relationships to exchange data. Unique to BGP is the requirement that neighbors must be manually defined: BGP does not automatically discover connected neighbors. Also unique to BGP is that it is a layer 7 protocol, which means it operates on top of a lower-level routing protocol (or a static route).These two properties may be viewed as setbacks to BGP, but they give BGP the ability to form neighbor relationships between routers that are not directly connected to each other.

BGP has 4 message types that it sends to neighbors. Through the message exchange process, neighbors are formed and kept alive via Keepalive messages. When a route is added or removed from the BGP routing process, BGP sends the change incrementally to its neighbors via an Update message.

Message	Description
Open	The open message starts a session and is similar to a Hello packet.
Keepalive	These are sent every 60 seconds; the keepalive keeps peers open. The holddown timer resets after receiving a hello.
Update	Updates are sent when a route changes.
Notification	Notifications carry error messages and close sessions.

Figure 36: BGP Messages

BGP maintains a record of its neighbor states, which can be verified with the **show ip bgp summary** command. The states are given in Figure 37.

State	Description
Idle	Not connected: The router is looking for the neighbor.
Connect	The neighbor has completed the TCP handshake.
OpenSent (Active)	Open has been sent; we are waiting for a reply.
OpenConfirm	We have received a reply to the Open message.
Established	Full BGP state has been established.

Figure 37: BGP Neighbor States

BPG Path Selection

BGP's metric is the most complex of all the routing protocols, but it also allows for the most customization. The variables in BGP are known as *attributes*. The items in Figure 38 describe how BGP picks routes. It is given in the order of priority. With item #1 being the most significant and item #13 being the least significant. When BGP is choosing between two routes, it compares the values in this table. Only one condition needs to be different for a path to be chosen and further processing terminated. In other words, BGP will go down the list checking these attributes and comparing them to each path's values. As soon as it finds an attribute that determines a better path, it stops comparison and inserts the route into the Router Information Base (RIB). BGP stores its paths in a variable list called AS_PATH. If BGP receives a path that contains its own AS number in it, it does not process the path, as it knows the message came from a loop.

The BGP Update message informs neighbors about the state of a path. Prefix/length information in BGP is called Network Layer Reachability Information (NLRI). NLRI is a variable-length field that an update message uses to advertise the end-to-end path and a number of path attributes about that path. This path attribute information can be used by BGP to make routing decisions.

Name	**Description**
Weight	Take the path with the highest *weight*. This is a Cisco proprietary.
LOCAL_PREF	Take the path with the highest **local-preference.**
Origin	Take the path from a local **network** or **aggregate** BGP subcommand, or a route redistributed from an IGP.
AS_PATH	The shortest AS_PATH will be preferred. This can be ignored with the **bgp bestpath as-path ignore** command.
Origin Type	IGP's have a lower origin type than EGP.
MED	Take the path with the lowest Multi-Exit Discriminator.
Confederation	Take an eBGP path over an iBGP path.
IGP	Take the path with the lowest IGP metric to the BGP next hop.
Multipath	Determine if multipath is applicable.
Oldest Path	Take the path that was received first.
Router-ID	Take the route with the lower Router-ID.
Cluster	If BGP Round-Robin is used, the Cluster-ID replaces the Router-ID.
Address	Take the path with the lowest neighbor address.

Figure 38: BGP Path Attributes

There are 4 kinds of path attributes:

- **Well-known Mandatory Attributes** – propagated to peers in update messages. They include AS_PATH, AS_NEXT-HOP, and Origin.
- **Well-known Discretionary Attributes** – include LOCAL_PREF and ATOMIC_AGGREGATE. They optionally appear in update messages.
- **Optional Transitive Attributes** – include AGGREGATOR and COMMUNITY. These attributes are transitive attributes that should be passed on.
- **Optional Non-transitive Attributes** – include the MED. Since it is non-transitive, a router does not necessarily have to pass it on to downstream neighbors.

Some of the attributes are relevant to the ROUTE exam and are described in more detail here.

The first attribute, *Weight*, is a Cisco-specific attribute that is local to the router. By "local to the router", we mean that this attribute is not passed on to BGP neighbors. Weight is often used as an easy method to configure which routes will go where.

LOCAL_PREF tells the router which route is preferred. It is similar to the *Weight* attribute, but is not Cisco proprietary. It is propogated through the AS.

Origin tells the router which routes have been locally originated. The origin can be eBGP, iBGP, or Incomplete. A path originated inside the local AS is iBGP, a path outside is eBGP. Incomplete paths usually occur when redistribution is used.

AS_PATH is the multi-length field that contains the NLRI (Network Layer Reachability Information). In short, this is a list of other ASes the route traverses.

Multi-Exit Discriminator (MED) is an optional attribute that is sent out from a router that tells upstream routers that one path is preferred over another. It is used to tell neighbors about the preference, rather than setting the preference for this router. It tells neighboring ASes that our AS has an optimal entry point. It is used for inbound traffic engineering, telling upstream routers which path is preferred.

Implementing an eBGP Based Solution

BGP is useful for multihoming connections to different ISPs. There are three scenarios used with multi-homing:

1. Each ISP passes on the default route to our AS.
2. Each ISP passes only a default route and provider-owned routes to the AS. A limited number of routes can be injected into our IGP from the routes our provider sends us.
3. Each ISP passes all routes to the AS. This requires all routers to be running BGP because of the size of the routing table.

When Not to Use BGP

BGP is not for use everywhere. BGP is overkill for all but the largest of enterprises. Its complexity and hardware overhead make it an unlikely candidate for smaller businesses.

Because it doesn't automatically find neighbors, it might be too much configuration for smaller businesses. Most enterprises require faster convergence than BGP offers. BGP is a slow protocol. It can take up to a minute for a route to appear.

Most enterprise routers have no need to accept the entire Internet routing table. Because of its size, many routers do not even have the RAM to hold it. BGP is necessary when you need to precisely control routes, or when your network edge is multihomed and AS path information is required.

The synchronization rule limits BGP's effectiveness in an enterprise. Since BGP is a layer-7 protocol, IP connectivity between routers must already be established. Recall that the synchronization rule means routes must exist in an IGP before BGP will consider them. Therefore, if an IGP is already running, there may be little incentive to running BGP. Synchronization can be disabled with the BGP subcommand **no synchronization**.

Determining eBGP Network Resource Requirements

BGP requires careful planning to answer the following questions:

- What routes are we going to send to our neighbors?
- What routes are we going to receive?
- Do our routers require a memory upgrade to hold the BGP table?
- What routes do we need to filter so unwanted BGP traffic doesn't transit our AS?
- Do we have a full mesh for our iBGP routers?
- Have we been assigned a public AS number (if applicable)?
- Do we need to redistribute any routes in to BGP?

Creating an eBGP Implementation Plan

An eBGP implementation plan should include a BGP control policy. The policy should define how filters will be applied, what routes to advertise and to whom, and how the routes will be tweaked to optimize performance.

Creating an eBGP Verification Plan

To verify BGP, we need to verify what routes we have received, as well as what routes our neighbors have received. We also need to make sure that our iBGP mesh is working correctly and that there are no black holes for traffic to fall into. Verifying BGP means meeting our design requirements. This includes correct filters, traffic manipulation, policies, performance, and security.

Configuring eBGP Routing

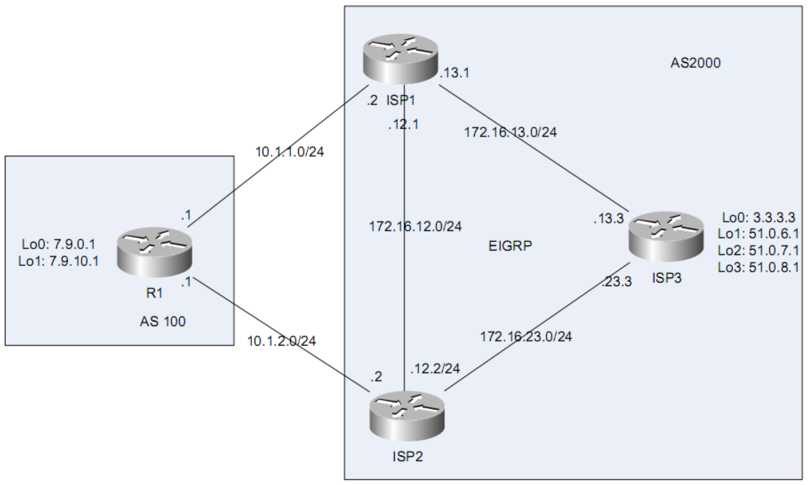

Figure 39: eBGP Routing Scenario

In this scenario, we have an enterprise router that we want connected to one ISP via two links. The first step is to start the BGP process for our AS. After that, we define our neighbors and the networks we want to advertise. A router can only run one instance of BGP at a time. The BGP instance is specified by the AS number. Unlike IGPs, BGP does not automatically detect neighbors: they must be manually entered. This is done using the **neighbor** command. A neighbor cannot form unless it is entered on both neighbors. After our neighbors have been entered, we specify networks we want to advertise with the **network** command.

```
R1(config-if)#router bgp 100

R1(config-router)#neighbor 10.1.1.2 remote-as 2000

R1(config-router)#neighbor 10.1.2.2 remote-as 2000

R1(config-router)#network 7.9.0.0 mask 255.255.255.0

R1(config-router)#network 7.9.10.0 mask 255.255.255.0

R1(config-router)#network 10.1.1.0 mask 255.255.255.0

R1(config-router)#network 10.1.2.0 mask 255.255.255.0
```

BGP will not advertise a network unless it is entered with a matching mask. If you are unsure of the network and mask, you can use **show ip route** and enter the network in question.

The **neighbor** command is the main command for configuring how BGP interacts with its neighbors. To illustrate this, we can use the **neighbor** command to set up BGP authentication. BGP Authentication is relatively easy to configure.

```
R1(config-router)#neighbor 10.1.1.2 password pa$$word

R1(config-router)#neighbor 10.1.2.2 password pa$$word
```

On R1, we have specified 2 neighbors (10.1.1.2 and 10.1.2.2), but we haven't set them up yet. On R1, this becomes evident when our neighbor relationships don't form quickly. When BGP connects to a new neighbor, the following message is logged:

```
*Mar 1 00:33:49.411: %BGP-5-ADJCHANGE: neighbor 10.1.1.2 Up
```

If this message is not seen as expected, use the **show ip bgp summary** command:

```
R1# show ip bgp summary

BGP router identifier 10.10.2.1, local AS number 100

BGP table version is 1, main routing table version 1

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd

10.1.1.2        4  2000       0       0        0    0    0 never    Active

10.1.2.2        4  2000       0       0        0    0    0 never    Active
```

This table tells us that our neighbor relationships haven't formed. The Up/Down timer says they have never been up or down. The **Active** state means that BGP is trying to form a relationship: it is sending Hello messages unicast to 10.1.1.2 and 10.1.2.2, but not getting a response. Now we should configure ISP1 and ISP2.

```
ISP1(config)#router bgp 2000

ISP1(config-router)#neighbor 10.1.1.1 remote-as 100



ISP2(config)#router bgp 2000

ISP2(config-router)#neighbor 10.1.2.1 remote-as 100
```

Our neighbor relationships should form. We can then use the following command to see what paths we are advertising:

```
R1#show ip bgp

BGP table version is 8, local router ID is 7.9.10.1

Status codes: s suppressed, d damped, h history, * valid, > best, i
- internal, r RIB-failure, S Stale

Origin codes: i - IGP, e - EGP, ? - incomplete


   Network          Next Hop          Metric LocPrf   Weight Path

*> 7.9.0.0/24       0.0.0.0                0            32768 i

*> 7.9.10.0/24      0.0.0.0                0            32768 i

*> 10.1.1.0/24      10.1.1.2                            0 2000 i

*> 10.1.2.0/24      10.1.2.2                            0 2000 i
```

The first column, ***>**, is not just for looks. The legend indicates that ***** means a path is valid, while **>** indicates the path is the best path learned. On the exam, you want both of these criteria to be present. The **Network** and **Next Hop** columns are self-explanatory. As stated, **Weight** is a Cisco-proprietary local attribute that can be tweaked to influence how a path will be taken. **Path** indicates the AS of the next hop, with the **i** designation for an AS that we are either in or connected to. The **i** here indicates what AS the prefix originated in.

The **show ip bgp summary** command is also useful to see our neighbors.

```
R1#show ip bgp summary

BGP router identifier 7.9.10.1, local AS number 100

BGP table version is 8, main routing table version 8

7 network entries using 840 bytes of memory

7 path entries using 364 bytes of memory

3/2 BGP path/bestpath attribute entries using 372 bytes of memory

1 BGP AS-PATH entries using 24 bytes of memory

0 BGP route-map cache entries using 0 bytes of memory

0 BGP filter-list cache entries using 0 bytes of memory

Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory

BGP using 1632 total bytes of memory

BGP activity 7/0 prefixes, 7/0 paths, scan interval 60 secs


Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd

10.1.1.2        4  2000      52      57        8    0    0 00:36:32        0

10.1.2.2        4  2000      49      51        8    0    0 00:41:30        3
```

We see our local RID and AS number, and the version of the BGP table. Each time an entry is added or removed from the table, the table version should increment. In addition to summarizing the information we need to know about our neighbors, **show ip bgp summary** also shows us statistics on the local system. We can see memory usage and the number of messages waiting in the queue to each neighbor.

At this point, we have established basic connectivity from our enterprise to ISP1 and ISP2. We will now configure the ISP side.

The ISP is composed of 3 routers running in AS 2000. EIGRP is running between them. The reason for EIGRP is two-fold. First, since this is an iBGP configuration, we are subject to the synchronization rule, which states that a route must be present in an IGP before BGP will use it. Secondly, since BGP is a layer 7 protocol, we must have IP connectivity between routers. This means that we would either have to statically define the networks to achieve reachability, or we can instead put the routes in an IGP.

In addition to the configuration we already applied to ISP1 and ISP2, we have the following:

```
ISP2(config)#router bgp 2000

ISP2(config-router)#network 1.1.1.1 mask 255.255.255.255

ISP2(config-router)#network 10.1.1.0 mask 255.255.255.0

ISP2(config-router)#network 172.16.0.0

ISP2(config-router)#neighbor 172.16.13.3 remote-as 2000

ISP2(config-router)#no auto-summary
```

Router ISP3 we will configure as a Route Reflector (RR). A *Route Reflector* collects BGP routes and then distributes them to other routers known as *clients*. This is to get around the full-mesh requirement of iBGP. Instead of each router peering with each other router, all routers can peer with a central RR to learn routes. All routes from the clients are advertised in to the RR.

The following configuration allows us to avoid setting ISP1 and ISP2 as neighbors with each other. Instead, they will learn routes through ISP3:

```
ISP3(config)#router bgp 2000

ISP3(config-router)#network 3.3.3.3 mask 255.255.255.255

ISP3(config-router)#neighbor 172.16.13.1 remote-as 2000

ISP3(config-router)#neighbor 172.16.13.1 route-reflector-client

ISP3(config-router)#neighbor 172.16.23.2 remote-as 2000

ISP3(config-router)#neighbor 172.16.23.2 route-reflector-client

ISP3(config-router)#network 51.0.6.0 mask 255.255.255.0

ISP3(config-router)#network 51.0.7.0 mask 255.255.255.0

ISP3(config-router)#network 51.0.8.0 mask 255.255.255.0

ISP3(config-router)#no auto-summary
```

The networks 51.0.6.0/24 through 51.0.8.0/24 are used to represent Internet routes. Our routing table on each router should look similar to the following:

```
ISP3#show ip route bgp

     1.0.0.0/32 is subnetted, 1 subnets

B       1.1.1.1 [200/0] via 172.16.13.1, 00:44:07

     2.0.0.0/32 is subnetted, 1 subnets

B       2.2.2.2 [200/0] via 172.16.23.2, 1d03h

     7.0.0.0/24 is subnetted, 2 subnets

B       7.9.10.0 [200/0] via 172.16.13.1, 00:41:41

B       7.9.0.0 [200/0] via 172.16.13.1, 00:41:41

     10.0.0.0/24 is subnetted, 1 subnets

B       10.1.1.0 [200/0] via 172.16.13.1, 00:44:07
```

Because we are not filtering anything, each router should have the same routes, including our enterprise router in AS100.

Now that we have basic communication between AS100 and AS2000, we can begin optimizing BGP. For example, many BGP peers use the loopback address as the address that they peer to. The reason for this is that if the IP addresses change, the neighbor statements won't have to be re-written. It is worth noting that eBGP peers always almost peer directly with neighboring IPs and not with loopbacks. Loopbacks are generally used for iBGP scenarios.

To configure this, we specify a far end router (by its loopback address) and specify what IP address we want our updates to appear they come from. If 111.111.111.111 is the far end address of R1, ISP1 is configured like this:

```
ISP1(config-router)#neighbor 111.111.111.111 update-source loopback 0

ISP1(config-router)#neighbor 111.111.111.111 ebgp-multihop 2
```

The first statement specifies that we want to send 111.111.111.111 the IP address of 1.1.1.1 (our local loopback) as the update source. The second statement tells BGP to look for up to 2 hops to find our loopback address. This is because the conversion from a physical to logical interface counts as a hop. We configure R1 as follows:

```
R1(config)#interface Loopback3

R1(config-int)#ip address 111.111.111.111 255.255.255.255

R1(config-int)#router bgp 100

R1(config-int)#neighbor 1.1.1.1 update-source loopback 3

R1(config-int)#neighbor 1.1.1.1 ebgp-multihop 2
```

Technically, at this point, this command should fail. The reason for this is because we do not have a route to 111.111.111.111 from ISP1. We can either specify a static route, or inject our loopback interfaces in to BGP with the **network** subcommand. If we inject our loopback routes into BGP, we need to also make sure that our IGP has the routes, or that we have a static route configured for the prefix.

Next-Hop-Self

At the edge where iBGP becomes eBGP, and vice versa, we sometimes have to set the **next-hop-self** attribute when defining a neighbor. Normally, the next-hop attribute is the next IP address to reach a destination and is chosen automatically, but when The **next-hop-self** attribute disables automatic selection and specifies the IP of the router we are on. In networks where the neighbors do not have direct access to each other on the same subnet, BGP's automatic next-hop selection mechanism can result in broken routing.

Manipulating Path Attributes

We can tweak BGP to prefer one path over another. This is done by manipulating paths with the **neighbor** command. Many tweaks can be applied with a **route-map.**

```
R1(config)#route-map toISP1 10

R1(config-route-map)#set ?

  as-path           Prepend string for a BGP AS-path attribute

  automatic-tag     Automatically compute TAG value

  clns              OSI summary address

  comm-list         set BGP community list (for deletion)

  community         BGP community attribute

  dampening         Set BGP route flap dampening parameters

  default           Set default information

  extcommunity      BGP extended community attribute

  interface         Output interface

  ip                IP specific information

  ipv6              IPv6 specific information

  level             Where to import route

  local-preference  BGP local preference path attribute

  metric            Metric value for destination routing protocol

  metric-type       Type of metric for destination routing protocol

  mpls-label        Set MPLS label for prefix

  nlri              BGP NLRI type

  origin            BGP origin code
```

```
tag              Tag value for destination routing protocol
traffic-index    BGP traffic classification number for accounting
vrf              Define VRF name
weight           BGP weight for routing table
```

Recall that when choosing a path, BGP will make comparisons of the 13 attributes and as soon as it has a match, it will take the path. In this example, we configure a **route-map** and set R1 to prefer neighbor ISP1 with the following command:

```
R1(config-route-map)#route-map LOCAL_PREF permit 10
R1(config-route-map)# set local-preference 2000
R1(config-router)#neighbor 10.1.1.2 route-map LOCAL_PREF in
```

After setting the local preference for ISP1 higher than ISP2, R1 will prefer ISP1 routes. We can also go the other way, by setting ISP2 to have a value lower than ISP1. In this example, we inject an extra hop in AS_PATH as advertised to ISP2, which causes ISP1 to be preferred.

```
R1(config-route-map)#route-map AS_PATH permit 10
R1(config-route-map)#set as-path prepend 100
R1(config-router)#neighbor 10.1.2.2 route-map AS_PATH out
```

After changing an attribute, you must restart the BGP process. The best way to do this is with a soft reset by using the command **clear ip bgp * soft [in | out]**.

Filtering Routes

We can filter which routes BGP adds to the routing table using **access-lists**, **prefix-lists**, and **route-maps**.

For each neighbor, specify the list and direction the filter should be applied. For example, this code causes R1 to filter two routes from the routing table from ISP1 and ISP2.

```
R1(config-router)#neighbor 10.1.2.2 distribute-list ACL_Filter in
R1(config)#ip access-list ACL_Filter
R1(config-std-nacl)#permit 51.0.6.0 0.0.0.255
R1(config-std-nacl)#permit 51.0.7.0 0.0.0.255
```

We can also filter routes using a **prefix-list**. In this example, ISP1 stops sending the 51.0.0.0 networks to R1. Line #10 in this example is a default allow that basically says "match any network with prefix length less than or equal to 32 bits."

```
ISP1(config)#ip prefix-list FilterInet seq 5 deny 51.0.0.0/16
ISP1(config)#ip prefix-list FilterInet seq 10 permit 0.0.0.0/0 le 32
ISP1(config-router)#neighbor 10.1.1.1 prefix-list FilterInet out
```

Finally, we can filter routes using the **route-map** command. In this example, ISP2 stops sending the 51.0.6.0 and 51.0.7.0/24 networks to R1.

```
ISP1(config)#ip access-list standard 51

ISP1(config-std-nacl)#deny 51.0.6.0 0.0.0.255

ISP1(config-std-nacl)#deny 51.0.7.0 0.0.0.255

ISP1(config-std-nacl)#permit any

ISP2(config-router)#neighbor 10.1.2.1 route-map Stop51

 Out

ISP2(config)#route-map Stop51 deny 10

ISP2(config-route-map)#match ip address 51
```

Peer Groups

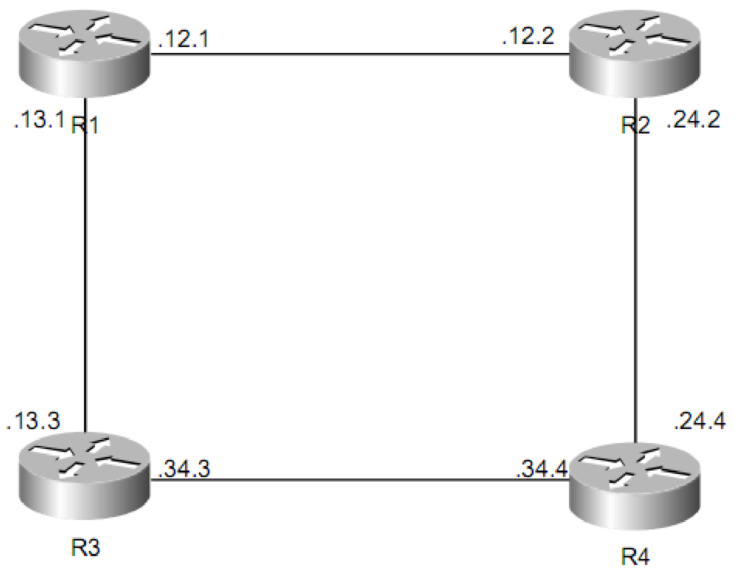

Figure 40: BGP Network Topology. *All addresses start with 192.168.*

For our next exercise in BGP configuration, consider the topology in Figure 40. When we have a number of neighbor commands to issue, we can put the neighbors in a peer group and apply the commands to the group. This means we only have to type our configuration statements once for the group instead of individually for each neighbor.

Without peer groups, our configuration for R1 might look like this:

```
R1(config-router)#neighbor 192.168.12.2 remote-as 300

R1(config-router)#neighbor 192.168.24.4 remote-as 300

R1(config-router)#neighbor 192.168.13.3 remote-as 300

R1(config-router)#neighbor 192.168.12.2 weight 10

R1(config-router)#neighbor 192.168.24.4 weight 10

R1(config-router)#neighbor 192.168.13.3 weight 10

R1(config-router)#neighbor 192.168.12.2 next-hop-self

R1(config-router)#neighbor 192.168.24.4 next-hop-self

R1(config-router)#neighbor 192.168.13.3 next-hop-self
```

We can eliminate quite a bit of typing (and potential for typos) by adding routers R2, R3, and R4 to a peer-group. First, we create the peer group and add the members.

```
R1(config-router)#neighbor 192.168.12.2 remote-as 300

R1(config-router)#neighbor 192.168.24.4 remote-as 300

R1(config-router)#neighbor 192.168.13.3 remote-as 300

R1(config-router)#neighbor our_routers peer-group

R1(config-router)#neighbor 192.168.12.2 peer-group our_routers

R1(config-router)#neighbor 192.168.24.4 peer-group our_routers

R1(config-router)#neighbor 192.168.13.3 peer-group our_routers
```

Now that this is configured, future commands can apply to all neighbors by referencing the peer-group called **our_routers**.

```
R1(config-router)#neighbor our_routers weight 10

R1(config-router)#neighbor our_routers next-hop-self
```

Verifying a eBGP Solution Implementation

Aside from **show ip route**, there are 2 main BGP **show** commands.

First, **show ip bgp summary** reveals valuable info including the local RID and AS number, and how much memory BGP is using to store prefixes. Also in the table are a list of the neighbors and the number of prefixes received. The remote AS appears, as well as the version of the BGP table, and the uptimes.

```
R1#show bgp summary

BGP router identifier 111.111.111.111, local AS number 100

BGP table version is 13, main routing table version 13

10 network entries using 1200 bytes of memory

12 path entries using 624 bytes of memory

5/4 BGP path/bestpath attribute entries using 620 bytes of memory

1 BGP AS-PATH entries using 24 bytes of memory

0 BGP route-map cache entries using 0 bytes of memory

0 BGP filter-list cache entries using 0 bytes of memory

Bitfield cache entries: current 1 (at peak 1) using 32 bytes of memory

BGP using 2500 total bytes of memory

BGP activity 77/67 prefixes, 156/144 paths, scan interval 60 secs


Neighbor  V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/
PfxRcd

1.1.1.1   4  2000    2823    2825       13    0    0 00:26:51        7

10.1.1.2  4  2000    4737    4750       13    0    0 00:27:58        2

10.1.2.2  4  2000    4748    4772       13    0    0 00:27:59        0
```

If the relationship isn't forming, the neighbors will most likely be "Active".

```
Neighbor    V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down State/PfxRcd

1.1.1.1     4  2000    2860    2860        0    0    0 00:06:47 Active

10.1.1.2    4  2000    4767    4785        0    0    0 00:07:02 Active

10.1.2.2    4  2000    4788    4815       22    0    0 01:07:02    0
```

Possible reasons for a BGP neighbor not becoming Established are:

- A near-end or far-end interface is down.
- An AS number is mismatched between the actual and expected **remote-as.**
- A neighbor is misconfigured or not-configured.
- There is no route to a neighbor.

The second major BGP show command is **show ip bgp**. This command allows you to see the BGP table, also known as the Routing Information Base or RIB.

```
R1#show ip bgp

BGP table version is 31, local router ID is 7.9.0.10

Status codes: s suppressed, d damped, h history, * valid, > best, i –
internal, r RIB-failure, S Stale

Origin codes: i - IGP, e - EGP, ? - incomplete


   Network          Next Hop            Metric LocPrf Weight Path

r> 1.1.1.1/32       1.1.1.1                  0             0 2000 i

*> 2.2.2.2/32       1.1.1.1                                0 2000 i

*> 3.3.3.3/32       1.1.1.1                                0 2000 i

*> 7.9.0.0/24       0.0.0.0                  0         32768 i

*> 7.9.10.0/24      0.0.0.0                  0         32768 i

r> 10.1.1.0/24      1.1.1.1                  0             0 2000 i

*  51.0.6.0/24      1.1.1.1                                0 2000 i

*>                  10.1.1.2                      5000     0 2000 i

*  51.0.7.0/24      1.1.1.1                                0 2000 i

*>                  10.1.1.2                      5000     0 2000 i

*> 51.0.8.0/24      1.1.1.1                                0 2000 i
```

This table also tells us the versions and RID, and then shows us the paths BGP has learned about. If a route is valid, it is denoted by a *, while the best path is denoted by a >. > routes will be put in the routing table. If a path has more than one next-hop, it is given multiple lines in the table. The remote AS is given in the table (as either i or the AS number, in this case 2000). The origin AS is represented by "i." In our example, it is either internal or from AS 2000 (listed in the Path).

The **r** state means Routing Information Base error. The most common reason for this is that BGP won't insert this route in the routing table because it already exists from a protocol with a lower Administrative Distance. We can determine this by looking at the RIB:

```
ISP1#show ip bgp rib-failure

Network       Next Hop    RIB-failure   RIB-NH Matches

10.1.1.0/24   10.1.1.1    Higher admin  n/a

                          distance
```

Most of the steps to configure BGP revolve around the **neighbor** command and how neighbors are manipulated. We can use the **debug ip bgp** to debug our neighbor process. The code below shows a neighbor coming up.

```
*Mar  4 05:41:16.368: BGP: 10.1.1.2 went from Idle to Active

*Mar  4 05:41:16.404: BGP: 10.1.1.2 open active, local address 10.1.1.1

*Mar  4 05:41:17.848: BGP: 10.1.1.2 went from Active to OpenSent

*Mar  4 05:41:17.852: BGP: 10.1.1.2 sending OPEN, version 4, my as: 100,
holdtime 180 seconds

*Mar  4 05:41:17.968: BGP: 10.1.1.2 rcv message type 1, length (excl.
header) 26

*Mar  4 05:41:17.972: BGP: 10.1.1.2 rcv OPEN, version 4, holdtime 180
seconds

*Mar  4 05:41:17.972: BGP: 10.1.1.2 rcv OPEN w/ OPTION parameter len: 16

*Mar  4 05:41:17.976: BGP: 10.1.1.2 rcvd OPEN w/ optional parameter type 2
(Capability) len 6

*Mar  4 05:41:17.976: BGP: 10.1.1.2 OPEN has CAPABILITY code: 1, length 4

*Mar  4 05:41:17.980: BGP: 10.1.1.2 OPEN has MP_EXT CAP for afi/safi: 1/1

*Mar  4 05:41:17.980: BGP: 10.1.1.2 rcvd OPEN w/ optional parameter type 2
(Capability) len 2

*Mar  4 05:41:17.984: BGP: 10.1.1.2 OPEN has CAPABILITY code: 128, length
0

*Mar  4 05:41:17.988: BGP: 10.1.1.2 OPEN has ROUTE-REFRESH capability(old)
for all address-families

*Mar  4 05:41:17.988: BGP: 10.1.1.2 rcvd OPEN w/ optional parameter type 2
(Capability) len 2

*Mar  4 05:41:17.988: BGP: 10.1.1.2 OPEN has CAPABILITY code: 2, length 0

R1#: 10.1.1.2 OPEN has ROUTE-REFRESH capability(new) for all address-
families
```

```
BGP: 10.1.1.2 rcvd OPEN w/ remote AS 2000

*Mar  4 05:41:17.988: BGP: 10.1.1.2 went from OpenSent to OpenConfirm

*Mar  4 05:41:18.012: BGP: 10.1.1.2 went from OpenConfirm to Established

*Mar  4 05:41:18.012: %BGP-5-ADJCHANGE: neighbor 10.1.1.2 Up

*Mar  4 05:42:02.524: BGP: Applying map to find origin for 7.9.0.0/24

*Mar  4 05:42:02.524: BGP: Applying map to find origin for 7.9.10.0/24
```

In the debug above, the BGP neighbor states are highlighted. These states match up to the states presented in the introduction section.

Documenting the Results of an eBGP Implementation

BGP implementation and verification documents should be created to assist in maintenance and the next phase of network evolution. By documenting BGP's logical and physical topology, you will have a solid place to begin work when changes must be made. BGP has 2 primary show commands that you can use to verify routes.

Domain 4: IPv6

Introduction

ROUTE places more emphasis on Internet Protocol version 6 than the BSCI exam did. At this point in your Cisco career, it's safe to say that Cisco will continue to push IPv6 to its professionals and experts. As the Internet grows and vendors continue to increase support for IPv6, you should be familiar with IPv6 for both your career and the exam.

In the early 1990's the Internet grew dramatically and network engineers began to realize the IPv4 address space would become completely assigned, making further expansion impossible. Several "band-aids" were used to address the immediate need for more addresses. First, classless routing did away with the concept of assigning an ISP or enterprise to an entire /8, /16, or /24 block of addresses. This allowed ISPs to be given better-fitting address blocks, resulting in fewer addresses assigned and un-utilized. The downside to this was that routing tables began to grow, as each AS owned a customized prefix. To combat this, addresses were assigned to ISPs by Regional Internet Registrars, who allowed greater summarization and smaller routing tables. This abstracted the smaller ISPs into summarized routing entries based on common geographical location. Another technique that created a short-term solution to the addressing problem was the invention of NAT/PAT. This allowed entire companies to use private addresses for their internal networks and combine their public traffic into a small number of port/IP combinations.

An IPv4 address is 32 bits in size and is represented most commonly as a series of 4 8-bit octets; such as 127.0.0.1. The IPv4 address space is 2^{32} bits, or approximately 4.2 billion addresses. Many of these addresses are reserved for testing, private networks, and multicast, further diminishing the number of routable public addresses.

IPv6 was created as a long-term fix to the IP addressing problem, among other shortcomings with IPv4. An IPv6 address is composed of 128 bits, which exponentially increases the number of IP addresses to a "practically" infinite number.

IPv6 Addressing

IPv6 addresses are represented to humans by 32 hexadecimal numbers, which are grouped into 8 groups of 4 quartets, separated by colons. Hexadecimal numbers are values 0-15, with values 10, 11, 12, 13, 14, and 15 represented by the letters A, B, C, D, E, and F, respectively. Therefore, an IPv6 address might look like:

FE32:0022:FFFE:0000:0000:2100:CF3C:09FF

There are two rules which can help shorten the address and make it more readable for humans. First, consecutive starting 0s in any quartet can be left out:

FE32:22:FFFE:0000:0000:2100:CF3C:9FF

Secondly, the a group of 0000:0000 can be eliminated and written as "::"

FE32:22:FFFE::2100:CF3C:9FF

This rule has two caveats. First, this rule may only be applied once per address. Secondly, the ending 0's in a quartet cannot be deleted. That means that both of the following writings are incorrect:

FE32:22:FFFE::21::CF3C:9FF (incorrect, can't use :: twice)
FE32:22:FFFE::21:CF3C:9FF (incorrect, can't delete low-end 0)

Like IPv4 addresses, IPv6 uses a prefix to aggregate blocks of networks. An IPv6 prefix can range from /0 to /128. IPv6 has no concept of network class and any prefix is allowed by default, so the mask is always given in CIDR format, as seen in this example:

FE32:22:FFFE::2100:CF3C:9FF /56

This example says to match the first 56 bits of the address, we must expand the address back to its original form:

FE32:0022:FFFE:0000:0000:2100:CF3C:09FF

The underlined part matches /56.

IPv6 Address Types

There are several types of IPv6 addresses, given in this table:

Address Type	Denotation	Purpose
Unique Local	FD00::/8	these are addresses similar to IPv4 RFC 1918 addresses, more commonly recognized as 10.0.0.0/8, 172.16.0.0/16 and 192.168.0.0/24.
Multicast	FF::/8	Multicasts are the new broadcasts. However, only members of the multicast group receive them, instead of all interfaces in the subnet.
Anycast	Same as Unicast	Anycast packets are defined by setting an Anycast bit; the addresses are the same as unicast.

Figure 41: IPv6 Address Types

There are also a couple of special IPv6 addresses you should know.

- ::1 /128 is the loopback interface address equivalent to 127.0.0.1.
- :: /128 is an unspecified address equivalent to 0.0.0.0.
- FF02::1 link-local all nodes multicast. Similar to IPv4 network broadcast address.
- This address is used for neighbor solicitation (similar to an IPv4 ARP). A packet sent to FF02::1 contains ICMP message 135, which asks for the Layer-2 address of an IP. It is answered by a neighbor advertisement reply.
- ::/0 is the default unicast route.

That being said, Global Unicast addresses are usually composed of a 64 bit network portion and a 64 bit host portion. 48 bits of the network prefix are dedicated to the global address space. For Global Unicast addresses, the layout is as follows (RFC 2374):

Global Routing Prefix (identifies RIR and ISP)		Subnet	Host
3 bits	45 bits	16 bits	64 bits
The first hex values are always start with 2 or 3	The RIR and ISP identifier	Enterprises can use up to 16 bits for subnetting	64 bits are available for the host

Figure 42: The IPv6 Global Unicast Address

Example:

2143:10FF:39BD:AAB2:9311:244F:0F1B:0001 /64

In this example, we can immediately identify this as a Global Unicast address because of the 2143 in front. After that, we see that 3 quartets (48 bits) have been used to identify the Regional Registrar/ISP. This means our RIR/ISP is found by 1044F:39BD:36BD.

In this example, AAB2 is the subnet. The enterprise can create roughly 65.535 subnets using this quartet. The remaining 64 bits, or the second half of the IP, are allowable for host use.

IPv6 has no concept of a broadcast number. The closest thing that IPv6 has to a broadcast in a subnet is a Multicast or Anycast.

- Multicasts are sent to all routers who are listening with the specific multicast address.
- Anycasts are sent as unicasts with the stipulation that the first router to receive it should reply. They are used for discovery.

Link-Local addresses are automatically calculated by the PC and do not require end-user configuration. They start with FE8::/10, leaving 118 bits for the host. The host bits are calculated by using a physical address of the adapter. Ethernet uses a modified MAC address as the link-local address identifier. This method of automatically calculating an IP address is known as EUI-64 (Extended Unique Identifier) addressing.

EUI-64 will insert a designated string, FFFE, in the middle of the MAC address (between the vendor code and the device ID). FFFE was chosen because it was reserved and should never appear in a MAC address.

If the MAC address is 00-11-22-33-44-55, EUI-64 will generate the host portion of the address as:

0011:22FF:FE33:4455

One more step must be completed before this EUI-64 address process is complete. The 7th bit in the first byte is the Local/Universal bit. It must be flipped from a 0 to a 1.

00 (the first two values in the MAC address) is 00000000 in binary. Switching the 7th bit gives us 00000010 = 2. Prepending the Link-Local identifier to it and applying this rule, we come up with:

FE80:0000:0000:0000:0211:22FF:FE33:4455

Or, in shorter form:

FE80::0211:22FF:FE33:4455

Address Assignment

On Cisco routers, IPv4 addresses can be either manually assigned or discovered through a service like DHCP. However, IPv6 gives a few more options for address assignment. These include:

- **DHCP** – DHCPv6 is similar to DHCPv4, except using multicasts (sent to FF02::1:2). The default router is not discovered with DHCPv6, it is discovered with NDP (see below).
    - Stateful DHCP is when the DHCP server records the IP addresses in its database. This is akin to IPv4 implementations.
    - Stateless DHCP is used when the DHCP server does not record client data, it simply hands out information.
- **Stateless Autoconfiguration** – a device uses NDP to learn about routers. The routers receive a Router Solicitation (RS) from the host and reply with a Router Advertisement (RA) containing its prefix and the prefix length. The device calculates its own host address using EUI-64. DNS servers are learned by stateless DHCPv6.
- **Static Configuration** – similar to an IPv4 static IP address, however, you can specify either the entire 128 bit IP address or just the first 64 bits and let EUI-64 calculate the host portion. Routers are discovered with NDP and DNS servers are discovered with stateless DHCPv6.

All of the protocols learn about the default router with a protocol called Network Discovery Protocol (NDP). NDP will provide a list of all the prefixes (and lengths) for a given network segment. RS messages query any routers for information and are sent to multicast address FF02::2. RA messages are replies for information and are sent to FF02::1 (all IPv6 nodes).

IPv4 interacted with Layer 2 networks via ARP in order to send data to either local nodes or the default router. NDP provides this service for IPv6 networks. NDP will send a Neighbor Solicitation (NS) message. On a network, the NS is multicast and hosts will reply if they are the node a host is looking for.

Configuring IPv6 Interfaces

In this chapter, we will keep returning to the following topology, applying different routing protocols and configurations:

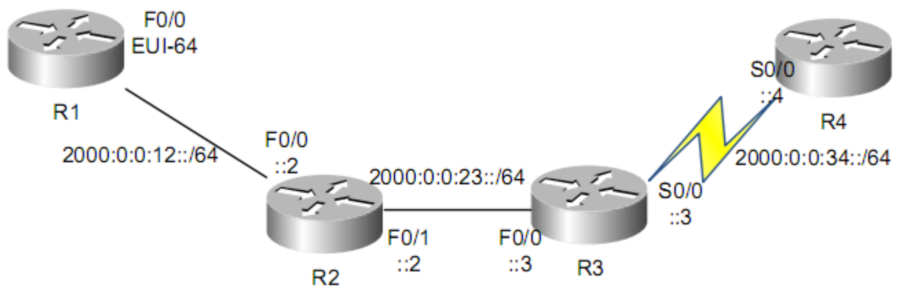

Figure 43: IPv6 topology for this chapter

Our IP addressing scheme can be described as follows:

- The 2000 in the first quartet indicates that these are Global Unicast addresses.
- The 0000:0000 in the second and third quartets indicate that our RIR and ISP have the numbers of all 0's. Although this isn't realistic on the Internet, it makes our labs easier to work with.
- The fourth quartet, bits 48-64, are for subnetting inside our organization.
- The last 64 bits are available for hosts. We will use a combination of static and EUI-64 addressing in this example.

We will begin configuring R2 and R3 - hese use static addresses and will later make up the "backbone" of our network - after we have applied routing protocols:

```
R2(config)#int fastEthernet 0/0

R2(config-if)#ipv6 address 2000:0:0:0012::2/64

R2(config-if)#int fastEthernet 0/1

R2(config-if)#ipv6 address 2000:0:0:0023::2/64

R3(config)#int fastEthernet 0/0

R3(config-if)#ipv6 address 2000:0:0:0023::3/64

R3(config)#int serial 0/0

R3(config-if)#ipv6  address 2000:0:0:0034::3/64
```

After issuing **no shutdown** on each interface, verify the connection between R2 and R3 by using a modified version of the **ping** command:

```
R3#ping ipv6 2000:0:0:0023::2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 2000:0:0:23::2, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 20/40/100 ms
```

R1 and R4 are configured with a statically defined prefix, but the last 64 bits (the host portion) are defined by EUI-64:

```
R1(config)#interface fastEthernet 0/0

R1(config-if)#ipv6 address 2000:0:0:12::/64 eui-64

R4(config)#interface serial 0/0

R4(config-if)#ipv6 address 2000:0:0:34::/64 eui-64
```

We can verify our IPv6 EUI-64 configuration with the following:

```
R4#show ipv6 interface brief

FastEthernet0/0            [administratively down/down]

Serial0/0                  [up/up]

    FE80::C003:6FF:FE98:0

    2000::34:C003:6FF:FE98:0

FastEthernet0/1            [administratively down/down]

Serial0/1                  [administratively down/down]
```

We see that since Serial 0/0 has IPv6 running on it, it has chosen a link-local address (beginning with FE80), as well as an IP with a statically configured prefix and prefix length, but an automatically chosen EUI-64 address. Because serial links have no MAC address, IOS creates one by artificially using the MAC address of one of the routers Ethernet interfaces.

We can also look at the output in more detail by using the **show ipv6 interface serial 0/0** command:

```
R4#show ipv6 interface serial 0/0

Serial0/0 is up, line protocol is up

  IPv6 is enabled, link-local address is FE80::C003:6FF:FE98:0

  No Virtual link-local address(es):

  Global unicast address(es):

    2000::34:C003:6FF:FE98:0, subnet is 2000:0:0:34::/64 [EUI]

  Joined group address(es):

    FF02::1

    FF02::1:FF98:0

  MTU is 1500 bytes

  ICMP error messages limited to one every 100 milliseconds

  ICMP redirects are enabled

  ICMP unreachables are sent

  ND DAD is enabled, number of DAD attempts: 1

  ND reachable time is 30000 milliseconds
```

From this output, we can derive some facts that are of importance to us:

- The local-link address chosen is FE80::C003:6FF:FE98:0.
- The entire global unicast address is 2000::34:C003:6FF:FE98:0. The subnet is 2000:0:0:34::/64 [EUI].
- The router has joined multicast address FF02::1. As stated, this is the IPv6 "all nodes" multicast address.

One piece of information is missing from this output. As stated previously, routers should advertise their status and reply to RS messages. Therefore, we expect to see multicast address FF02::2 applied to our interface, but we only see FF02::1. By default, Cisco routers act as hosts unless the router is told it should offer router service to IPv6. This is done with the **ipv6 unicast-routing** command. After issuing this command, we should see the following in the **show ipv6 interface serial 0/0** command:

```
...
Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF98:0
...
```

Since FF02::2 is present, this verifies that IPv6 unicast-routing has been enabled on this router.

Configuring IPv6 Routing Protocols

Now that all of our interfaces are up, we can begin to configure IPv6 routing between these routers.

RIPng

Routing Information Protocol (RIP) has been given new life as a more robust routing protocol with RIP Next-Generation (RIPng). RIPng is similar to RIPv2 in that it has the same update mechanisms, loop-prevention mechanisms, metrics, etc. RIPng differs from RIPv2 in the following ways:

- RIPng does not advertise routes for IPv4.
- RIPng uses UDP and IP on port 521.
- RIPng has no default summarization (but RIPng can advertise summary routes).
- RIPng uses FF02::9 (similar to 224.0.0.9, the RIPv2 multicast destination).
- RIPng uses IPSec-based AH/ESP authentication.

RIPng is still based on hop count and is a distance-vector protocol. The AD of RIPng is still 120, indicating that RIPng has the same trustworthiness as RIPv4. We should expect this, since the internals of RIP do not change between RIPv4 and RIPng.

In this example, we will enable RIPng on all routers using a command structure similar to the code below. This code will start a RIPng process and add the appropriate interfaces to the process.

```
R1(config)#ipv6 unicast-routing
R1(config)#ipv6 router rip Our_Network
R1(config-rtr)#interface fast 0/0
R1(config-if)#ipv6 rip Our_Network enable
```

In the routing table, we can verify that 2 RIPng routes have been learned from R4:

```
R4#show ipv6 route

IPv6 Routing Table - 5 entries

Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP

       U - Per-user Static route, M - MIPv6

       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary

       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2

       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2

       D - EIGRP, EX - EIGRP external

R   2000:0:0:12::/64 [120/3]

     via FE80::C003:FFF:FEC8:0, Serial0/0

R   2000:0:0:23::/64 [120/2]

     via FE80::C003:FFF:FEC8:0, Serial0/0

C   2000:0:0:34::/64 [0/0]

     via ::, Serial0/0

L   2000::34:C002:FFF:FEC8:0/128 [0/0]

     via ::, Serial0/0

L   FF00::/8 [0/0]

     via ::, Null0
```

The two bolded entries in the routing table show that R4 has learned routes between R2 and R3 and R2 and R1, the hop counts are indicated in the table as well.

OSPFv3

OSPFv3 is the version that supports IPv6. OSPFv3 is similar to OSPFv2 with the following exceptions:

- OSPFv3 uses multicast to all routers as FE02::5 and all DR as FF02::6.
- OSPFv3 does not require neighbors to be in the same IP subnet
- OSPFv3 allows multiple instances of OSPF per interface
- OSPFv3 uses built-in IPSec Authentication

Configure OSPFv3 in the following manner:

```
R1(config)#ipv6 unicast-routing

R1(config)#ipv6 router ospf 99

R1(config-rtr)#router-id 1.1.1.1

R1(config-rtr)#interface fast 0/0

R1(config-if)#ipv6 ospf 99 area 0
```

Unlike EIGRP for IPv6, OSPFv3 does not require a **no shutdown** command on the routing process.

We can verify OSPFv3 with the **show ipv6 route** command:

```
R1(config-if)#do show ipv6 route

IPv6 Routing Table - 5 entries

Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP

       U - Per-user Static route, M - MIPv6

       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary

       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2

       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2

       D - EIGRP, EX - EIGRP external

C   2000:0:0:12::/64 [0/0]

     via ::, FastEthernet0/0

L   2000::12:C000:10FF:FEB4:0/128 [0/0]

     via ::, FastEthernet0/0

OI  2000:0:0:23::/64 [110/20]

     via FE80::C001:10FF:FEB4:0, FastEthernet0/0

OI  2000:0:0:34::/64 [110/84]

     via FE80::C001:10FF:FEB4:0, FastEthernet0/0

L   FF00::/8 [0/0]

     via ::, Null0
```

In this example, we have put routers R2 and R3 into area 0, and routers R1 and R4 into areas 100 and 400, respectively. We can see that our routes on R1 are designated by **OI**, indicating that they are OSPF inter-area routes.

EIGRP for IPv6

EIGRP for IPv6 (sometimes known as EIGRPv6) is a Cisco-proprietary routing protocol that offers fast convergence, customizable metrics, and triggered updates. The internal workings of EIGRP are the same for both versions. If anything, EIGRP for IPv6 is simpler than the previous version for the following reasons:

- IPv6 has no concept of class, so routes are not automatically summarized.
- IPv6 requires that implementations be able to support IPSec, so security is built-in.
- EIGRP for IPv6 doesn't require that neighbors are in the same subnet, so it is more versatile.

EIGRP for IPv6 uses Protocol 88 (no UDP or TCP) with multicast address FF02::10.

Using our previous topology, we can configure each router in our EIGRP for IPv6 network with the following set of commands:

```
R1(config)#ipv6 router eigrp 99

R1(config-rtr)#no shut

R1(config-rtr)#router-id 1.1.1.1

R1(config-rtr)#int fast 0/0

R1(config-if)#ipv6 eigrp 99
```

Take note of the following facts that differ from standard EIGRP:

- The router process itself does not automatically start until a **no shutdown** command is issued. While this command usually is used under an interface, it happens to be required to start EIGRP for IPv6.
- The router-id is calculated the same way it is calculated in EIGRP for IPv4: looking for the highest up/up IP address on an interface with loopbacks taking priority. Since we do not have any IPv4 interfaces up in this case, we have manually specified the router-id. Note that the router-id is an IPv4 address. IPv6 addresses will not suffice. If a router can't determine its RID, EIGRP for IPv6 won't start.
- There is no **network** command. EIGRP for IPv6 must be implemented by interface.

After issuing these commands, we see our EIGRP for IPv6 routes appear in the IPv6 routing table.

```
R1#show ipv6 route
IPv6 Routing Table - 5 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route, M - MIPv6
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS - ISIS summary
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
       D - EIGRP, EX - EIGRP external
C   2000:0:0:12::/64 [0/0]
     via ::, FastEthernet0/0
L   2000::12:C000:10FF:FEB4:0/128 [0/0]
     via ::, FastEthernet0/0
D   2000:0:0:23::/64 [90/307200]
     via FE80::C001:10FF:FEB4:0, FastEthernet0/0
D   2000:0:0:34::/64 [90/2221056]
     via FE80::C001:10FF:FEB4:0, FastEthernet0/0
L   FF00::/8 [0/0]
     via ::, Null0
```

For all routing protocols covered, notice that the next hop listed in the IPv6 routing table is the Link-Local address.

Static Routes

To configure a static IPv6 route, use the **ipv6 route** command. You will have to specify the prefix/length of the route, plus either the outgoing interface or the next-hop-address. For example, to create a static route from R1 to the network between R2 and R3, we can use the command:

```
R1(config)#ipv6 route 2000:0:0:23::/64 2000:0:0:12::2
```

Or the command:

```
R1(config)#ipv6 route 2000:0:0:23::/64 fastEthernet 0/0
```

Configuring IPv6 Interoperation with IPv4

Very few organizations have the requirements or resources to do a "rip and replace" of IPv4 with IPv6. Even if they do, they must often connect to networks outside of their control where IPv4 must be used. Due to our dependence as users and engineers on IPv4 it is likely that the transition to IPv6 will be slow at best, and history has shown this to be true. With that in mind, we must be able to configure interoperation between our IPv4 networks and our IPv6 networks.

Three broad methods for interoperation are covered in this section. They are:

- Dual Stack Operation
- NAT Protocol Translator (NAT-PT)
- Tunneling

The goal of all of these methods is to ensure that two hosts that wish to communicate are running the same protocol. This can be either IPv4 or IPv6, but they must be able to transmit data to each other using one method or the other.

Dual Stack Operation

Dual stack operation is the simplest method of running IPv4 alongside IPv6. This interoperation method requires configuring each device to run both IPv4 and IPv6. Each router will pass traffic for IPv4 and IPv6 networks, and will maintain two routing tables, and run at least two instances of whatever routing protocols are selected.

Dual stack operation doesn't require any special configuration beyond the techniques already discussed. Implementation consists of setting up an addressing scheme and routing protocols.

While dual stacks are the simplest to set up, from an engineer's perspective, they also have some limitations. First, they do not alleviate our existing dependence on IPv4. So while adding IPv6 operations may be beneficial for future expansion or removal of IPv4 at a later date, it has limited immediate return. Secondly, configuring dual stacks requires that each router in an organization be set up for IPv6. This can be a tedious task in large organizations.

NAT-PT

NAT is the tool used in IPv4 to convert addresses from one to another. For example, NAT can convert from a public address to a statically defined address for a server in a DMZ, or it can be used to convert private RFC 1918 addresses into publically routable addresses for Internet usage in a home or office. The goal of NAT-PT is to convert IPv4 data into IPv6 data and vice-versa. This includes not only the addressing scheme but also the header.

Like IPv4 NAT, a router running NAT-PT listens for traffic destined on the "other side" of it. If the traffic meets criteria such as a source or destination address, it applies a mapping and maintains the connection's state while IPv4 and IPv6 hosts communicate. It acts as the translation point to and from the network.

NAT-PT has been deprecated recently. This technology is being replaced by dual-stack operation and the various forms of tunneling. IOS supports NAT-PT and it is still a potential concept on the exam.

Tunneling

Tunneling involves encapsulating an IPv6 packet inside an IPv4 packet and removing the encapsulation when the packet gets to the other side of the tunnel. This way, the packet is transported as IPv4 but with a complete IPv6 payload. Tunnels can be manual or automatic, and can be point-to-point or point-to-multipoint.

Static Tunnels

Manual and Generic Routing Encapsulation (GRE) Tunnels are essentially the same, with GRE having a more advanced header. GRE can carry traffic besides IP traffic. A tunnel is not a substitute for a route; a tunnel simply hides one form of traffic inside another. Manual and GRE tunnels create virtual point-to-point connections between routers that IPv6 can use when the source and destination aren't connected by an IPv6-capable path.

Configuring a tunnel involves defining a logical Tunnel adapter and defining how and what kind of traffic should flow through the tunnel.

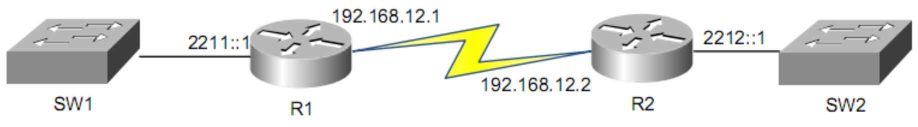

Figure 44: Point-to-Point Tunnels

```
R1(config)#int loopback 0

R1(config-if)#ip address 10.0.0.1 255.255.255.255

R1(config-if)#int tunnel 0

R1(config-if)#tunnel source loopback 0

R1(config-if)#tunnel destination 10.0.0.2

R1(config-if)#tunnel mode ipv6ip

R2(config)#int loopback 0

R2(config-if)#ip address 10.0.0.2 255.255.255.255

R2(config-if)#int tunnel 0

R2(config-if)#tunnel source loopback 0

R2(config-if)#tunnel destination 10.0.0.1

R2(config-if)#tunnel mode ipv6ip
```

As we can see, the configuration is roughly the same on each router. We have set up a symmetric tunnel so router R1 can send IPv6 packets to R2, and vice versa. R1 points to R2, and R2 points to R1. We must have IPv4 connectivity established, either through static routes or the use of a routing protocol. Because R1 and R2 are not directly connected (from an IPv6 perspective), we must also run an IPv6 routing protocol. If we wanted to run GRE instead of a manually configured tunnel, we only have to change 1 line of our configuration. The line that reads **tunnel mode ipv6ip** should read **tunnel mode gre ip**. GRE has a bit more overhead than a manual tunnel, but it allows for other protocols to be tunneled through the logical interfaces.

The main difference in the implementation and verification of Manual and GRE Tunnels is the address that is used locally in the router. In a manual tunnel, it is FE80::/96 with the last 32 bits coming from the tunnel's source IPv4 address. With GRE, EUI-64 is used, based on the router's lowest numbered MAC address.

Dynamic Tunnels

There are two methods for dynamic tunnel creation between routers. These are known as 6to4 and ISATAP. Both of these strategies are point-to-multipoint methods.

Configuring a dynamic 6to4 tunnel is similar to configuring a static tunnel. The mechanism behind 6to4 tunneling is in how it manipulates IPv6 addresses. First, the IANA has reserved the range 2002::/16 for 6to4 conversions. Although this is a Global Unicast Address, it is specifically reserved for 6to4. This command sets the tunnel's address as 2002:, followed by the hexadecimal representation of the loopback's IP address, which in this case is 10.0.0.1 (or 0a00:0001 in hex).

```
R1(config)#ipv6 unicast-routing

R1(config)#interface loopback 0

R1(config-if)#ip address 10.0.0.1 255.255.255.255

R1(config-if)#interface Tunnel 0

R1(config-if)#ipv6 address 2002:0a00:0001::1/128

R1(config-if)#tunnel source loopback 0

R1(config-if)#tunnel mode ipv6ip 6to4

R1(config)#ipv6 route 2002::/16 tunnel 0
```

The logic behind the 6to4 is simple. When a router has a packet destined for an IPv6 address, it looks in its routing table. When it sees the route matching 2002::/16, it knows to send it to tunnel 0. Because the 2002:: IPv6 IP address is built on an IPv4 address, it provides a "mapping" from IPv6 to IPv4. If our IPv4 routes have converged, then IPv6 should have a tunnel to other IPv6 devices.

When a new device is added the 6to4 network, it only has to be locally configured. The other routers don't need point-to-point connections built to transport IPv6 traffic over the IPv4 network.

ISATAP Tunnels

ISATAP Stands for Intra-Site Automatic Tunnel Addressing Protocol. It is similar to 6to4 in that it automatically configures an IP address composed of both IPv6 elements and IPv4 elements, which it then uses to pass traffic. An ISATAP tunnel has no specific reserved range, so Global Unicast Addresses can be used.

An ISATAP-generated IP address has the IPv4 address in the final quartets of the destination address (preceded by 0s). The middle of the address is the EUI-64 configuration.

```
R1(config)#ipv6 unicast-routing

R1(config)#interface loopback 0

R1(config-if)#ip address 10.0.0.1 255.255.255.255

R1(config-if)#interface Tunnel 0

R1(config-if)#ipv6 address 2000:0:0:0001::/64 eui-64

R1(config-if)#tunnel source loopback 0

R1(config-if)#tunnel mode ipv6ip isatap

R1(config)#ipv6 route 2000:0:0:0001::/64 tunnel 0
```

Domain 5: Redistribution

Introduction

This chapter covers IPv4 and IPv6 redistribution. Routes get in to a routing table one of two ways: either by being added with a **network** command or by being injected through redistribution. Redistribution is the process by which routers exchange routes from one routing protocol to another. Route redistribution blends routing domains. Routes can be shared between IGPs, or they can be converted to and from BGP routes.

Route redistribution facilitates company mergers and provides a natural place for businesses to divide their IT systems due to business requirements. Route redistribution also enables you to mix proprietary and non-proprietary routing protocols. For example, OSPF can be run on non-Cisco devices and EIGRP can be run on Cisco devices, but the routers can advertise routes from both protocols to the networks behind them. Furthermore, some company divisions have policies stating that they cannot run proprietary software or protocols, while the rest of the company has already standardized on EIGRP. In the EGP realm, selected BGP routes can be redistributed into a company's IGP, without having each router run BGP or having the overhead from a large number of BGP routes.

The basis of redistribution can be summarized in four facts:

- A router must have physical interface in each routing domain.
- Redistribution takes place from the routing table, not the topology table.
- A router must be running a routing protocol for each domain.
- Routes are redistributed with the **redistribute** command.

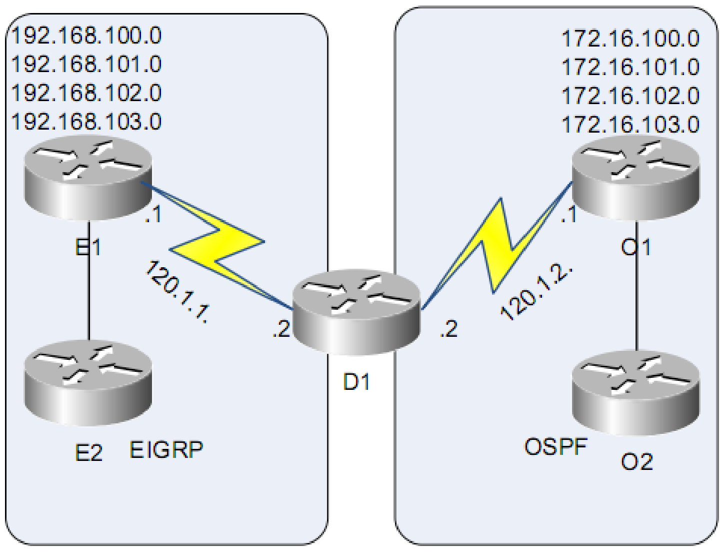

Figure 45: EIGRP and OSPF redistribution topology

Figure 45 shows the topology we will be using for our redistribution between OSPF and EIGRP. The EIGRP "native" routers are marked with E hostnames and the OSPF "native" routers are marked with O hostname. The router, D1, in the middle, will serve as our redistribution point. It has interfaces in both the EIGRP and OSPF domains. We will first inject routes from OSPF into EIGRP, and vice-versa. We will then get into more advanced IGP distribution topics, such as redundancy and loop prevention, and tweaking what parameters we advertise into a routing protocol.

Redistributing into EIGRP

The goal of redistribution is to get routes from one routing domain to another routing domain in order to facilitate inter-domain host communication. With this goal in mind, the D1 router can be made to send the 172.16.100-103 networks to EIGRP.

Before redistribution, we have enabled EIGRP on our network and verified its operation.

We used EIGRP routing process number 1000. Here is the routing table from E2:

```
E2# show ip route

Gateway of last resort is not set


C    192.168.12.0/24 is directly connected, FastEthernet0/0

D    192.168.102.0/24 [90/409600] via 192.168.12.1, 00:00:08, FastEthernet0/0

D    192.168.103.0/24 [90/409600] via 192.168.12.1, 00:00:08, FastEthernet0/0

D    192.168.100.0/24 [90/409600] via 192.168.12.1, 00:00:08, FastEthernet0/0

D    120.0.0.0/8 [90/2195456] via 192.168.12.1, 00:00:08, FastEthernet0/0

D    192.168.101.0/24 [90/409600] via 192.168.12.1, 00:00:08, FastEthernet0/0
```

Here is the routing table from O2:

```
Gateway of last resort is not set

     172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
C    172.16.12.0/24 is directly connected, FastEthernet0/0
O    172.16.101.1/32 [110/11] via 172.16.12.1, 00:16:30, FastEthernet0/0
O    172.16.100.1/32 [110/11] via 172.16.12.1, 00:16:30, FastEthernet0/0
O    172.16.103.1/32 [110/11] via 172.16.12.1, 00:16:30, FastEthernet0/0
O    172.16.102.1/32 [110/11] via 172.16.12.1, 00:16:30, FastEthernet0/0
     120.0.0.0/24 is subnetted, 1 subnets
O    120.1.2.0 [110/74] via 172.16.12.1, 00:02:21, FastEthernet0/0
```

Here is the routing table from D1:

```
Gateway of last resort is not set

D  192.168.12.0/24 [90/2195456] via 120.1.1.1, 00:09:31, Serial0/0
   172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
O  172.16.12.0/24 [110/74] via 120.1.2.1, 00:00:01, Serial0/1
O  172.16.101.1/32 [110/65] via 120.1.2.1, 00:00:01, Serial0/1
O  172.16.100.1/32 [110/65] via 120.1.2.1, 00:00:01, Serial0/1
O  172.16.103.1/32 [110/65] via 120.1.2.1, 00:00:01, Serial0/1
O  172.16.102.1/32 [110/65] via 120.1.2.1, 00:00:01, Serial0/1
D  192.168.102.0/24 [90/2297856] via 120.1.1.1, 00:09:31, Serial0/0
D  192.168.103.0/24 [90/2297856] via 120.1.1.1, 00:09:31, Serial0/0
D  192.168.100.0/24 [90/2297856] via 120.1.1.1, 00:09:31, Serial0/0
   120.0.0.0/24 is subnetted, 2 subnets
C  120.1.1.0 is directly connected, Serial0/0
C  120.1.2.0 is directly connected, Serial0/1
D  192.168.101.0/24 [90/2297856] via 120.1.1.1, 00:09:32, Serial0/0
```

Other than the router D1 itself, the hosts in either domain don't have routes to the other domain's networks.

To get the OSPF routes into EIGRP, our first example of redistribution looks like this:

```
D1(config)#router eigrp 1000

D1(config-router)#default-metric 2000 10 255 1 1500

D1(config-router)#redistribute ospf 2000
```

Injecting routes into each routing protocol is a little different For EIGRP, the values in the **default-metric** command are the *k* values of bandwidth, delay, reliability, load, and MTU. Here we have specified that routes injected into EIGRP will have a default bandwidth of 2000 kbps and a default delay of 10 ms. Because we have not tweaked our *k* value weights, the rest of the values don't matter. However, we must still specify them.

The **redistribute** command is always issued from inside the target routing process. It can be a little confusing, but get used to the idea that when issuing "redistribute", you are really asking the routing to bring in a set of routes rather than advertise them.

Here we have specified that EIGRP process ID 1000 should bring in routes from OSPF process ID 2000. We can verify this worked with the **show ip route** command on E2:

```
C  192.168.12.0/24 is directly connected, FastEthernet0/0

   172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks

D EX  172.16.12.0/24

 [170/2198016] via 192.168.12.1, 00:00:47, FastEthernet0/0

D EX  172.16.101.1/32

 [170/2198016] via 192.168.12.1, 00:00:47, FastEthernet0/0

D EX  172.16.100.1/32

 [170/2198016] via 192.168.12.1, 00:00:47, FastEthernet0/0

D EX  172.16.103.1/32

 [170/2198016] via 192.168.12.1, 00:00:47, FastEthernet0/0

D EX  172.16.102.1/32

 [170/2198016] via 192.168.12.1, 00:00:47, FastEthernet0/0

D  192.168.102.0/24 [90/409600] via 192.168.12.1, 00:52:42, FastEthernet0/0

D  192.168.103.0/24 [90/409600] via 192.168.12.1, 00:52:43, FastEthernet0/0

D  192.168.100.0/24 [90/409600] via 192.168.12.1, 00:52:43, FastEthernet0/0

D  120.0.0.0/8 [90/2195456] via 192.168.12.1, 00:52:43, FastEthernet0/0

D  192.168.101.0/24 [90/409600] via 192.168.12.1, 00:52:43, FastEthernet0/0
```

From the output, we can see that we now have "D" (EIGRP) routes for the 172.16.0.0 networks. We can't yet ping these networks, because the OSPF network has no return route to the EIGRP network. The routes in bold are marked as external routes, but are EIGRP routes nonetheless.

The administrative distance for routes redistributed into EIGRP is 170, while that of internal EIGRP routes is 90.

If we were to show the routing table on D1, we would see that our routing table is unchanged. However, the EIGRP topology table on D1 has picked up and processed the 172.16.0.0 network routes as EIGRP routes:

```
D1#show ip eigrp topology
IP-EIGRP Topology Table for AS(1000)/ID(120.1.1.2)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
 r - reply Status, s - sia Status

P 192.168.100.0/24, 1 successors, FD is 2297856
  via 120.1.1.1 (2297856/128256), Serial0/0
P 192.168.101.0/24, 1 successors, FD is 2297856
  via 120.1.1.1 (2297856/128256), Serial0/0
P 192.168.102.0/24, 1 successors, FD is 2297856
  via 120.1.1.1 (2297856/128256), Serial0/0
P 192.168.103.0/24, 1 successors, FD is 2297856
  via 120.1.1.1 (2297856/128256), Serial0/0
P 192.168.12.0/24, 1 successors, FD is 2195456
  via 120.1.1.1 (2195456/281600), Serial0/0
P 120.1.1.0/24, 1 successors, FD is 2169856
  via Connected, Serial0/0
P 120.1.2.0/24, 1 successors, FD is 1282560
  via Redistributed (1282560/0)
P 172.16.12.0/24, 1 successors, FD is 1282560
  via Redistributed (1282560/0)
P 172.16.101.1/32, 1 successors, FD is 1282560
  via Redistributed (1282560/0)
P 172.16.100.1/32, 1 successors, FD is 1282560
  via Redistributed (1282560/0)
P 172.16.103.1/32, 1 successors, FD is 1282560
  via Redistributed (1282560/0)
P 172.16.102.1/32, 1 successors, FD is 1282560
  via Redistributed (1282560/0)

D1#
```

Redistributing into OSPF

As stated, each routing protocol is a little different in what values must be set before a router will redistribute routes into it. While EIGRP requires you to specify the *k* value metrics, OSPF doesn't necessarily require any specification. However, OSPF will only accept classful networks by default, unless the **subnets** tag is appended to the end of the **redistribute** command:

```
D1(config)#router ospf 2000

D1(config-router)#redistribute eigrp 1000 subnets
```

Redistribution doesn't log any messages to let you know the redistribution is successful, we must view the routing table on an OSPF router to see if any routes have been added:

```
O2#show ip route

O E2 192.168.12.0/24 [110/20] via 172.16.12.1, 00:00:10, FastEthernet0/0

     172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks

C 172.16.12.0/24 is directly connected, FastEthernet0/0

O 172.16.101.1/32 [110/11] via 172.16.12.1, 00:52:00, FastEthernet0/0

O 172.16.100.1/32 [110/11] via 172.16.12.1, 00:52:00, FastEthernet0/0

O 172.16.103.1/32 [110/11] via 172.16.12.1, 00:52:00, FastEthernet0/0

O 172.16.102.1/32 [110/11] via 172.16.12.1, 00:52:00, FastEthernet0/0

O E2 192.168.102.0/24 [110/20] via 172.16.12.1, 00:00:10, FastEthernet0/0

O E2 192.168.103.0/24 [110/20] via 172.16.12.1, 00:00:10, FastEthernet0/0

O E2 192.168.100.0/24 [110/20] via 172.16.12.1, 00:00:10, FastEthernet0/0

     120.0.0.0/24 is subnetted, 2 subnets

O E2    120.1.1.0 [110/20] via 172.16.12.1, 00:00:10, FastEthernet0/0

O 120.1.2.0 [110/74] via 172.16.12.1, 00:37:51, FastEthernet0/0

O E2 192.168.101.0/24 [110/20] via 172.16.12.1, 00:00:11, FastEthernet0/0
```

The routes in bold are from our EIGRP routing domain. The 20 in the above routing table is OSPF's default that it applies to IGPs (OSPF applies 1 to BGP). This can be tweaked with the **default-metric** OSPF command. These routes have been brought into OSPF as External Type 2 routes (the default). Note that the metric of Type 2 routes doesn't increase inside the OSPF network. If we wanted to specify type 1 routes, we could have used this command:

```
D1(config-router)#redistribute eigrp 1000 subnets metric-type 1
```

This gives us the following output, from which we can see that the OSPF cost (10 for the 100 megabit link between O2 and O1 plus 64 for the serial link between O1 and D1, plus 20 for the original OSPF metric):

```
O E1 192.168.12.0/24 [110/94] via 172.16.12.1, 00:11:07, FastEthernet0/0

O E1 192.168.102.0/24 [110/94] via 172.16.12.1, 00:11:07, FastEthernet0/0

O E1 192.168.103.0/24 [110/94] via 172.16.12.1, 00:11:07, FastEthernet0/0

O E1 192.168.100.0/24 [110/94] via 172.16.12.1, 00:11:07, FastEthernet0/0

     120.0.0.0/24 is subnetted, 2 subnets

O E1    120.1.1.0 [110/94] via 172.16.12.1, 00:11:07, FastEthernet0/0

O E1 192.168.101.0/24 [110/94] via 172.16.12.1, 00:11:07, FastEthernet0/0
```

In summary:

- Type 2 routes have whatever metric we give them at the router doing the redistribution. They are the default.
- Type 1 routes have whatever metric we give them at the router doing the redistribution, plus the costs of the internal OSPF hops.

When would we want to use Type 1 routes? In our design, there is no benefit to using Type 1 routes, because all traffic destined to the EIGRP side only has one path, through router D1. However, if we had multiple routers acting as ASBRs and redistributing routes, we could specify that Type 1 External routes be used. This would tell OSPF to take the shortest path inside our OSPF domain before exiting it, when destined for an external network.

In O2, we can also view the OSPF database to verify that our routes are being processed as expected:

```
O2#show ip ospf database

            OSPF Router with ID (2.2.2.2) (Process ID 2000)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count

1.1.1.1         1.1.1.1         1006        0x80000009 0x001F6B 7

2.2.2.2         2.2.2.2         308         0x80000004 0x008EF2 1

100.100.100.100 100.100.100.100 766         0x80000007 0x003349 2

                Net Link States (Area 0)


Link ID         ADV Router      Age         Seq#       Checksum

172.16.12.2     2.2.2.2         308         0x80000003 0x00B7A1
```

```
                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag

120.1.1.0       100.100.100.100 766         0x80000001 0x00EEA5 0

192.168.12.0    100.100.100.100 766         0x80000001 0x00EDAB 0

192.168.100.0   100.100.100.100 766         0x80000001 0x00221F 0

192.168.101.0   100.100.100.100 766         0x80000001 0x001729 0

192.168.102.0   100.100.100.100 766         0x80000001 0x000C33 0

192.168.103.0   100.100.100.100 766         0x80000001 0x00013D 0
```

The **show ip ospf database** command tells us that these routes have been inserted as Type 5 LSAs. These are LSAs that are external to OSPF, which are LSAs generated by an ASBR. Because D1 interfaces with another routing protocol, it is an ASBR for the OSPF process.

We are now able to ping from the EIGRP side to the OSPF side and vice versa:

```
E2#ping 172.16.100.1

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 172.16.100.1, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 20/40/88 ms
```

OSPF limits what routers can perform redistribution:

- OSPF prohibits stubby and totally stubby areas from performing distribution.
- OSPF allows Not-So-Stubby Area redistribution.

Although Type 5 LSAs are not permitted in stub areas of any variety, NSSA can transport a Type 5 LSA as a Type 7 LSA. When sent into a regular area, it will be converted back to a Type 5 LSA at the ABR.

Tweaking Redistribution and Route-Maps

You can tweak the redistribution process for a variety of reasons:

- Filtering routes
- Optimizing the cost of injected routes
- Inter-protocol loop prevention
- Classifying traffic for later use
- Policy-based routing

The easiest way to tweak redistribution is using the **metric** command at the end of the **redistribute** command. This will tweak a specific metric for a specific routing process:

```
D1(config-router)# redistribute ospf 2000 metric 100000 100 255 1 1500
```

However, this in-line tweaking is limited in what it can do. At the CCNP level, the main tool you should be aware of for tweaking routes is the **route-map**. Although discussed earlier in the BGP chapter, we will revisit it in the context of redistribution to see how a route-map can be used to manipulate routes.

In BGP, we used route-maps to apply either inbound or outbound path attribute variables, such as setting the weight of a path, or appending AS hops in the AS_PATH variable. We can do similar things with a route-map when doing redistribution.

Applying a route-map is simple enough. In redistribution, we can append **route-map <name>** to the end of our redistribute statement, as in this example:

```
D1(config)#router eigrp 1000

D1(config-router)#redistribute ospf 2000 route-map Our_RM
```

Appending **route-map** to the end of the command basically tells EIGRP to import the routes from OSPF process number 2000, but apply this route-map first.

A route-map is like an aggregated access-list that can apply settings to it and match a variety of packets. Like an ACL, a route-map will be processed sequentially, and in fact uses sequence numbers like ACLs. If multiple match statements are used in the same sequence, IOS uses AND logic, so each one must be matched. If a route-map has a single match statement with multiple criteria, OR logic will be used. Route-maps have an implicit **deny all** at the end of them.

In our example, assume we want to stop the routes for networks 192.168.102.0 and 192.168.103.0 from getting in to our OSPF network. Further assume that we want to stop the networks 172.16.102.0 and 172.16.103.0 from getting into our EIGRP network. Like ACL logic, we can either choose to permit the matching routes and deny the rest, or deny the matching routes and permit the rest.

First, to deny the 192.168.102 & 103 networks and permit the rest, we can use the route-ap with the following syntax:

```
D1(config)#access-list 10 deny 192.168.102.0 0.0.0.255

D1(config)#access-list 10 deny 192.168.103.0 0.0.0.255

D1(config)#access-list 10 permit any

D1(config)#route-map Stop-192-168-102-103 permit 15

D1(config-route-map)#match ip address 10

D1(config)router ospf 2000

D1(config-router) redistribute eigrp 1000 subnets route-map Stop-192-168-
102-103
```

We can see from the output on O2, that we have filtered the 102 and 103 networks. They are not longer being advertised to the OSPF routing domain:

```
O E2 192.168.12.0/24 [110/20] via 172.16.12.1, 00:06:22, FastEthernet0/0

     172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks

C 172.16.12.0/24 is directly connected, FastEthernet0/0

O 172.16.101.1/32 [110/11] via 172.16.12.1, 02:52:52, FastEthernet0/0

O 172.16.100.1/32 [110/11] via 172.16.12.1, 02:52:52, FastEthernet0/0

O 172.16.103.1/32 [110/11] via 172.16.12.1, 02:52:52, FastEthernet0/0

O 172.16.102.1/32 [110/11] via 172.16.12.1, 02:52:52, FastEthernet0/0

O E2 192.168.100.0/24 [110/20] via 172.16.12.1, 00:06:22, FastEthernet0/0

     120.0.0.0/24 is subnetted, 2 subnets

O E2    120.1.1.0 [110/20] via 172.16.12.1, 00:06:22, FastEthernet0/0

O 120.1.2.0 [110/74] via 172.16.12.1, 02:38:43, FastEthernet0/0

O E2 192.168.101.0/24 [110/20] via 172.16.12.1, 00:06:22, FastEthernet0/0
```

A route-map must match a set of addresses. Although this can be done a number of ways, we have chosen to use a standard ACL numbered 10. The route-map named Stop-192-168-102-103 is a **permit** line, telling the redistribution system that these addresses are okay. However, the access-list matches the 102 and 103 networks and throws them out of the group of okay addresses, before redistribution has a chance to process them.

By looking at the ACLs, we can see what traffic is hitting our route-map:

```
D1#show access-list

Standard IP access list 66

    10 deny   192.168.102.0, wildcard bits 0.0.0.255 (1 match)

    20 deny   192.168.103.0, wildcard bits 0.0.0.255 (1 match)

    30 permit any (5 matches)
```

However, the **show route-map** command is of limited value, because it doesn't show if any traffic is hitting our route-map:

```
D1#show route-map

route-map Stop-192-168-102-103, permit, sequence 15

  Match clauses:

    ip address (access-lists): 66

  Set clauses:

  Policy routing matches: 0 packets, 0 bytes
```

Besides filtering routes, the other thing we can do with a route-map is set some value to matching traffic. In the next example, we want to adjust the metrics of route 172.168.102.0 while keeping the rest the same. We can apply a route-map on our redistribution router so that routes sent to the EIGRP process are tweaked. In this example, we will use a **prefix-list** to match routes. These were also used in the BGP chapter.

```
D1(config)#ip prefix-list Network-102 permit 172.16.102.0/24 ge 25 le 32

D1(config)#route-map Change-102-Metric permit 10

D1(config-route-map)#match ip address prefix-list Network-102

D1(config-route-map)#set metric 10 23420 255 1 1500

D1(config-route-map)#router eigrp 1000

D1(config-router)#redistribute ospf 2000 route-map Change-102-Metric


E2#show ip eigrp topology 172.16.100.1/32

IP-EIGRP (AS 1000): Topology entry for 172.16.100.1/32

  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 256599552

  Routing Descriptor Blocks:

  192.168.12.1 (FastEthernet0/0), from 192.168.12.1, Send flag is 0x0

      Composite metric is (256599552/256573952), Route is External

      Vector metric:

   Minimum bandwidth is 10 Kbit

        Total delay is 23420 microseconds

        Reliability is 255/255

        Load is 1/255

        Minimum MTU is 1500

        Hop count is 2

      External data:

        Originating router is 120.1.1.2

        AS number of route is 2000

        External protocol is OSPF, external metric is 65

        Administrator tag is 0 (0x00000000)
```

```
E2(config-route-map)#set ip ?

  address     Specify IP address

  default     Set default information

  df          Set DF bit

  next-hop    Next hop address

  precedence  Set precedence field

  qos-group   Set QOS Group ID

  tos         Set type of service field
```

For example, we could set the IP precedence to a group of traffic, or the ToS bit for traffic queuing. We can also specify the **next-hop** address. Although not part of redistribution, you can match traffic from a particular source and send it to one next-hop address, and match traffic from another source to send it to another next-hop. This would allow you to route based on source address rather than destination address and create some interesting network configurations. You could send your voice traffic out one interface and your data traffic out another, or you could send your businesses accounting traffic to one next-hop and your manufacturing traffic to another, without creating VLANs. More on this is discussed in the next chapter, "Path Control".

Redistribution and Routing Loops

Although all routing protocols have their own loop-prevention mechanisms built in, these mechanisms are helpless when routes are redistributed. We must build in additional logic to the Layer 3 design so that routes aren't redistributed repeatedly.

There is an exception to the routing problem when one of the two domains is EIGRP. EIGRP has a built-in route prevention mechanism. Recall that a router will only put the best route to a destination in its routing table. EIGRP Internal routes have an AD of 90 and External routes have an AD of 170. Therefore, the external routes will never be re-added to the routing table, and the loop is broken.

RIPv2 and OSPF have identical AD for both their internal and externally received routes, so they are not automatically immune to redistribution loops.

In our original design, we rely on a single router (D1) to not only provide the connection between the EIGRP and OSPF domains, but also to handle all of the redistribution between the two systems. A better design is to introduce another router, D2, to provide redundancy. However, this design is not as easy as it sounds. If D1 distributes a route, and it becomes part of the routing domain, who is to say that D2 won't pick up the route and send it back to its original domain?

Also imagine that our company added a policy stating that we cannot run proprietary protocols. Therefore, the EIGRP domain must be converted to something else. In this case, we will run RIPv2 on all of our former EIGRP routers.

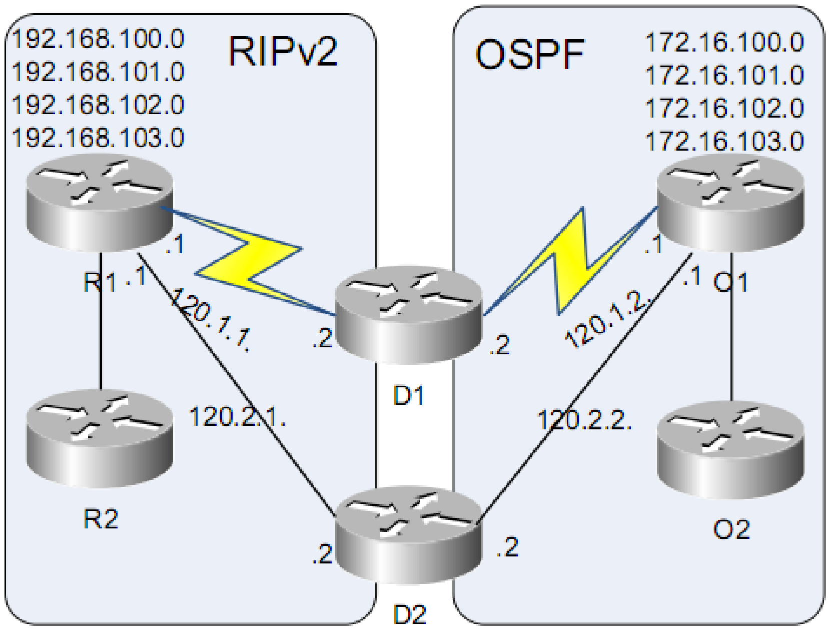

Figure 46: Adding Routers for Redundancy.
D2 has been added for redundancy and EIGRP has been converted to RIP

First, we will configure two-way redistribution:

```
D1(config)#router rip

D1(config-router)#redistribute ospf 2000 metric 2

D2(config)#router rip

D2(config-router)#redistribute ospf 2000 metric 2

D1(config-router)#router ospf 2000

D1(config-router)#redistribute rip subnets

D2(config-router)#router ospf 2000

D2(config-router)#redistribute rip subnets
```

When redistributing routes in to RIP, there is one caveat. We must specify the metric of the routes. IOS will let us enter the redistribution command without the metric, but redistribution simply won't work. This can create a situation that is very difficult to troubleshoot. If we do not specify the metric in line with the redistribute command, we can use the **default-metric** rip subcommand. Either way, RIP sees these routes as having an infinite metric by default, so we must explicitly define a metric that is reasonable for our design.

Now let's look at the routing table of D2:

```
O E2 192.168.12.0/24 [110/20] via 120.2.2.1, 00:04:31, FastEthernet0/1
     172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
O       172.16.12.0/24 [110/20] via 120.2.2.1, 00:15:20, FastEthernet0/1
O       172.16.100.1/32 [110/11] via 120.2.2.1, 00:15:20, FastEthernet0/1
O       172.16.101.1/32 [110/11] via 120.2.2.1, 00:15:20, FastEthernet0/1
O       172.16.100.1/32 [110/11] via 120.2.2.1, 00:15:20, FastEthernet0/1
O       172.16.103.1/32 [110/11] via 120.2.2.1, 00:15:20, FastEthernet0/1
O       172.16.102.1/32 [110/11] via 120.2.2.1, 00:15:20, FastEthernet0/1
O E2 192.168.102.0/24 [110/20] via 120.2.2.1, 00:04:31, FastEthernet0/1
O E2 192.168.103.0/24 [110/20] via 120.2.2.1, 00:04:31, FastEthernet0/1
O E2 192.168.100.0/24 [110/20] via 120.2.2.1, 00:04:31, FastEthernet0/1
     120.0.0.0/24 is subnetted, 4 subnets
C       120.2.2.0 is directly connected, FastEthernet0/1
O E2    120.1.1.0 [110/20] via 120.2.2.1, 00:04:31, FastEthernet0/1
C       120.2.1.0 is directly connected, FastEthernet0/0
O       120.1.2.0 [110/74] via 120.2.2.1, 00:15:21, FastEthernet0/1
O E2 192.168.101.0/24 [110/20] via 120.2.2.1, 00:04:32, FastEthernet0/1
```

Notice that our RIP-side routes are now being sent through our OSPF routers, with the next hop given as router O1. This results in sub-optimal routing. This is because OSPF has a lower AD than RIP (110 vs. 120, respectively). Therefore, when D2 heard an advertisement for the RIP routes we injected in to OSPF by D1, it replaced those entries in its routing table with OSPF entries. To illustrate this, we know that D2 should be able to make a direct connection to R1, which has the 192.168.100.1 interface. But traceroute shows a different story:

```
D2#traceroute 192.168.100.1


Type escape sequence to abort.

Tracing the route to 192.168.100.1


  1 120.2.2.1 64 msec 48 msec 44 msec

  2 120.1.2.2 32 msec 28 msec 20 msec

  3 120.1.1.1 64 msec *  76 msec

D2#
```

We can see that the traffic travels back into our OSPF domain to O1, and then to D1, before finally going into the RIP side of our network.

Upon closer investigation, we can see that this route, although originating on the RIP side, is known via OSPF:

```
D2#show ip route 192.168.100.0

Routing entry for 192.168.100.0/24

  Known via "ospf 2000", distance 110, metric 20, type extern 2,
forward metric 74

  Redistributing via rip

  Advertised by rip metric 2

  Last update from 120.2.2.1 on FastEthernet0/1, 00:15:24 ago

  Routing Descriptor Blocks:

  * 120.2.2.1, from 111.111.111.111, 00:15:24 ago, via FastEthernet0/1

      Route metric is 20, traffic share count is 1
```

D1's RID is in bold above, indicating that we have received a redistributed route from D1 rather than a RIP route directly from R1.

In this configuration, we must prevent a routing loop from occurring. The easiest way to do this is to set the External AD in OSPF higher than the internal AD (like EIGRP's internal and external routes) so that we guarantee our OSPF routers will use their internal routes first. Because redistribution is based on the routes in the routing table and not the topology table, only the best routes will make it in to the routing table.

This can be accomplished by running the **distance ospf external 203** OSPF subcommand. By setting the AD of externally-learned OSPF routes to 203 OSPF will prefer its internal routes for any specific subnet/prefix. We can verify this by looking at the routing table on D2 again:

```
Gateway of last resort is not set

R    192.168.12.0/24 [120/1] via 120.2.1.1, 00:00:03, FastEthernet0/0

     172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks

O       172.16.12.0/24 [110/20] via 120.2.2.1, 00:00:03, FastEthernet0/1

O       172.16.100.1/32 [110/11] via 120.2.2.1, 00:00:03, FastEthernet0/1

O       172.16.101.1/32 [110/11] via 120.2.2.1, 00:00:03, FastEthernet0/1

O       172.16.100.1/32 [110/11] via 120.2.2.1, 00:00:03, FastEthernet0/1

O       172.16.103.1/32 [110/11] via 120.2.2.1, 00:00:03, FastEthernet0/1
```

```
O       172.16.102.1/32 [110/11] via 120.2.2.1, 00:00:03, FastEthernet0/1
R    192.168.102.0/24 [120/1] via 120.2.1.1, 00:00:03, FastEthernet0/0
R    192.168.103.0/24 [120/1] via 120.2.1.1, 00:00:03, FastEthernet0/0
R    192.168.100.0/24 [120/1] via 120.2.1.1, 00:00:03, FastEthernet0/0
     120.0.0.0/24 is subnetted, 4 subnets
C       120.2.2.0 is directly connected, FastEthernet0/1
R       120.1.1.0 [120/1] via 120.2.1.1, 00:00:03, FastEthernet0/0
C       120.2.1.0 is directly connected, FastEthernet0/0
O       120.1.2.0 [110/74] via 120.2.2.1, 00:00:04, FastEthernet0/1
R    192.168.101.0/24 [120/1] via 120.2.1.1, 00:00:03, FastEthernet0/0
```

And verify by traceroute:

```
D2#traceroute 192.168.100.1

Type escape sequence to abort.
Tracing the route to 192.168.100.1
  1 120.2.1.1 76 msec *  72 msec
D2#
```

In summary, we have told the OSPF process on our distribution routers that External OSPF routes should be treated with lower preference than Internal OSPF routes (and RIP routes). Therefore, our RIP routes appeared back in the routing table, and we not only removed RIP traffic from being routed through the OSPF domain, we also decreased the hop count by 2 hops.

The **distance ospf external 200** OSPF subcommand will affect all routes that OSPF receives, and should be used with caution. A better solution might be to only set those routes we expect to see from the RIP side to a higher AD. We can do this on a route-by-route basis using an access-list. Configure the following on D1 and D2:

```
D2(config)#access-list 10 permit 192.168.100.0 0.0.0.255
D2(config)#access-list 10 permit 192.168.101.0 0.0.0.255
D2(config)#router ospf 2000
D2(config-router)#distance 200 0.0.0.0 255.255.255.255 10
```

This should assign, from OSPF's perspective, a distance of 200 for all routes it receives that match access-list 10. Here is the routing table from D2:

```
O E2 192.168.12.0/24 [110/20] via 120.2.2.1, 00:00:03, FastEthernet0/1
     172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks
O 172.16.12.0/24 [110/20] via 120.2.2.1, 00:00:06, FastEthernet0/1
O 172.16.101.1/32 [110/11] via 120.1.2.1, 00:00:06, Serial0/1
O 172.16.101.1/32 [110/11] via 120.2.2.1, 00:00:06, FastEthernet0/1
O 172.16.100.1/32 [110/11] via 120.2.2.1, 00:00:06, FastEthernet0/1
O 172.16.103.1/32 [110/11] via 120.2.2.1, 00:00:06, FastEthernet0/1
O 172.16.102.1/32 [110/11] via 120.2.2.1, 00:00:06, FastEthernet0/1
O E2 192.168.102.0/24 [110/20] via 120.2.2.1, 00:00:03, FastEthernet0/1
O E2 192.168.103.0/24 [110/20] via 120.2.2.1, 00:00:03, FastEthernet0/1
R    192.168.100.0/24 [120/1] via 120.2.1.1, 00:00:12, FastEthernet0/0
     120.0.0.0/24 is subnetted, 4 subnets
C 120.2.2.0 is directly connected, FastEthernet0/1
O E2    120.1.1.0 [110/20] via 120.2.2.1, 00:00:06, FastEthernet0/1
C 120.2.1.0 is directly connected, FastEthernet0/0
O 120.1.2.0 [110/74] via 120.2.2.1, 00:00:07, FastEthernet0/1
R    192.168.101.0/24 [120/1] via 120.2.1.1, 00:00:13, FastEthernet0/0
```

We have now told D1 and D2 to use RIP's advertised route to reach the 192.168.100 and 101 networks, while the other networks (103 and 104) are still being advertised back into our redistribution routers from the OSPF side. We set the OSPF-side AD for 100 and 101 to 200, making them less desirable than RIP's original AD of 120.

We can also verify this with the **show ip protocols** command:

```
Routing Protocol is "ospf 2000"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 111.111.111.111
  It is an autonomous system boundary router
  Redistributing External Routes from,
    rip, includes subnets in redistribution
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
  Routing on Interfaces Configured Explicitly (Area 0):
    Serial0/1
 Reference bandwidth unit is 100 mbps
  Routing Information Sources:
    Gateway         Distance      Last Update
    222.222.222.222      200      00:00:12
    172.16.103.1         200      00:00:12
  Distance: (default is 110)
    Address         Wild mask       Distance  List
    0.0.0.0         255.255.255.255      200  10
```

For the example, I've removed the access-lists and set the ADs back to default. **show ip protocols** now returns back to normal:

```
D1#show ip proto
Routing Protocol is "ospf 2000"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 111.111.111.111
  It is an autonomous system boundary router
  Redistributing External Routes from,
    rip, includes subnets in redistribution
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
  Routing on Interfaces Configured Explicitly (Area 0):
    Serial0/1
 Reference bandwidth unit is 100 mbps
  Routing Information Sources:
    Gateway         Distance      Last Update
    222.222.222.222      110      00:09:04
    172.16.103.1         110      00:09:04
  Distance: (default is 110)

Routing Protocol is “rip”
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 6 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: ospf 2000, rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Serial0/0             2     2
    Serial0/1             2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    120.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    120.1.1.1            120      00:00:05
  Distance: (default is 120)
```

Route Tags

Another method we can use to filter routes is based on an arbitrary tag that we assign to outbound advertisements. After we tag outbound advertisements, we can tell a routing process not to inject advertisements carrying that tag into the table. This is the same logic BGP uses to determine if a loop exists; if BGP sees its own AS listed in the AS_PATH attribute of a path advertisement, it knows it is receiving the path back via some kind of loop, and the advertisement is ignored. Unfortunately, IGPs do not have an automatic process of doing this for redistribution.

The first step in tagging routes is to apply the tags. Consider the following example code on D1 and D2:

```
D1(config)#access-list 99 permit 192.168.100.0

D1(config#)access-list 99 permit 192.168.102.0


D2(config-route-map)#route-map tag-rip deny 10

D2(config-route-map)#match tag 4444

D2(config-route-map)#route-map tag-rip permit 20

D2(config-route-map)#match ip address 99

D2(config-route-map)#set tag 4444

D1(config-router)#redistribute rip subnets route-map tag-rip
```

The access-list *permits* these networks to be matched by the route-map. If we used the deny statement in our ACL, the traffic would not be matched. Traffic matched with a **permit** is included in route-maps parent logic, while denied traffic is not. We have specified we want to match the 2 RIP networks with our ACL and set their tags to 4444.

To verify that this worked, go to any OSPF router and use **show ip route 192.168.100.0**. The route tag should be present on the router.

```
O2#show ip route 192.168.100.0

Routing entry for 192.168.100.0/24

  Known via "ospf 2000", distance 110, metric 20

  Tag 4444, type extern 2, forward metric 74

  Last update from 172.16.12.1 on FastEthernet0/0, 00:06:53 ago

  Routing Descriptor Blocks:

  * 172.16.12.1, from 111.111.111.111, 00:06:53 ago, via FastEthernet0/0

      Route metric is 20, traffic share count is 1

      Route tag 4444
```

As we can see, the tags are transitive – they will follow the advertisements wherever they go. At this point in time, we haven't actually done anything besides apply an arbitrary tag to our routes. To prevent routing loops, we could set the metric using the **set metric** route-map subcommand, but since this section focuses on tags, we will work with those.

Logically if D1 and D2 see a route that already has either a tag of 1111 or 2222 set, it will know those routes have already been redistributed and it will reject them.

Distribute-Lists

One option we can use for filtering routes is to use the **distribute-list** routing process subcommand. Distribute list works like an ACL for either incoming or outgoing redistribution.

If we want to stop our 192.168.100 and 101 networks from entering the OSPF domain, we can use configure a distribute list in either the inbound or outbound direction (the same configuration applies on R2):

```
D1(config)#access-list 55 deny 192.168.100.0

D1(config)#access-list 55 deny 192.168.101.0

D1(config)#access-list 55 permit any

D1(config)#router ospf 2000

D1(config-router)#distribute-list 55 out

D1(config-router)#redistribute rip subnets
```

If we use **show ip route** on any router in our OSPF domain, the 192.168.100 and 101 networks will not appear.

Static, Connected, and Default

So far, we have only discussed redistributing routes from one routing process into another. However, we can also redistribute static routes, networks that are connected directly to a router, and we can distribute a default route.

The static route will not be redistributed if either:

- The next hop address is unreachable.
- The outbound interface is down.

You can use the **redistribute static** command to redistribute static routes. How it works depends on the routing protocol used. For EIGRP, redistribute static will be an external route, while "injecting" the static route into EIGRP with the **network** command will result in an Internal EIGRP route.

Because the **redistribute static** or **redistribute connected** commands might add more networks than desired, they can be filtered with either a route-map or by the distribute-list command, just like any dynamic routing protocol.

Sometimes, in addition to redistributing our dynamic protocols from one domain to another, we want to distribute a default route. This is usually done by adding a static route and redistributing it into the routing protocol. In OSPF and RIP, a router (an ASBR in OSPF) can send out a default route to the rest of the areas. If the router doesn't have a default route in the table, but we want to send a default route anyway, we can use the **always** clause at the end of our **default-information** statement. . It can be done like this:

```
D1(config)#router ospf 2000

D1(config)#default-information originate always
```

This command tells the OSPF domain about exit points from the ASBR. The syntax is similar for RIP. The routing table in our OSPF domain now contains:

```
Gateway of last resort is 172.16.12.1 to network 0.0.0.0

172.16.0.0/16 is variably subnetted, 5 subnets, 2 masks

C  172.16.12.0/24 is directly connected, FastEthernet0/0

O  172.16.101.1/32 [110/11] via 172.16.12.1, 04:03:36, FastEthernet0/0

O  172.16.100.1/32 [110/11] via 172.16.12.1, 04:03:36, FastEthernet0/0

O  172.16.103.1/32 [110/11] via 172.16.12.1, 04:03:36, FastEthernet0/0

O  172.16.102.1/32 [110/11] via 172.16.12.1, 04:03:36, FastEthernet0/0

O E2 192.168.102.0/24 [110/20] via 172.16.12.1, 00:37:37, FastEthernet0/0

O E2 192.168.100.0/24 [110/20] via 172.16.12.1, 00:37:37, FastEthernet0/0

120.0.0.0/24 is subnetted, 2 subnets

O  120.2.2.0 [110/20] via 172.16.12.1, 01:47:58, FastEthernet0/0

O  120.1.2.0 [110/74] via 172.16.12.1, 04:02:21, FastEthernet0/0

O*E2 0.0.0.0/0 [110/1] via 172.16.12.1, 00:00:01, FastEthernet0/0
```

IPv6 Redistribution

IPv6 redistribution is similar to IPv4 redistribution, with a few differences:

- IPv4 route-maps and prefix-lists must be updated to IPv6 addressing schemes.
- The **include-connected** clause at the end of a **redistribute** command is needed to redistribute both connected and dynamically-learned routes.
- Since IPv6 is classless, the **subnets** clause is not required for OSPF redistribution.
- Because each interface can have a variety of IPv6 addresses, /128 (host) routes are not redistributed.

Our IPv6 network for this example will use the same topology as we used before. In this example, we will redistribute OSPF into EIGRP and EIGRP into OSPF.

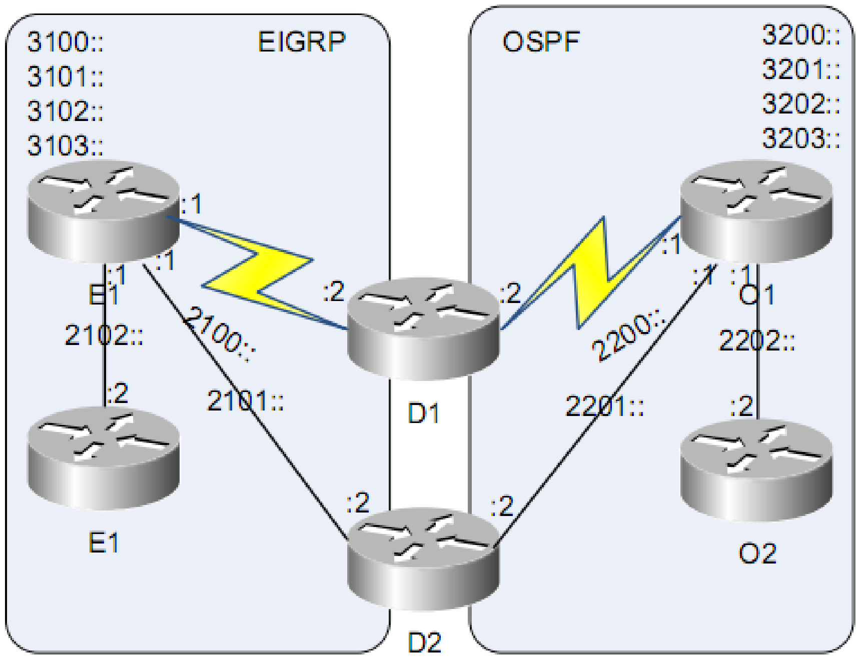

Figure 47: IPv6 Topology. *All prefix lengths are /64.*

For convenience and IPv6 review, the EIGRP and OSPF configurations for routers E1 and O1 are given here:

```
E1(config)#int loop 3100
E1(config-if)#ipv6 address 3100::1/64
E1(config-if)#ipv6 eigrp 1000
E1(config-if)#int loop 3101
E1(config-if)#ipv6 address 3101::1/64
E1(config-if)#ipv6 eigrp 1000
E1(config-if)#int loop 3102
E1(config-if)#ipv6 address 3102::1/64
E1(config-if)#ipv6 eigrp 1000
E1(config-if)#int loop 3103
E1(config-if)#ipv6 address 3103::1/64
E1(config-if)#ipv6 eigrp 1000
E1(config-if)#int serial 0/0
```

```
E1(config-if)#ipv6 address 2100::1/64

E1(config-if)#ipv6 eigrp 1000

E1(config-if)#no shut

E1(config-if)#int fast0/0

E1(config-if)#ipv6 address 2101::1/64

E1(config-if)#ipv6 eigrp 1000

E1(config-if)#no shut

E1(config-if)#int fast 0/1

E1(config-if)#ipv address 2102::1/64

E1(config-if)#ipv6 eigrp 1000

E1(config-if)#no shut

E1(config-if)#ipv6 unicast-routing

E1(config)#ipv6 router eigrp 1000

E1(config-rtr)#router-id 1.1.1.1

E1(config-rtr)#no shut
```

Recall that EIGRP for IPv6 has 2 relevant changes for this configuration when compared to IPv4:

- There is no **network** command so we must add interfaces to the EIGRP process.
- The routing process itself must be started with a **no shut** command.

The OSPF configuration for O1 is given here:

```
O1(config)#ipv6 unicast-routing

O1(config)#ipv6 router ospf 2000

O1(config-rtr)#router-id 1.1.1.1

O1(config)#int loop 3200

O1(config-if)#ipv6 address 3200::1/64

O1(config-if)#ipv6 ospf 2000 area 0

O1(config-rtr)#int loop 3200

O1(config-if)#ipv6 ospf 2000 area 0

O1(config-if)#int loop 3201

O1(config-if)#ipv6 address 3201::1/64
```

```
O1(config-if)#ipv6 ospf 2000 area 0

O1(config-if)#int loop 3202

O1(config-if)#ipv6 address 3202::1/64

O1(config-if)#ipv6 ospf 2000 area 0

O1(config-if)#int loop 3203

O1(config-if)#ipv6 address 3203::1/64

O1(config-if)#ipv6 ospf 2000 area 0

O1(config)#int serial 0/0

O1(config-if)#ipv6 address 2200::1/64

O1(config-if)#ipv6 ospf 2000 area 0

O1(config-if)#no shut

O1(config-if)#int fast 0/0

O1(config-if)#ipv6 address 2201::1/64

O1(config-if)#ipv6 ospf 2000 area 0

O1(config-if)#no shut

O1(config-if)#int fast 0/1

O1(config-if)#ipv6 address 2202::1/64

O1(config-if)#ipv6 ospf 2000 area 0

O1(config-if)#no shut
```

OSPFv3 will not start routing unless the router can determine an appropriate router ID, either by explicit definition or by using the usual rules.

After configuring each IPv6 routing domain, we can verify our redistribution routers have all of the routes using **show ipv6 route**.

```
C 2100::/64 [0/0] via ::, Serial0/0

L 2100::2/128 [0/0] via ::, Serial0/0

D 2101::/64 [90/2195456] via FE80::C009:14FF:FEEC:0, Serial0/0

D 2102::/64 [90/2195456] via FE80::C009:14FF:FEEC:0, Serial0/0

C 2200::/64 [0/0] via ::, Serial0/1

L 2200::2/128 [0/0] via ::, Serial0/1

O 2201::/64 [110/74] via FE80::C00A:FFF:FE28:0, Serial0/1

O 2202::/64 [110/74] via FE80::C00A:FFF:FE28:0, Serial0/1

D 3100::/64 [90/2297856] via FE80::C009:14FF:FEEC:0, Serial0/0

D 3101::/64 [90/2297856] via FE80::C009:14FF:FEEC:0, Serial0/0

D 3102::/64 [90/2297856] via FE80::C009:14FF:FEEC:0, Serial0/0

D 3103::/64 [90/2297856] via FE80::C009:14FF:FEEC:0, Serial0/0

O 3200::1/128 [110/64] via FE80::C00A:FFF:FE28:0, Serial0/1

O 3201::1/128 [110/64] via FE80::C00A:FFF:FE28:0, Serial0/1

O 3202::1/128 [110/64] via FE80::C00A:FFF:FE28:0, Serial0/1

O 3203::1/128 [110/64] via FE80::C00A:FFF:FE28:0, Serial0/1

L FF00::/8 [0/0] via ::, Null0
```

The **redistribute** command works the same in IPv6 as it does in IPv4. As before, we must specify a seed metric for injecting OSPF routes into EIGRP:

```
D1(config)#ipv6 router eigrp 1000

D1(config-rtr)#redistribute ospf 2000 metric 1544 20 255 1 1500


D2(config)#ipv6 router eigrp 1000

D2(config-rtr)#redistribute ospf 2000 metric 1544 20 255 1 1500
```

We can't verify our work with a ping yet, because the OSPF routers have no knowledge of how to send return data to the EIGRP side. However, we can verify our work by looking at D1 and D2 with the **show ipv6 protocols** command:

```
IPv6 Routing Protocol is "connected"

IPv6 Routing Protocol is "static"

IPv6 Routing Protocol is "eigrp 1000"

  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0

  EIGRP maximum hopcount 100

  EIGRP maximum metric variance 1

  Interfaces:

    Serial0/0

  Redistribution:

    Redistributing protocol ospf 2000 with metric 0

  Maximum path: 16

  Distance: internal 90 external 170

IPv6 Routing Protocol is "ospf 2000"

  Interfaces (Area 0):

    Serial0/1

  Redistribution:

    None
```

Actually, 4 different routing protocols are shown in the **show ipv6 protocols** command: our 2 dynamic processes, plus connected routes and static routes.

To perform two-way redistribution, we can distribute EIGRP into OSPF:

Our routing table on O2 now looks like this, with EIGRP's external routes bolded:

```
OE2  2101::/64 [110/20] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

OE2  2102::/64 [110/20] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

O 2200::/64 [110/74] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

O 2201::/64 [110/20] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

C 2202::/64 [0/0] via ::, FastEthernet0/0

L 2202::2/128 [0/0] via ::, FastEthernet0/0

OE2  3100::/64 [110/20] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

OE2  3101::/64 [110/20] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

OE2  3102::/64 [110/20] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

OE2  3103::/64 [110/20] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

O 3200::1/128 [110/10] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

O 3201::1/128 [110/10] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

O 3202::1/128 [110/10] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

O 3203::1/128 [110/10] via FE80::C00A:FFF:FE28:1, FastEthernet0/0

L FF00::/8 [0/0] via ::, Null0
```

The **include-connected** clause at the end of a redistribute command is used in IPv6 because, by default, a router performing redistribution won't automatically redistribute connected routes. To illustrate this, let us first disable both fast Ethernet interfaces on D2. Then, on E2, issue the **show ipv6 route** command. The network for 2200::/64 will have no entry. This is because we haven't told D1 to include-connected routes when it performs redistribution. The reason we had to disable D2 is to simplify this demonstration, so D2 didn't take over for D1's lack of advertising a 2200::/64 prefix.

To be sure that all our EIGRP routers have paths around our OSPF network, we should add the **include-connected** command to the end of our **redistribute** statements. This problem (and the solution) is specific to IPv6.

The same goes for OSPF routes we take from EIGRP. The 2100::/64 network will be missing from our OSPF routing tables because D1 won't advertise this connected route unless we redistribute with the **redistribute eigrp 1000 include-connected** OSPF subcommand.

We can also manipulate IPv6 routes the same way we would manipulate IPv4 routes. This includes distribute-lists and route-maps.

```
D1(config-route-map)#match ipv6 address ?

  WORD          IPv6 access-list name

  prefix-list  IPv6 prefix-list
```

Redistributing BGP

Knowledge of BGP redistribution is recommended for your design processes. That being said, redistributing BGP into an IGP is often a bad idea for three reasons:

- Extensive filtering should take place, as IGPs can't handle the number of routes BGP can.
- Sending a BGP route into an IGP causes the AS_PATH to be lost, if the route is re-sent into BGP, a BGP routing loop could occur. Remember that BGP checks the AS_PATH attribute to see if a route contains its own ASN. If an IGP erases that ASN, then we could end up creating a loop.
- IGPs do not understand BGP metrics. While BGP is highly tunable, IGPs don't have an ability to tweak all of the values.

We can inject routes into BGP the same way we inject them in to other protocols. If we inject an entire protocol, say all of our EIGRP process, into BGP, we may end up with problems. However, careful filtering should take place so we aren't advertising unnecessary routes to our BGP peers.

One way to do this is by creating a static route with destination Null0 that is a summary route of the prefixes we want to send to BGP. Another way is to aggregate consecutive addresses into one BGP summary route using the **aggregate-address** command inside a BGP process.

We can still apply distribute-list and route-map commands to filter and tweak our redistribution. Using route-map **set** commands, we can modify the BGP metrics as routes come in to our BGP process.

Consider the following topology, which is the same topology from the BGP chapter:

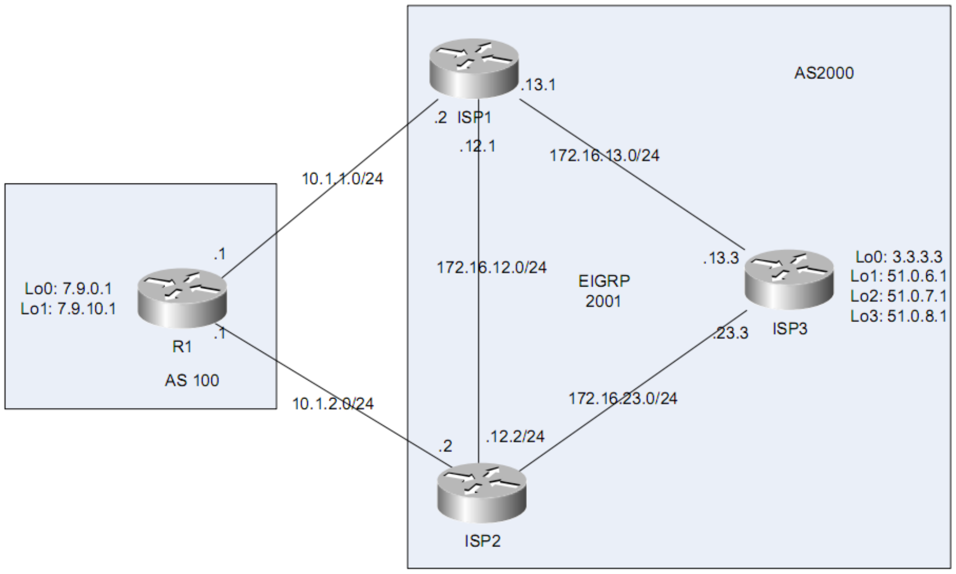

Figure 48: BGP Redistribution on ISP1 and ISP2

In our BGP chapter, we set up BGP to advertise each of the networks in this diagram, with EIGRP supporting iBGP in AS2000. In this chapter we will illustrate how routes can be redistributed between BGP and EIGRP on ISP1 and ISP2.

Assume that we removed the 51 networks in AS2000 from BGP by issuing the **no network 51.0.0.0** statement. Now our 51 networks are only via EIGRP. If we wanted to put them back into BGP by means of redistribution instead of the **network** command, we could issue the following command:

```
ISP1(config)#access-list 51 permit 51.0.6.0 0.0.0.255

ISP1(config)#access-list 51 permit 51.0.7.0 0.0.0.255

ISP1(config)#access-list 51 permit 51.0.8.0 0.0.0.255

ISP1(config)#router bgp 2000

ISP1(config-router)#distribute-list 51 in

ISP1(config-router)#redistribute eigrp 2000
```

We can verify this command worked and only brought in the routes we want by using **show ip bgp**:

```
ISP1#show ip bgp

BGP table version is 12, local router ID is 172.16.13.2

Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal, r RIB-failure, S Stale

Origin codes: i - IGP, e - EGP, ? - incomplete


   Network          Next Hop   Metric LocPrf Weight Path

* i7.9.0.0/24       10.1.2.1        0    100      0 100 i

*>                  10.1.1.1        0             0 100 i

* i7.9.10.0/24      10.1.2.1        0    100      0 100 i

*>                  10.1.1.1        0             0 100 i

r i10.1.1.0/24      10.1.2.1        0    100      0 100 i

r>                  10.1.1.1        0             0 100 i

* i10.1.2.0/24      10.1.2.1        0    100      0 100 i

*>                  10.1.1.1        0             0 100 i

*> 51.0.6.0/24      172.16.13.3     156160         32768 ?

*> 51.0.7.0/24      172.16.13.3     156160         32768 ?

*> 51.0.8.0/24      172.16.13.3     156160         32768 ?

*> 172.16.12.0/24   0.0.0.0         0             32768 ?

*> 172.16.13.0/24   0.0.0.0         0             32768 ?

*> 172.16.23.0/24   172.16.13.3   2884160         32768 ?
```

We don't need to worry that the next hop is given as "?" in the output because EIGRP has no concept of path, the path is simply blank. As stated, this could lead to BGP routing loops, as BGP uses the list of paths to determine if there is a loop. This isn't a problem in our topology, BGP simply receives a route with a next hop and a metric.

Domain 6: Path Control & Branch Offices

This final chapter discusses four items:

- Policy Based Routing (PBR)
- Offset lists
- IP Service-Level Agreements (IP SLA)
- Branch Office & Teleworker Services

Policy Based Routing

PBR is a feature found in Cisco routers that allows you to route based on more criteria than just the destination address. Typically, when a router identifies a next-hop and exit interface for data, it does so based on the data's destination address. The routing protocol in use will dictate which route makes it in to the routing table.

With PBR, we can pick routes based on information like the source IP address. PBR can be useful for a variety of situations:

- Routing voice traffic over a different link than data traffic to the same destination
- Keeping disparate networks separate, such as traffic from the faculty subnets separate from traffic from the student subnets
- Load balancing without the help of a dynamic routing protocol

If OSPF and EIGRP are running on the network in Figure 49, they will most likely send traffic from R1 to R5 through R3. Unless we have tweaked the *k* values or variance, or changed the OSPF costs, our fast Ethernet path will have a lower metric than our serial path, and it will be placed in the routing table of R2.

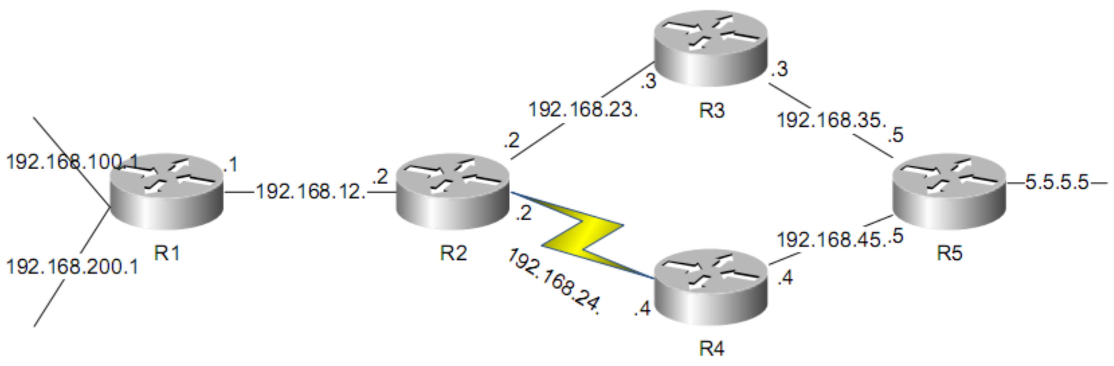

Figure 49: PBR Topology

The target address we are interested in reaching is 5.5.5.5. We have enabled OSPF on all interfaces and can verify this by looking at R2's routing table:

```
C 192.168.12.0/24 is directly connected, FastEthernet0/0

O 192.168.45.0/24 [110/30] via 192.168.23.3, 00:38:34, FastEthernet0/1

C 192.168.24.0/24 is directly connected, Serial0/0

 5.0.0.0/32 is subnetted, 1 subnets

O 5.5.5.5 [110/21] via 192.168.23.3, 00:38:34, FastEthernet0/1

C 192.168.23.0/24 is directly connected, FastEthernet0/1

O 192.168.35.0/24 [110/20] via 192.168.23.3, 00:38:34, FastEthernet0/1
```

OSPF put R3's address as the next-hop IP, and we can tell by the metric that we calculated the cost using fast Ethernet links and not serial links.

Say we want to route some specific traffic over our serial link through R4 to R5. In this case, we will route traffic from 192.168.100.0 /24 and 192.168.200.0 /24 over the serial link, while keeping the remaining traffic on the fast Ethernet network.

First we will create an access-list (or prefix-list, if that will better suit your needs) to match the traffic we want to manipulate. Since R2 is the router making the choice, we will do this on R2.

```
R2(config)#ip access-list standard 20

R2(config-std-nacl)#permit 192.168.100.0 0.0.0.255

R2(config-std-nacl)#permit 192.168.200.0 0.0.0.255
```

Then, we create a route-map that matches the IP access list and sets the next-hop value:

```
R2(config-std-nacl)#route-map Our_PBR permit 10

R2(config-route-map)#match ip address 20

R2(config-route-map)#set ip next-hop 192.168.24.4
```

Finally, we must apply the route-map as a policy on the incoming interface:

```
R2(config)#int fast 0/0

R2(config-if)#ip policy route-map Our_PBR
```

Verifying the PBR can be accomplished with a series of traceroutes. Because we only have 1 device attached to R2's incoming interface, we can set loopback addresses and extended traceroute commands to tell traceroute to set the source address to the loopback address:

```
R1#traceroute

Protocol [ip]:ip

Target IP address: 5.5.5.5

Source address: 192.168.100.1

Numeric display [n]:

Timeout in seconds [3]:

Probe count [3]:

Minimum Time to Live [1]:

Maximum Time to Live [30]:

Port Number [33434]:

Loose, Strict, Record, Timestamp, Verbose[none]:

Type escape sequence to abort.

Tracing the route to 5.5.5.5


  1 192.168.12.2 68 msec 64 msec 44 msec

  2 192.168.24.4 116 msec 72 msec 96 msec

  3 192.168.45.5 140 msec *  72 msec
```

Hop #2, in bold, is through R4. An extended traceroute sourced from 192.168.200.1 will show us the same thing. A regular traceroute shows that traffic not matching our policy is still going through R3:

```
R1#traceroute 5.5.5.5


Type escape sequence to abort.

Tracing the route to 5.5.5.5


  1 192.168.12.2 76 msec 64 msec 28 msec

  2 192.168.23.3 112 msec 60 msec 80 msec

  3 192.168.35.5 120 msec *  140 msec
```

On R2, we can verify that our route-map was applied as a policy to the correct interface:

```
R2#show ip policy

Interface       Route map

Fa0/0           Our_PBR
```

And we can see what the policy is doing with the **debug ip policy** command:

```
R2#debug ip policy

Policy routing debugging is on

*Mar  1 01:38:41.907: IP: s=192.168.12.1 (FastEthernet0/0), d=5.5.5.5,
len 28, FIB policy rejected(no match) - normal forwarding

*Mar  1 01:40:23.579: IP: s=192.168.100.1 (FastEthernet0/0), d=5.5.5.5,
g=192.168.24.4, len 100, FIB policy routed
```

PBR can be optimized for redundancy. Instead of giving the **set ip next-hop** command, we could have given one or more exit interfaces with **set interface**. The router uses the first interface in the list that is up/up. There are also two variations on setting the next hop and the outbound interface. **set ip default next-hop** tells the router to use an explicit route (non default) to this network if it has one. The same goes for **set default interface;** if a router has learned a route, it will be used, otherwise the traffic is policy-routed out the given interface or set of interfaces.

Offset Lists

An offset list can manipulate the way traffic is advertised to other routers or the way a router sees certain traffic. Paths can be controlled by purposefully advertising higher metrics to part of the networks, while leaving others alone. The offset list is only applicable to RIP and EIGRP.

Consider the following EIGRP network:

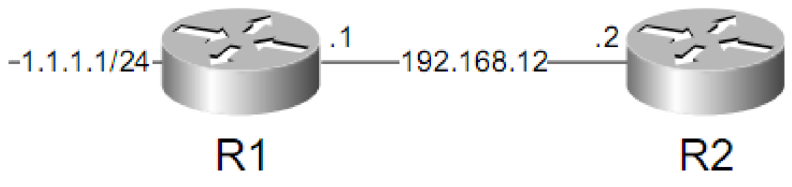

Figure 50: Offset-list example topology

In this example, **show ip route** shows us that R2 is originally calculating a metric of 409600 to the 1.1.1.0 /24 network.

```
D 1.1.1.0 [90/409600] via 192.168.12.1, 00:00:08, FastEthernet0/0
```

If we wanted to increase this path's metric, we can either increase the advertised distance out of R1, or increase the distance perceived by R2 using an offset list. The first example increases R2's perceived metric by 20000.

```
R2(config)#ip access-list standard 50

R2(config-std-nacl)#permit 1.1.1.0 0.0.0.255

R2(config-router)#offset-list 50 in 20000 fastEthernet 0/0
```

show ip route now gives us:

```
D 1.1.1.0 [90/429600] via 192.168.12.1, 00:00:08, FastEthernet0/0
```

The second example increases the metric that R1 advertises by 4000:

```
R1(config)#ip access-list standard 50

R1(config-std-nacl)#permit 1.1.1.0 0.0.0.255

R1(config)#router eigrp 100

R1(config-router)#offset-list 50 out 4000


D 1.1.1.0 [90/433600] via 192.168.12.1, 00:00:05, FastEthernet0/0
```

As you can see, an offset-list is composed of an access-list to match the traffic, a direction and optionally an interface name. On R2 we specified an interface, while on R1 we did not.

Changing the metric of a route allows us to control paths. By setting one route higher than another, we can make it less preferable for a router to send traffic through.

IP SLA

The IP SLA is composed of two objects: a sender and responder. The sender is the device that initiates a test, and the responder responds to the test. If the answer is within valid thresholds, the IP SLA agreement is met and the router does not take any action. In this section, we look at how to configure our router to ping our main ISP, and if our ISP fails, to switch traffic over to our other ISP.

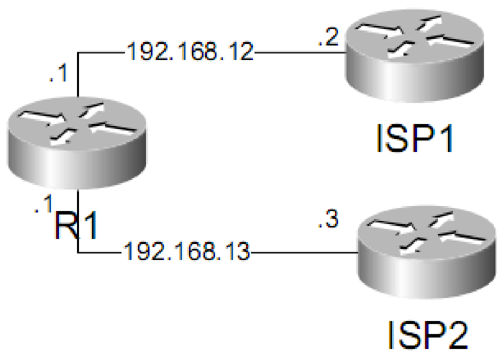

Figure 51: IP SLA Operation

```
R1(config)#ip sla 1000

R1(config-ip-sla)#icmp-echo 192.168.12.2

R1(config-ip-sla-echo)#timeout 500

R1(config-ip-sla-echo)#freq 3

R1(config)#ip sla schedule 1000 start-time now life forever

R1(config)#track 1 rtr 1000 reachability

R1(config)#ip route 0.0.0.0 0.0.0.0 192.168.12.2 track 1

R1(config)#ip route 0.0.0.0 0.0.0.0 192.168.13.3 2
```

This configuration has basically 3 components

- An IP SLA object called 1000 which defines criteria to meet
- A track object number 1 that keeps track of reachability
- Part of the routing table

R1 will ping (icmp-echo) ISP1 every 3 seconds and wait up to 500 ms for the reply ping. If a ping fails, track 1 will declare the SLA dead. When track 1 declares the SLA dead, it pulls the default route to 192.168.12.2 out of the routing table. An alternative default route via 192.168.13.3 was already known, but it wasn't in the routing table because the route via ISP1 had a better administrative distance. Since ISP1 is now gone, ISP2 can take over.

The SLA criteria do not have to be simply a ping, which is either returned or lost. You can measure jitter, response time, and delay on a variety of objects including UDP streams, FTP & HTTP file transfers, and even DNS query time. Monitoring the jitter of UDP streams is important for VoIP conversation quality, while monitoring round-trip time is good for validating the quality of our link.

Branch Office Routing

Profile of a Branch Office

Many branch offices today are built from a handful of technologies that paint a picture of a secure, reliable, and affordable solution that connects the branch to both the head office and the Internet. Branch office routers provide the services needed to both support the office and provide a connection to the rest of the enterprise.

The traditional design of a BO is using some form of leased line, which logically creates a private connection back to the headquarters (HQ), as shown in Figure 52.

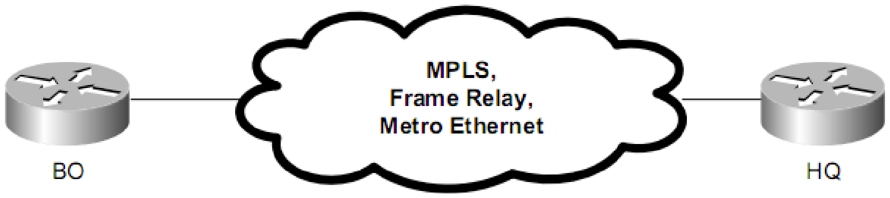

Figure 52: Private Branch Office Technologies

Using a private network for each branch office has a major advantage: since you own the network (logically speaking), you can run whatever routing protocols you want on the network. IGPs can, and often do, traverse MPLS (Multi-Protocol Label Switching), frame-relay, and metro Ethernet links. With these technologies, routers can appear to be just a hop away, although they are traversing a service provider network.

However, private networks also have a major disadvantage: cost. Because each link must be acquired on an individual and specific basis, this network design is often limiting to support branch and teleworkers.

A more feasible solution for many businesses is to route the traffic over an inexpensive connection to the public Internet (Figure 53). While this is cost-effective, it requires additional configuration for security and routing.

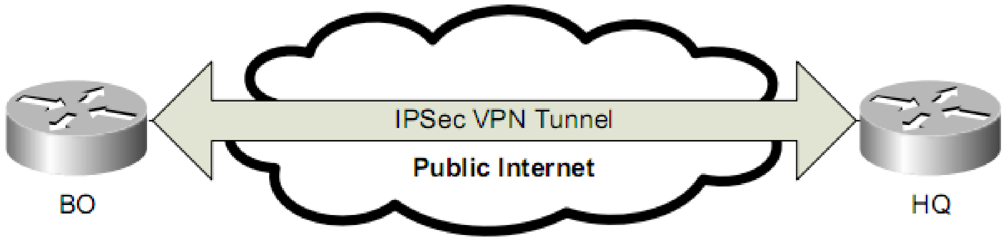

Figure 53: Sending Branch Office traffic through the public internet

To address the first concern, security, the routers on each end of the system form what is known as a Virtual Private Network, or more commonly, a VPN. A VPN makes the BO router appear like it has an interface in the HQ's network. The HQ office and BO office mutually authenticate each other, agree on a set of cryptography keys, and exchange encrypted data between them. With most routers, this is done using a protocol called IPsec. IPsec is discussed in the next section.

Secondly, because the data must traverse the public Internet, IGPs must be given special attention. Many IGPs don't work over the Internet because neighbors must be directly connected. To get around this, an IGP can be wrapped into a tunnel that rides on top of a VPN called Generic Route Encapsulation (GRE). GRE is discussed after IPsec.

IPsec

When a branch office or teleworker needs to exchange data with the headquarters, it can make use of the public Internet. However, the only effective way to do this is if we can be sure the data is from the correct source and not tampered with in transit. We also need to be sure that third parties can't read the data and make use of it. To meet these goals, Cisco routers use the IPsec protocol suite to create VPN connections.

As a protocol suite, an IPsec VPN is composed of many elements

Each IPsec VPN contains:

1. A Security Association (SA) set up by Internet Key Exchange (IKE)
2. Authentication Header (AH) which provides authentication of data origin and provides integrity of the transmission
3. Encapsulating Security Payload (ESP) – encrypted data

However, IPsec does not specify which encryption or authentication protocols to use. These can be selected by the engineer.

IPsec also runs in two modes, either tunnel mode or transport mode. In tunnel mode, the entire IP packet, including the header, is encrypted, authenticated and put into a new IP packet. In transport mode, only the application data of the IP packet is encrypted, the rest of the packet, including the header, is left unencrypted to make routing easier.

The following configuration creates a VPN between two routers (the adjacent router config not shown) using a pre-shared key (p@ssw0rd). Since IPsec is a suite of protocols, rather than just a single protocol, we must create a *transform* and to define what algorithms we want to use for security. The configuration below tells the IPsec transform to apply DES encryption, use SHA for authentication, and to apply these on the fast Ethernet 0/0 interface. Additionally, an ACL (in this case, all IP traffic) tells the router what data should be sent encrypted.

```
R1(config)#crypto isakmp policy 1

R1(config-isakmp)#authentication pre-share

R1(config-isakmp)#group 2


R1(config)#crypto isakmp key p@ssw0rd address 192.168.12.2

R1(config)#crypto IPsec transform-set our_transform esp-des esp-sha-hmac


R1(cfg-crypto-trans)#crypto map our_map 10 IPsec-isakmp

R1(config-crypto-map)#set peer 192.168.12.2

R1(config-crypto-map)#set transform-set our_transform

R1(config-crypto-map)#match address 150

R1(config)#access-list 150 permit ip any any

R1(config)#interface fast 0/0

R1(config-if)#crypto map our_map
```

Newer Cisco routers, such as most of the ISR series, support SSL VPNs. They use the same SSL protocol that is used for web sites. SSL VPNs have been gaining popularity because some implementations don't require a client to be installed on the client machine. These VPNs are also easier to route through firewalls than IPsec VPNs because they use a well-known port (443 by default).

GRE

Because multicasts can't be forwarded over public networks, a Generic Route Encapsulation (GRE) tunnel can assist you in running an IGP across a public network. GRE encapsulates matching traffic and sends it as a unicast across otherwise impassible routes. At the other end of the GRE tunnel, the packet is unencapsulated. The GRE tunnel is like a point-to-point link that transports dynamic routing information to a branch office. You can also configure GRE to work inside IPsec.
The following configuration creates a GRE interface on a router and it tells the GRE interface that the tunnel terminates at IP 192.168.12.2. This is the far-end IP address.

```
R1(config)#interface tunnel 0

R1(config-if)#tunnel source fastEthernet 0/0

R1(config-if)#tunnel destination 192.168.12.2
```

On R2 we would configure:

```
R2(config)#interface tunnel 0

R2(config-if)#tunnel source fastEthernet 0/0

R2(config-if)#tunnel destination 192.168.12.1
```

Assuming we have IP connectivity between R1 and R2, our GRE tunnel should work. A common practice when hard-coding IP addresses into a configuration file is to use the far-end loopback interface. This is used so that if the actual IP address of the router changes, our tunnel won't break, and we won't need to change our local config to compensate for changes in neighbors.

DSL & Cable Technologies

The term "broadband" became mainstream in the early 2000's when it became synonymous with high-speed Internet access (as opposed to dial-up), however, this is a bit of a misnomer. In a technical sense, broadband technologies use the bandwidth available in frequencies beyond that of human hearing.
As opposed to an analog modem, whose familiar tones can be heard when picking up the telephone when a connection is in progress, broadband technologies use the bandwidth in regions that humans typically can't hear. Since most telephone conversations only use the range up to about 4000 Hz, the ranges above this frequency can be used for the exchange of digital information. This is how DSL (Digital Subscriber Line) technology works. Not only is there much more bandwidth in inaudible ranges than in audible ranges, but higher frequency oscillations can switch data faster than the technology used in dial-up modems.

A cable modem works the same way, but instead of working alongside the voice bands, it works alongside the video bands. Cable TV does not use the entire range of frequencies transmittable over the coax, so the modem sends and receives data in the unused frequencies, leaving the television signal untouched.

The "last mile" copper wire connection between the Central Office (CO) and the home or office is called the Local Loop. Because the signal degenerates over distance between the CO and the customer's DSL termination point, several different varieties of DSL have been developed that vary in effective distance and throughput. Too much distance between the customer DSL termination point and the CO will result in frame errors and retries, thus limiting the effectiveness of DSL. The CO has a device known as the DSLAM, or Digital Subscriber Line Access Multiplexer. The DSLAM sends the IP traffic from the customer's DSL line to the ISP network, and the voice traffic to the Public Switched Telephone Network (PSTN).

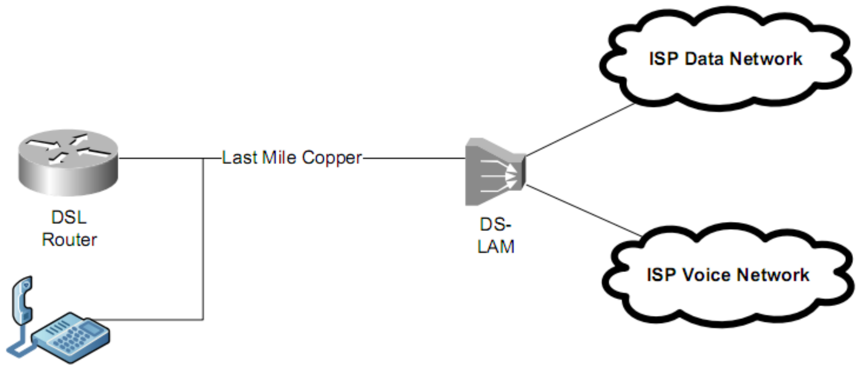

Figure 54: DSL Conceptual Layout

At layer 2, DSL borrows from two previously mentioned technologies from the CCNA curriculum: Asynchronous Transfer Mode (ATM) and Point-to-Point Protocol (PPP). ATM DSL uses a virtual circuit concept similar to frame-relay, and recall that ATM divides data into fixed-size packages known as cells. Each cell contains 48 bytes of data and 5 bytes of header information. PPP is used for authentication via CHAP. The entire package, at layer 2, is known as PPPoA. Like frame-relay, ATM uses the concept of the Permanent Virtual Circuit (PVC), which is configured on our ATM interface in the example below. The PVCs must match both on our client and at the ISP. It is likely the ISP will assign you a PVC number rather than negotiate one.

Cisco routers use the concept of the *Dialer* interface. The dialer is a logical interface which can be bound to a physical interface. In the code below, we configure the dialer (with CHAP authentication) and then configure the ATM interface. Finally, we add a default route to the upstream ISP by means of dialer 0.

```
Router(config)#interface dialer 0

Router(config-if)#ip address negotiated

Router(config-if)#encapsulation ppp

Router(config-if)#dialer pool 1

Router(config-if)#ppp chap password p@ssw0rd


Router(config)#interface atm 0

Router(config-if)#pvc

Router(config-if-atm-vc)#encap aal5mux ppp dialer

Router(config-if-atm-vc)#dialer pool-member 1

Router(config-if-atm-vc)#no shutdown

Router(config)#ip route 0.0.0.0 0.0.0.0 dialer 0
```

Address Helper Protocols

In this section, we describe a couple of protocols that are commonly used with enterprises, branch offices and teleworkers. These protocols aren't directly related to routing, but are helpful nonetheless. They are Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP). Although "Address Helper Protocols" is not an official name, these protocols help provide the assignment and manipulation of IP addresses both inside and outside an enterprise. Home and Small-Office routers often have these features automatically enabled, while enterprise devices such as routers and firewalls require manual configuration. If you recently took the CCNA exam, these topics may already be familiar to you.

NAT

Network Address Translation was developed as a way to "multiplex" many concurrent traffic flows to appear to start and/or terminate on a different IP address. It is often used as an address conservation method. An enterprise or office can use a private, non-routable addressing scheme inside the corporation and translate its inbound or outbound traffic to a drastically smaller set of public IP addresses.
Inside addresses are usually in the range of private addresses defined by RFC 1918.

The most common implementation of NAT is called Port Address Translation (PAT), also known as NAT Overload. With this method, each IP flow through the router is assigned an IP port number from 1024-65535, and the router re-writes the source or destination port dynamically. With this method, approximately 65000 concurrent IP sessions can use the same IP address. The router maintains a translation table of the inside address and port number and outside address and port number. When a new flow goes through the router that is not in the table, it creates a new entry and dynamically assigns unused ports. This combination of address and port number provides much more flexibility than just source and destination address alone. PAT also provides a security mechanism to hide the internal addressing structure from the outside world. This helps prevent IP address spoofing and reconnaissance. Additionally, NAT translations can be monitored and rate-limited to detect and slow the spread of viruses.

To demonstrate a NAT configuration, we will use two different routers. First, we will use a 3640 running IOS 12.2. In this example, our internal IP addressing scheme will be the 10.10.0.0 /22 network. We want to NAT all internal connections to share the outside interface's IP address.

```
Router(config)#interface fast 0/0

Router(config-if)#ip address 10.10.0.1 255.255.252.0

Router(config-if)#ip nat inside

Router(config-if)#interface fast 1/0

Router(config-if)#ip address 203.22.22.1 255.255.255.252

Router(config-if)#ip nat outside

Router(config)#ip nat inside source list our-ip-acl interface
fastethernet 1/0 overload

Router(config)#ip access-list standard our-ip-acl

Router(config-nacl)#permit 10.10.0.0 0.0.3.255
```

The command **ip nat inside source list our-ip-acl interface fastethernet 1/0 overload** has several parts to it. The first parameter is inside, indicating that we are applying this translation to our inside addresses.

The second parameter is **source list our-ip-acl**, which tells IOS that we want to NAT every source IP matching the ACL. The command **interface fastethernet 1/0** tells IOS the destination, or rather what IOS should change the packet source to before it is sent to the outside. We could have specified an IP here, but in this configuration we tell IOS to use whatever IP address we have on the outside interface. A configuration like this would be useful when using DHCP on that interface. Finally, **overload** tells IOS that this is going to be PAT and not strictly NAT.

IOS should begin NATting our traffic. On newer versions the configuration is the same but IOS sets up NAT a little differently. When configuring NAT, the following message is logged:

```
*Mar  3 03:34:56.075: %LINEPROTO-5-UPDOWN: Line protocol on Interface NVI0, changed state to up
```

IOS in versions 12.3T and later creates a NVI, or NAT Virtual Interface, to handle the translation process. The purpose of this is to avoid declaring each interface to be inside or outside. Instead, we can just use the generic term **ip nat enable**. Cisco did this as a work-around some issues in IOS when using redundant routers.

DHCP

DHCP is used by most people every day to dynamically assign addresses to hosts and other devices. Cisco routers can function as DHCP servers or clients. Client configurations are common in teleworker and branch office address assignment. An interface is configured as a DHCP client using the **ip address dhcp** interface subcommand. To configure a Cisco router as a DHCP server, use the following configuration:

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

Router(dhcp-config)#network 192.168.123.0 255.255.255.0

Router(dhcp-config)#default-router 192.168.123.1

Router(dhcp-config)#dns-server 192.168.123.2 8.8.8.8

Router(config)#ip dhcp excluded-address 192.168.123.1
```

The configuration tells the router to run a DHCP server process and give out addresses on the 192.168.123.0 /24 network, with the exception of 192.168.123.1 (which we assume is the IP of the router itself). DHCP has a variety of options and Cisco's server implementation can be used to specify many more items than just an IP address, default gateway, and DNS servers.

Practice Questions

Chapter 1

1. A network engineer is on a router that is running EIGRP in ASN 100. They configure the following command on interface fa0/0:ip hold-time eigrp 100 10After this command was entered, the EIGRP neighbor relationship fails on fa0/0. Why did this happen?
Choose the best answer.

❍ A. The EIGRP hello timers do not match a neighbor connected to fa0/0.

❍ B. The engineer must perform a shutdown and then a no shutdown to bring the EIGRP neighbor relationship back.

❍ C. The EIGRP hold timers do not match on a neighbor connected to fa0/0.

❍ D. You cannot modify EIGRP hold timers on broadcast networks such as Ethernet.

2. You are troubleshooting an EIGRP authentication problem that is preventing an EIGRP neighbor relationship from forming. You run the following debug command:debug eigrp packet You see the following debug output:

```
*Jan 21 16:50:18.749: EIGRP: pkt key id = 2,
authentication mismatch

*Jan 21 16:50:18.749: EIGRP: Serial0/0/1: ignored packet from
192.168.1.101, opc ode = 5 (invalid authentication)

*Jan 21 16:50:18.749: EIGRP: Dropping peer, invalid
authentication*Jan 21 16:50:18.749: EIGRP: Sending HELLO
on Serial0/0/1

*Jan 21 16:50:18.749: AS 100, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ
un/rely 0/0*Jan 21 16:50:18.753: %DUAL-5-NBRCHANGE: IP-EIGRP(0)
100: Neighbor 192.168.1.101 (Serial0/0/1) is down: Auth failure
```

Given the debug output, what is likely the problem? Choose the best answer.

❍ A. One neighbor is using MD5 authentication while the other is using clear-text authentication.

❍ B. The authentication key numbers do not match.

❍ C. The MD5 passwords do not match.

❍ D. Authentication is not configured on the remote router.

❍ E. Authentication is not configured on the local router.

3. If you have multiple EIGRP authentication key numbers configured on a router, which one will EIGRP choose to send to neighbors? Choose the best answer.

❍ A. The router will send the password of the lowest numbered key first. If the password does not match the neighbor's password, it will try the next lowest and continue this process until the passwords match or until the router has attempted every key.

❍ B. The router will send the password of the highest configured key number.

❍ C. The router will send the password of the lowest configured key number.

❍ D. The router will send the password of the highest numbered key first. If the password does not match the neighbor's password, it will try the next highest and continue this process until the passwords match or until the router has attempted every key.

4. When configuring EIGRP on new WAN technologies such as an MPLS VPN or Metro Ethernet, it is important that an engineer understand what layer of the OSI model these technologies function on to determine how EIGRP will form neighbor relationships. From the choices below, choose the proper OSI layer each WAN technology runs on. Choose two.

❍ A. MPLS VPN is a layer 2 technology

❍ B. MPLS VPN is a layer 3 technology

❍ C. MPLS VPN is a layer 4 technology

❍ D. Metro Ethernet is a layer 2 technology

❍ E. Metro Ethernet is a layer 3 technology

5. An engineer is trying to summarize the following networks using the "ip summary-address eigrp" command:10.8.88.0/2510.8.89.48/2910.8.64.96/27. Which network and subnet mask below would be the smallest EIGRP summary address to include all three subnets?
Choose the best answer.

❍ A. 10.8.0.0 255.255.192.0

❍ B. 10.8.64.0 255.255.128.0

❍ C. 10.8.64.0 255.255.192.0

❍ D. 10.8.64.0 255.255.224.0

Chapter 2

1. You are in charge of designing an OSPF network that uses Frame Relay WAN connections. The design calls for a single central site with 20 remote sites using a partial mesh topology with Frame Relay PVCs. One PVC is between the central site and each of the 20 remote site routers. All routers use point-to-point subinterfaces and one subnet per PVC. Given this information, which of the following is true? Choose the best answer.

   ❍ A. The remote site router has 20 fully adjacent neighborships on the WAN.

   ❍ B. The central site router has 20 fully adjacent neighborships on the WAN.

   ❍ C. The central site router has a neighborship with the Frame Relay switch.

   ❍ D. The remote site router has a neighborship with the Frame Relay switch.

2. A network engineer is reviewing the design of a fully functional OSPF network that only uses FastEthernet and GigabitEthernet connections. The engineer is trying to determine what routers will be responsible for flooding Type 2 LSAs on the network. Which of the following is true? Choose the best answer.

   ❍ A. Because the entire network is NBMA, type 2 LSAs will never be seen on the network.

   ❍ B. Because the entire network is a broadcast technology, type 2 LSAs will never be seen on the network.

   ❍ C. Type 2 LSA messages will be flooded by elected DR routers.

   ❍ D. Type 2 LSA messages will be flooded during the DR election process by all routers configured to run OSPF.

3. When a router analyzes the OSPF LSDB to calculate the best route to each subnet for internal routes, it does the calculations in a specific order. Which answer below puts the calculations in the correct order of operation? Choose the best answer.

❍ A.
1. Runs SPF to find all possible paths through the area's topology, from itself to each subnet.
2. Finds all subnets inside the area, based on the stub interfaces listed in the Type 1 LSAs and based on any Type 2 network LSAs.
3. Calculates the OSPF interface costs for all outgoing interfaces in each route, picking the lowest total cost route for each subnet as the best route.

❍ B.
1. Finds all subnets inside the area, based on the stub interfaces listed in the Type 1 LSAs and based on any Type 2 network LSAs.
2. Runs SPF to find all possible paths through the area's topology, from itself to each subnet.
3. Calculates the OSPF interface costs for all outgoing interfaces in each route, picking the lowest total cost route for each subnet as the best route.

❍ C.
1. Finds all subnets inside the area, based on the stub interfaces listed in the Type 1 LSAs and based on any Type 2 network LSAs.
2. Calculates the OSPF interface costs for all outgoing interfaces in each route, picking the lowest total cost route for each subnet as the best route.
3. Runs SPF to find all possible paths through the area's topology, from itself to each subnet.

❍ D.
1. Calculates the OSPF interface costs for all outgoing interfaces in each route, picking the lowest total cost route for each subnet as the best route.
2. Finds all subnets inside the area, based on the stub interfaces listed in the Type 1 LSAs and based on any Type 2 network LSAs.
3. Runs SPF to find all possible paths through the area's topology, from itself to each subnet.

4. What does the following OSPF router command do when performed on an ASBR? default-information originate Choose the best answer.

❍ A. It tells the router to flood a default route into OSPF if the router has a default route in its neighbors IP routing table.

❍ B. It tells the router to flood a default route into OSPF if the router has a default route in its local IP routing table.

❍ C. It tells the router to send Type 3 summary LSAs instead of Type 5 LSAs.

❍ D. It tells the router to send Type 5 summary LSAs instead of Type 3 LSAs.

5. You are reviewing a network implementation and notice that the design shows OSPF areas 1 and 2 directly connecting to area 0 using different ABRs. The design shows Type 3 LSA filtering into non-backbone areas as well as into the backbone. Why is this? Choose the best answer.

❍ A. This diagram is incorrect as the ABR will send Type 3 LSAs into backbone areas only.

❍ B. This diagram is incorrect as the ABR will send Type 3 LSAs into non-backbone areas only.

❍ C. ABR's create Type 3 LSAs and sends them to both non-backbone and backbone areas.

❍ D. This diagram is incorrect as the ABR will send Type 5 LSAs into backbone areas only.

Chapter 3

1. An engineer wants to redistribute OSPF into EIGRP. They log in to the boarder router that runs both OSPF and EIGRP and execute the following commands:

```
Router(config)#router eigrp 100
Router(config-router)#redistribute ospf 2
Router(config-router)#endRouter#
```

Unfortunately, this did not cause OSPF routes to redistribute into EIGRP. What is likely the problem? Choose two.

❍ A. The OSPF process ID is something other than 2.

❍ B. OSPF cannot be redistributed into EIGRP.

❍ C. OSPF must first be redistributed into static routes prior to being redistributed into EIGRP.

❍ D. EIGRP needs to be configured with default metric values before it will redistribute OSPF.

2. What is the main benefit for using different OSPF external route (E1 and E2) types? Choose the best answer.

❍ A. When multiple stub networks advertise the same subnet

❍ B. When multiple ASBRs advertise the same subnet

❍ C. When multiple stub networks advertise a different subnet

❍ D. When multiple ASBRs advertise a different subnet

3. The redistribute command has two mechanisms that allow filtering of routes. What are they? Choose two.

❍ A. route-filter

❍ B. access-list

❍ C. match

❍ D. route-map

4. To use route tags to prevent domain loop problems when performing redistribution you can implement which of the following? Choose two.

❍ A. Set the tag value to null for all routes.

❍ B. Set the tag value so that it matches the metric for each route. This will properly identify routes taken from domain A and advertised into domain B.

❍ C. Set the tag value that identifies routes taken from domain A and advertised into domain B.

❍ D. When redistributing from domain B to domain A, match the tag value and filter routes with that tag.

Chapter 4

1. BGP has two different classes of peer. What are they called? Choose two.

❍ A. BGP E1

❍ B. BGP E2

❍ C. eBGP

❍ D. iBGP

❍ E. BGP OA

❍ F. BGP IA

2. You are designing a network with a single connection to the Internet. What are two reasonable choices you should consider? Choose two.

❍ A. BGP learned default route

❍ B. BGP partial updates

❍ C. BGP full updates

❍ D. Static default route

3. What command configures a BGP neighbor at IP address 10.99.99.254? Choose the best answer.

❍ A. network 10.99.99.254 remote-as 1

❍ B. network 10.99.99.254 remote-as 1

❍ C. neighbor 10.99.99.254 update-source as 1

❍ D. neighbor 10.99.99.254 remote-as 1

Chapter 5

1. Given the following IPv6 address and mask, what is the correct way to write the network prefix?
   2000:1234:5556:2344:4321:3456:a123:1111/56
   Choose the best answer.

   ❍ A. 2000:1234:5556:2344::/56

   ❍ B. 2000:1234:5556:2344:0000:0000:0000:0000/56

   ❍ C. 2000:1234:5556:2300::/56

   ❍ D. 2000:1234:5556::/56

2. Which of the following commands correctly configures an IPv6 address on an interface and sets it to operate using RIPng with the name CCNP-lab? Choose the best answer.

   ❍ A.
   ```
   Router(Config)# interface FastEthernet0/0
   Router(Config-if)# ipv6 address 2011::1/64
   Router(Config-if)# ip ripng CCNP-lab enable
   ```

   ❍ B.
   ```
   Router(Config)# interface FastEthernet0/0
   Router(Config-if)# ipv6 address 2011::1/64
   Router(Config-if)# ipv6 rip CCNP-lab enable
   ```

   ❍ C.
   ```
   Router(Config-if)# interface FastEthernet0/0
   Router(Config-if)# ipv6 address 2011::1/64
   ```

   ❍ D.
   ```
   Router(Config-if)# interface FastEthernet0/0
   Router(Config-if)# ipv6 address 2011::1/64
   Router(Config-if)# ipv6 rip CCNP-lab
   ```

3. Why would a network engineer choose to configure IPv6 tunnels over natively configuring IPv6 in a dual-stack environment? Choose the best answer.

   ❍ A. In case all routers on the network need to be IPv6 aware

   ❍ B. If IPv6 traffic needs to be encrypted

   ❍ C. To segment IPv4 traffic from IPv6 traffic

   ❍ D. If the network requirements are to support small pockets of IPv6 hosts

Chapter 6

1. An engineer is reviewing a router configuration and finds the following commands:

```
R1(config)# ip route 0.0.0.0 0.0.0.0 10.1.1.1 3 track 1
R1(config)# ip route 0.0.0.0 0.0.0.0 172.16.1.1 2 track 2
```

As long as the routes are reachable according to the SLA configuration, which static route will be placed into the routing table? Choose the best answer.

❍ A. The route to 10.1.1.1 will be in the routing table because the AD is set to 3.

❍ B. The route to 10.1.1.1 will be in the routing table because it is the first route being tracked.

❍ C. The route to 172.16.1.1 will be in the routing table because the AD is set to 2.

❍ D. The route to 172.16.1.1 will be in the routing table because the it is the highest route being tracked.

2. There are four possible methods to set IP Precedence based on IP address and interface using policy-based routing. If there are multiple set statements in the rule, what priority does the router put them in? Choose the best answer.

❍ A.
1. set next-hop <ip address>
2. set interface <interface>
3. set ip default next-hop <ip address>
4. set default interface <interface>

❍ B.
1. set ip default next-hop <ip address>
2. set default interface <interface>
3. set next-hop ip <ip address>
4. set interface <interface>

❍ C.
1. set next-hop ip <ip address>
2. set ip default next-hop <ip address>
3. set interface <interface>
4. set default interface <interface>

❍ D.
1. set ip default next-hop <ip address>
2. set next-hop ip <ip address>
3. set interface <interface>
4. set default interface <interface>

3. A network engineer is reviewing an IP SLA track configuration as shown here:

```
R1(config)# track 1 ip sla 10 state
R1(config-track)# delay up 60 down 60
```

What does the "delay up 60 down 60" command specify? Choose the best answer.

❍ A. It is a timer for use by the dynamic routing protocol for hold-downs due to flapping interfaces.

❍ B. It is used to set the PBR trust level for the tracked object.

❍ C. It helps to regulate flapping of the tracking state by not placing the route into or out of the routing table until the timer has expired.

❍ D. It is a metric used to assist in load-balancing between SLA routes of the same network.

4. When verifying an IPsec VPN tunnel, which of the following show commands lists the crypto ACLs, peers and the interface where the crypto map is applied? Choose the best answer.

❍ A. show crypto isakmp

❍ B. show crypto isakmp sa

❍ C. show crypto map

❍ D. show crypto ipsec sa

5. How is it that a phone line can provide both legacy voice calls and data communication over the same analog line that is common in most homes and offices? Choose the best answer.

❍ A. Analog voice uses frequencies at 4000 Hz and below. DSL uses frequencies above 4000 Hz for data so it does not interfere with the analog voice communications.

❍ B. Analog voice uses frequencies at 4000 Hz and above. DSL uses frequencies below 4000 Hz for data so it does not interfere with the analog voice communications.

❍ C. The voice calls are digitized at the CO DSLAM and then turned back into analog form on the DSL router. That allows voice and data to be transported together.

❍ D. Data is converted into analog and multiplexed with analog voice communication. The data is then put back into a digital format when it reaches the DSL router.

Answers & Explanations

Chapter 1

1. Answer: C

Explanation A. Incorrect. The EIGRP hello timer was not modified from the default so these timers properly match.

Explanation B. Incorrect. The reason the neighbor relationship failed is because the EIGRP hold timers do not match. Once you change the neighbor router to have hold timer of 10 seconds, the neighbor relationship will automatically be reestablished.

Explanation C. Correct. The EIGRP hold timers must match for routers to become neighbors. By default, the hold timer is three times the hello interval, 15 seconds and 180 seconds (NBMA networks).

Explanation D. Incorrect. Broadcast networks such as Ethernet have a default hold timer of 15 seconds. This timer can be modified, but must match between neighbors to form a relationship.

2. Answer: B

Explanation A. Incorrect. EIGRP only supports MD5 authentication.

Explanation B. Correct. The log messages indicate that the authentication key numbers between members do not match.

Explanation C. Incorrect. If the MD5 passwords did not match, you would see different log messages.

Explanation D. Incorrect. If authentication was not configured on the remote router, you would see different log messages.

Explanation E. Incorrect. If authentication was not configured on the local router, you would see different log messages.

3. Answer: C

Explanation A. Incorrect. The router will only send the EIGRP password of the lowest key number configured on the router.

Explanation B. Incorrect. The router will only send the EIGRP password of the lowest key number configured on the router.

Explanation C. Correct. The router will only send the EIGRP password of the lowest key number configured on the router.

Explanation D. Incorrect. The router will only send the EIGRP password of the lowest key number configured on the router.

4. Answers: B, D

Explanation A. Incorrect. MPLS VPN operates at layer 3.

Explanation B. Correct. MPLS VPN operates at layer 3.

Explanation C. Incorrect. MPLS VPN operates at layer 3.

Explanation D. Correct. Metro Ethernet operates at layer 2.

Explanation E. Incorrect. Metro Ethernet operates at layer 2.

5. Answer: D

Explanation A. Incorrect. This is not a valid summary statement for the three networks that need to be summarized.

Explanation B. Incorrect. This is not the smallest network summary statement for the three given subnets.

Explanation C. Incorrect. This is not the smallest network summary statement for the three given subnets.

Explanation D. Correct. This statement is the correct and most specific statement possible.

Chapter 2

1. Answer: B

Explanation A. Incorrect. The remote site router will only be adjacent to the central site router.

Explanation B. Correct. The central site router will become fully adjacent with each of the 20 remote site routers.

Explanation C. Incorrect. Frame Relay is a layer 2 technology and therefore never participates in any type of IP routing.

Explanation D. Incorrect. Frame Relay is a layer 2 technology and therefore never participates in any type of IP routing.

2. Answer: C

Explanation A. Incorrect. Ethernet is a broadcast network technology. Type 2 LSAs are used by DR routers after the DR election has occurred.

Explanation B. Incorrect. Because Ethernet is a broadcast technology, Type 2 LSAs are used by DR routers after the DR election has occurred.

Explanation C. Correct. Because Ethernet is a broadcast technology, Type 2 LSAs are used by DR routers after the DR election has occurred.

Explanation D. Incorrect. Because Ethernet is a broadcast technology, Type 2 LSAs are used by DR routers after the DR election has occurred.

3. Answer: B

Explanation A. Incorrect. This is not the proper OSPF calculation order for choosing internal OSPF routes.

Explanation B. Correct. This is the correct OSPF calculation for choosing internal OSPF routes.

Explanation C. Incorrect. This is not the proper OSPF calculation order for choosing internal OSPF routes.

Explanation D. Incorrect. This is not the proper OSPF calculation order for choosing internal OSPF routes.

4. Answer: B

Explanation A. Incorrect. The default route is flooded from the ASBR to other OSPF routers if the default route is also currently in the ASBR's local routing table.

Explanation B. Correct. The default route is flooded from the ASBR to other OSPF routers only if the default route is also in the ASBR's local routing table.

Explanation C. Incorrect. The default route is flooded from the ASBR to other OSPF routers if the default route is also currently in the ASBR's local routing table.

Explanation D. Incorrect. The default route is flooded from the ASBR to other OSPF routers if the default route is also currently in the ASBR's local routing table.

5. Answer: B

Explanation A. Incorrect. The ABR will send Type 3 updates into non-backbone areas only.

Explanation B. Correct. The ABR will send Type 3 updates into non-backbone areas only.

Explanation C. Incorrect. The diagram is inaccurate as the ABR will send Type 3 updates into non-backbone areas only.

Explanation D. Incorrect. The ABR will send Type 3 updates into non-backbone areas only.

Chapter 3

1. Answers: A, D

Explanation A. Correct. This is a valid statement.

Explanation B. Incorrect. This is not a valid statement.

Explanation C. Incorrect. This is not a valid statement.

Explanation D. Correct. You must configure default metric values in EIGRP to redistribute any routing protocol or static/connected networks. The only exception is when you want to redistribute EIGRP into EIGRP that uses different AS numbers.

2. Answer: B

Explanation A. Incorrect. The main benefit is found when multiple ASBRs advertise the same subnet.

Explanation B. Correct. The benefits of using different external routes types are found when multiple ASBRs advertise the same subnet. The different route types let the router automatically set different metrics to the same subnet which forces the router to choose the E1 routes over the E2 routes.

Explanation C. Incorrect. The main benefit is found when multiple ASBRs advertise the same subnet.

Explanation D. Incorrect. The main benefit is found when multiple ASBRs advertise the same subnet.

3. Answers: C, D

Explanation A. Incorrect. This is not a way to filter routes.

Explanation B. Incorrect. This is not a way to filter routes.

Explanation C. Correct. The match command is one way to filter routes.

Explanation D. Correct. The route-map command is one way to filter routes.

4. Answers: C, D

Explanation A. Incorrect. By default, the tag value is set to null.

Explanation B. Incorrect. This is not a valid technique.

Explanation C. Correct. This is a valid technique for using tags to prevent domain loop problems.

Explanation D. Correct. This is a valid technique for using tags to prevent domain loop problems.

Chapter 4

1. Answers: C, D

Explanation A. Incorrect. This is not the correct name for one of the two classes of BGP peers.

Explanation B. Incorrect. This is not the correct name for one of the two classes of BGP peers.

Explanation C. Correct. The two classes are external BGP (eBGP) and internal BGP (iBGP).

Explanation D. Correct. The two classes are external BGP (eBGP) and internal BGP (iBGP).

Explanation E. Incorrect. This is not the correct name for one of the two classes of BGP peers.

Explanation F. Incorrect. This is not the correct name for one of the two classes of BGP peers.

2. Answers: A, D

Explanation A. Correct. Because there is only one Internet connection, there is no reason to pull in BGP partial or full routes since they would all point to the same place.

Explanation B. Incorrect. Because there is only one Internet connection, there is no reason to pull in BGP partial or full routes since they would all point to the same place.

Explanation C. Incorrect. Because there is only one Internet connection, there is no reason to pull in BGP partial or full routes since they would all point to the same place.

Explanation D. Correct. Because there is only one Internet connection, there is no reason to pull in BGP partial or full routes since they would all point to the same place.

3. Answer: D

Explanation A. Incorrect. The correct answer is neighbor 10.99.99.254 remote-as 1.

Explanation B. Incorrect. The correct answer is neighbor 10.99.99.254 remote-as 1.

Explanation C. Incorrect. The correct answer is neighbor 10.99.99.254 remote-as 1.

Explanation D. Correct. This command configures a neighbor at IP 10.99.99.254 on AS 1.

Chapter 5

1. Answer: C

Explanation A. Incorrect. The correct prefix for a /56 is 2000:1234:5556:2300::/56.

Explanation B. Incorrect. The correct prefix for a /56 is 2000:1234:5556:2300::/56.

Explanation C. Correct. This is the correct prefix in which the address in question resides.

Explanation D. Incorrect. The correct prefix for a /56 is 2000:1234:5556:2300::/56.

2. Answer: B

Explanation A. Incorrect. The correct way to enable the interface to run RIPng would be ipv6 rip CCNP-lab enable.

Explanation B. Correct. This is the correct way to configure RIPng on an interface.

Explanation C. Incorrect. The correct way to enable the interface to run RIPng would be ipv6 rip CCNP-lab enable.

Explanation D. Incorrect. The correct way to enable the interface to run RIPng would be ipv6 rip CCNP-lab enable.

3. Answer: D

Explanation A. Incorrect. This is not a reason to configure IPv6 tunneling instead of configuring IPv6 natively.

Explanation B. Incorrect. This is not a reason to configure IPv6 tunneling instead of configuring IPv6 natively.

Explanation C. Incorrect. This is not a reason to configure IPv6 tunneling instead of configuring IPv6 natively.

Explanation D. Correct. If you only need to support pockets of IPv6 hosts, you can tunnel the IPv6 traffic over IPv4 through the use of tunnels.

Chapter 6

1. Answer: C

Explanation A. Incorrect. As long as both routes are reachable, the route to 172.16.1.1 will be used because it has a lower AD (2 compared to 3).

Explanation B. Incorrect. As long as both routes are reachable, the route to 172.16.1.1 will be used because it has a lower AD (2 compared to 3).

Explanation C. Correct. As long as both routes are reachable, the route to 172.16.1.1 will be used because it has a lower AD (2 compared to 3).

Explanation D. Incorrect. As long as both routes are reachable, the route to 172.16.1.1 will be used because it has a lower AD (2 compared to 3).

2. Answer: A

Explanation A. Correct. This is the order the router will use if there are multiple set statements.

Explanation B. Incorrect. This is not the correct order of precedence.

Explanation C. Incorrect. This is not the correct order of precedence.

Explanation D. Incorrect. This is not the correct order of precedence.

3. Answer: C

Explanation A. Incorrect. The time regulates flapping of the tracking state. It has noting to do with dynamic routing protocols.

Explanation B. Incorrect. It is used to regulate flapping of the tracking state.

Explanation C. Correct. It is used to regulate flapping of the tracking state.

Explanation D. Incorrect. It is used to regulate flapping of the tracking state.

4. Answer: C

Explanation A. Incorrect. The proper command is show crypto map.

Explanation B. Incorrect. The proper command is show crypto map.

Explanation C. Correct. This command is useful when viewing the ACLs associated to the VPN tunnel and where it is applied.

Explanation D. Incorrect. The proper command is show crypto map.

5. Answer: A

Explanation A. Correct. Analog voice calls are designed to collect sounds that range from 300 to 3300 Hz. DSL operates above 4000 Hz and therefore does not interfere with voice communications.

Explanation B. Incorrect. Analog voice calls are designed to collect sounds that range from 300 to 3300 Hz. DSL operates above 4000 Hz and therefore does not interfere with voice communications.

Explanation C. Incorrect. Analog voice calls are designed to collect sounds that range from 300 to 3300 Hz. DSL operates above 4000 Hz and therefore does not interfere with voice communications.

Explanation D. Incorrect. Analog voice calls are designed to collect sounds that range from 300 to 3300 Hz. DSL operates above 4000 Hz and therefore does not interfere with voice communications.