CCNA™
CISCO® CERTIFIED NETWORK ASSOCIATE

9

Routing Introduction

CERTIFICATION OBJECTIVES

The last two chapters focused on products and protocols that function at layer-2. This chapter moves up one layer in the OSI Reference Model to discuss layer-3, the network layer. Layer-3 devices are generically called *routers*. Routers basically have two functions:

1. To find a layer-3 path to a destination network
2. To move packets from one interface to another to get a packet to its destination

In order to accomplish the first function, a router will need to:

- Learn about routers it is connected to in order to learn the networks that they know about
- Find locations of destination network numbers
- Choose a *best* path to each destination
- Maintain the most up-to-date routing information about how to reach destination networks

In order to accomplish its second function, the router will need to examine the destination IP address in an incoming IP packet, determine the network number of the destination, look in its routing table, and switch the packet to an outgoing interface. As you will see in this chapter, the routing table contains a list of destination network numbers, the status of these networks, which interface the router should use to reach the destination, and which neighboring router the router should use if the destination is more than one hop away.

This chapter covers an overview of routing, including how to set up static routes and how dynamic routing protocols—distance vector, link state, and hybrid protocols—function. Chapter 10 covers the configuration of two distance vector routing protocols, and Chapter 11 covers the configuration of a link state routing protocol and a hybrid routing protocol.

Types of Routes

A router can learn a route via one of two methods: *static* and *dynamic*. A static route is a route that is manually configured on the router. There are actually two ways that a router can learn a static route. First, a router will look at its active interfaces, examine the addresses configured on the interfaces and determine the corresponding network numbers, and populate the routing table with this information. This is commonly called a *connected* route. The second way that a router can learn a static route is for you to manually configure it.

exam Watch

Remember the difference between a routed protocol and a routing protocol.

Dynamic routes are routes that a router learns by running a routing protocol. Routing protocols will learn about routes from other neighboring routers running the same routing protocol. Dynamic routing protocols share network numbers a router knows about and reachability information concerning these networks. Through this sharing process, eventually a router will learn about all of the reachable network numbers in the network. There is a difference between the terms *routed* protocol and *routing* protocol. A routing protocol learns about routes for a routed protocol. A routed protocol is a layer-3 protocol, like IP or IPX. A routed protocol carries user traffic such as e-mail, file transfers, and web downloads. Table 9-1 shows some common routed protocols and the routing protocols that they use.

This book only focuses on routing for IP traffic and covers the basics of the RIP, IGRP, OSPF, and EIGRP routing protocols.

Autonomous Systems

Some routing protocols understand the concept of an autonomous system, and some do not. An *autonomous system (AS)* is a group of networks under a single administrative control, which could be your company, a division within your company, or a group of companies. An *Interior Gateway Protocol (IGP)* refers to a routing protocol that handles routing within a single autonomous system. IGPs include RIP, IGRP, EIGRP, OSPF, and IS-IS. An *Exterior Gateway Protocol (EGP)* handles routing between different autonomous systems. Today, there is only one active EGP: the Border Gateway Protocol (BGP). BGP is used to route traffic across the Internet backbone between different autonomous systems.

Not every routing protocol understands the concept of an AS. An AS can provide distinct boundaries for a routing protocol, and thus provides some advantages. For instance, you can control how far a network can be propagated by routers. Plus, you can control what routes you will advertise to other autonomous systems and what routes you'll accept from these systems.

TABLE 9-1 Routed and Routing Protocols

Routed Protocols	Routing Protocols
IP	RIP, IGRP, OSPF, EIGRP, BGP, IS-IS
IPX	RIP, NLSP, EIGRP
AppleTalk	RMTP, AURP, EIGRP

exam Watch

An autonomous system (AS) is a group of networks under a single administrative control. Each AS is assigned a unique number in order to differentiate it from other autonomous systems.

To distinguish one autonomous system from another, an AS can be assigned a unique number from 1 to 65,535. The Internet Assigned Numbers Authority (IANA) is responsible for assigning these numbers. Just like the public and private IP addresses defined in RFC 1918, there are public and private AS numbers. If you will be connected to the Internet backbone, are running BGP, and want to accept BGP routes from the Internet, you will need a public AS number. However, if you only need to break up your internal network into different systems, you only need to use the private numbers. Routing protocols that understand the concept of an AS are IGRP, EIGRP, OSPF, IS-IS, and BGP. RIP doesn't understand autonomous systems, while OSPF does; but OSPF doesn't require you to configure the AS number, whereas other protocols, such as IGRP and EIGRP, do. Cisco's BSCI exam spends a lot of time discussing autonomous systems and routing between them. The CCNA exam focuses only on the basics of IGPs.

Administrative Distance

One of the items mentioned in the chapter introduction is that each router needs to choose a *best* path to a destination. This can become somewhat complicated if the router is receiving routing update information for a single network from multiple sources, such as connected, static, and IGP routing protocols, and must choose *one* of these sources as the best and place it in the router's routing table. As you will see in this section and the section "Dynamic Routing Protocol," there are two things a router looks at when choosing a *best* path.

The first thing a router looks at is the administrative distance for a route source. Administrative distance is a Cisco-proprietary mechanism used to rank the IP routing protocols. As an example, if a router were running two IGPs, RIP and IGRP, and were learning network 10.0.0.0/8 from both of these routing protocols, which one should the router pick and place in its routing table? Which one should the router *believe* more? Actually, the term *administrative distance* is somewhat misleading, since the term has nothing to do with measuring distance. The term *believability* better describes the process.

Administrative distance ranks the IP routing protocols, assigning a value, or weight, to each protocol. Distances can range from 0 to 255. A smaller distance is more believable by a router, with the best distance being 0 and the worst, 255. Table 9-2 displays some of the default administrative distances Cisco has assigned to its routing protocols:

TABLE 9-2 Administrative Distance Values

Administrative Distance	Route Type
0	Connected interface
0 or 1	Static route
90	Internal EIGRP route (within the same AS)
100	IGRP route
110	OSPF route
120	RIP route
170	External EIGRP (from another AS)
255	Unknown route (is considered an invalid route and will not be used)

Going back to our previous example of a router learning network 10.0.0.0/8 from RIP and IGRP, since RIP has a value of 120 and IGRP, 100, the router will use the IGRP route, since this protocol has as a better (lower) administrative distance value.

Static Routes

A *static route* is a manually configured route on your router. Static routes are typically used in smaller networks. With a network that has hundreds of routes, static routes are not scalable, since you would have to configure each route, and any redundant paths for that route, on each router. This section covers the configuration of static routes and some of the issues associated with them.

Static Route Configuration

To configure a static route for IP, use one of these two commands:

```
Router(config)# ip route destination_network_# [subnet_mask]
                  IP_address_of_next_hop_neighbor
[administrative_distance] [permanent]
   -or-
Router(config)# ip route destination_network_# [subnet_mask]
                  interface_to_exit
[administrative_distance] [permanent]
```

The first parameter that you must specify is the destination network number. If you omit the subnet mask for the network number, it defaults to the Class A (255.0.0.0),

B (255.255.0.0), or C (255.255.255.0) default subnet mask, depending on the network number of the destination.

After the subnet mask parameter, you have two ways to specify how to reach the destination network: you can tell the router either the next hop neighbor's IP address or the interface the router should exit to reach the destination network. You should use the former method if the link is a multiaccess link (the link has more than two devices on it, three routers, for instance). You can use the latter method if it is a point-to-point link. In this instance, you must specify the *name* of the interface on the router, like **`serial0`**.

Optionally, you can change the administrative distance of a static route. If you omit this value, it will have one of two defaults, depending on the configuration of the previous parameter. If you specified the next hop neighbor's IP address, then the administrative distance defaults to 1. If you specified the interface on the router it should use to reach the destination, the router treats the route as a connected route and assigns an administrative distance of 0 to it. Please note that you can create multiple static routes to the *same* destination. For instance, you might have primary and backup paths to the destination. For the primary path, use the default administrative distance value. For the backup path, use a number higher than this, such as 2. Once you have configured a backup path, the router will use the primary path, and if the interface on the router fails for the primary path, the router will use the backup route.

exam Watch

Know the syntax for creating a static IP route.

The **`permanent`** parameter will keep the static route in the routing table even when the interface the router uses for the static route fails. If you omit this parameter, and the interface fails that the static route uses, the router will remove this route from its routing table and attempt to find an alternative path to place in the routing table. You might want to use this parameter if you never want packets to use another path to a destination, perhaps because of security reasons.

Default Route Configuration

A *default route* is a special type of static route. Where a static route specifies a path a router should use to reach a *specific* destination, a default route specifies a path the router should use if it *doesn't* know how to reach the destination.

Note that if a router does not have any path in its routing table telling it how to reach a destination, and the router receives a packet destined for this network, the router will

drop the packet. This is different from a switch, which will flood unknown destinations. Therefore, a default route can serve as a *catch-all*: if there is no specific path to the destination, the router will use the default route to reach it.

To set up a default route, use the following syntax for a static route:

```
Router(config)# ip route 0.0.0.0 0.0.0.0
                  IP_address_of_next_hop_neighbor
[administrative_distance] [permanent]
   -or-
Router(config)# ip route 0.0.0.0 0.0.0.0
                  interface_to_exit
[administrative_distance] [permanent]
```

exam Watch

A default route has a network number of 0.0.0.0 and a subnet mask of 0.0.0.0.

The network number of 0.0.0.0/0 at first appears a bit strange. Recall from Chapter 3, however, that network 0.0.0.0 represents all networks, and a mask of all 0's in the bit position represents all hosts in the specified network.

Default Routes and Distance Vector Protocols

A default route sometimes causes problems for certain routing protocols. There are two additional categories that a routing protocol can fall under: classful and classless. Examples of classful protocols include RIPv1 and IGRP. Examples of classless protocols include RIPv2, OSPF, EIGRP, IS-IS, and BGP.

A *classful* routing protocol understands only class subnets. For instance, if you have 192.168.1.0/23 in a routing update, a classful routing protocol wouldn't understand it, since a Class C network requires 24 bits of network numbers. This creates a problem with a default route, which has a /0 mask.

Also, when a classful router advertises a route out its interface, it does not include the subnet mask. For example, you might have 192.168.1.1/26 configured on your router's interface, and the router receives a routing updated with 192.168.1.0. With a classful routing protocol, the router will comprehend subnet masks only for network numbers configured on its interfaces. In this example, the router assumes that for 192.168.1.0, the only valid mask is /26. Therefore, if the routers sees the 192.168.1.0/26 as the network number, but the network is really 192.168.1.027, this can create a lot of routing confusion.

Classless protocols, on the other hand, do not have any issues accepting routing updates with any bit value for a subnet mask. However, for classful protocols, you

must configure the following command to accept nonconforming subnet masks, such as a default route:

```
Router(config)# ip classless
```

This command is also used to deal with *discontiguous* subnets in a network that is using a classful protocol: subnets separated by a different class network. For example, let's assume that you have networks 172.16.1.0/24, 172.16.2.0/24, and 172.16.3.0/24. However, a different class network, 192.168.1.0/24, sits between the first two Class B subnets and 172.16.3.0/24. In this situation, the router connected to 172.16.1.0/24 and 172.16.2.0/24, when it receives 172.16.0.0 from the side of the network connected to the discontiguous subnet, will *ignore* this routing entry.

Remember that when routes cross a class boundary in a classful protocol, the network number is sent as its classful number. Therefore, the router connected to 192.168.1.0/24 and 172.16.3.0/24, when it advertises updates across the 192.168.1.0/24 subnet, will advertise 172.16.0.0—not the actual subnet number. Since the router connected to 172.16.1.0/24 and 172.16.2.0/24 ignores the 172.16.0.0 routing information, it will not be able to reach 172.16.3.0. On top of this problem, even if you have a default route configured, since the router is connected to the 172.16.0.0 subnets, it assumes that 172.16.3.0 must also be connected; and if it isn't in the routing table, then the route cannot be reached.

By using the **ip** `classless` command, you are overriding this behavior; you're allowing your classful router to use a default route to reach discontiguous subnets. Not that this is a recommended design practice, but it does allow you to solve reachability problems for discontiguous subnets.

exam Watch

Classful protocols, such as IP RIPv1 and IGRP, understand only class subnets—you can apply only one subnet mask to a class address. Classless protocols, such as RIPv2, EIGRP, OSPF, and IS-IS, do not have this restriction.

Verifying Static Route Configuration

To verify the configuration of static and default routes on your router, use the **show** `ip` **route** command:

```
Router# show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP,
```

```
        M - mobile, B - BGP, D - EIGRP, EX - EIGRP external,
        O - OSPF, IA - OSPF inter area, N1 - OSPF NSSA
        external type 1, N2 - OSPF NSSA external type 2,
        E1 - OSPF external type 1, E2 - OSPF external type 2,
        E - EGP, i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2,
        * - candidate default, U - per-user static route, o - ODR,
        T - traffic engineered route

Gateway of last resort is not set
     172.16.0.0/24 is subnetted, 3 subnets
C       172.16.1.0 is directly connected, Ethernet0
C       172.16.2.0 is directly connected, Serial0
S       172.16.3.0 is directly connected, Serial0
```

exam Watch

Be familiar with the output of the `show ip route` command.

The top portion of the display for this command has a table of codes. These codes, which describe a type of route that may appear in the routing table, are shown in the first column at the bottom part of the display. In this example, there are two connected routes, and one static route— the static route is treated as a directly connected route, since it was created by specifying the interface to exit the router. This command is discussed in depth in Chapters 10 and 11.

9.01. The CD contains a multimedia demonstration of setting up static routes on a router.

EXERCISE 9-1

Static Route Configuration

These last few sections dealt with static routes and their configuration. This exercise will help you reinforce this material for the configuration of static routes. You'll perform this lab using Boson's NetSim™ simulator. This exercise has you set static routes on the two routers (2600 and 2500). You can find a picture of the network diagram for Boson's NetSim™ simulator in the Introduction of this book. After starting up the simulator, click on the *LabNavigator* button. Next, double-click on *Exercise 9-1* and click on the *Load Lab* button. This will load the lab configuration based on Chapter 5's and 7's exercises.

1. On the 2600, verify that the `fa0/0` and `s0` interfaces are up. If not, bring them up. Examine the IP addresses configured on the 2600 and look at its routing table.

At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600*. On the 2600, Use the **show** `interfaces` command to verify your configuration. If `fa0/0` and `s0` are not up, go into the interfaces (`fa0/0` and `s0`) and enable them: **no** `shutdown`. Use the **show** `interfaces` command to verify that the IP addresses you configured in Chapter 5 are still there. Use the **show** `ip route` command. You should have two connected networks: 192.168.1.0 connected to `fa0/0` and 192.168.2.0 connected to `s0`.

2. On the 2500, verify that the `e0` and `s0` interfaces are up. If not, bring them up. Examine the IP addresses configured on the 2500 and look at its routing table.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2500*. On the 2500, Use the **show** `interfaces` command to verify your configuration. If `e0` and `s0` are not up, go into the interfaces (`e0` and `s0`) and enable them: **no** `shutdown`. Use the **show interfaces** command to verify your configuration. Also use the **show** `interfaces` command to verify that the IP addresses you configured on Chapter 5 are still there. Use the **show** `ip` **route** command. You should have two connected networks: 192.168.3.0 connected to `e0` and 192.168.2.0 connected to `s0`.

3. Test connectivity between Host1 and the 2600. Test connectivity between Host3 and the 2500. Test connectivity between Host3 and Host1.

   At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host1*. From Host1, ping the 2600: **ping** `192.168.1.1`. The ping should be successful. If it is not, then you may have used the configuration from the VLAN lab in Chapter 8 and have a VLAN configuration problem. At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host3*. From Host3, ping the 2500 router: **ping** `192.168.3.1`. The ping should be successful. Also from Host3, ping Host1: **ping** `192.168.1.10`. The ping should fail: there is no route from the 2500 to this destination. Look at the 2500's routing table: it doesn't list 192.168.1.0/24: **show** `ip route`.

4. On the 2500, configure a static route to 192.168.1.0/24, which is connected to the 2600. View the routing table.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2500*. Configure the static route: **ip** `route 192.168.1.0 255.255.255.0 192.168.2.1`. View the static route: **show** `ip route`. Make sure that 192.168.1.0/24 shows up in the routing table as a static route (S).

5. On the 2600, configure a static route to 192.168.3.0/24, which is connected to the 2500. View the routing table.

At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600*. Configure the static route: **ip** `route 192.168.3.0 255.255.255.0 192.168.2.2`. View the static route: **show** `ip route`. Make sure that 192.168.3.0/24 shows up in the routing table as a static route (S).

6. From Host3, ping the `fa0/0` interface of the 2600. From Host3, ping Host1.

   At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host3*. Access Host3 and ping the `fa0/0` interface of the 2600 router: **ping** `192.168.1.1`. The ping should be successful. Ping Host1 **ping** `192.168.1.10`. The ping should be successful.

Now you should be more comfortable with configuring static routes. In the next section, you will grow acquainted with routing between VLANs by using a router-on-a-stick.

Router-on-a-Stick

Typically, we think of routing as traffic coming in one interface and leaving another interface. As you learned in Chapter 8, however, trunks can be used to support multiple broadcast domains, where each broadcast domain has a unique layer-3 network or subnet number. Certain router models, like the 2600 series, support trunk connections. A *router-on-a-stick* is a router that has a single trunk connection to a switch and that routes between the VLANs on this trunk connection. You could easily do this without a trunk (access-link connections), but each VLAN would require a separate access-link (physical) connection on the router, and this would increase the price of the router solution.

exam Watch

A router-on-a-stick is a router that has a single trunk connection to a switch and that routes between different VLANs on this trunk.

For instance, if you had five VLANs, and your router didn't support trunking, you would need five physical LAN interfaces on your router in order to route between the five VLANs. However, with a trunk connection, you can route between all five VLANs on a *single* interface. Because of cost and scalability, most administrators prefer using a router-on-a-stick approach to solve their routing problems in switched networks.

Subinterface Configuration

In order to set up a router-on-a-stick, you need to break up your router's physical interface into multiple logical interfaces, called *subinterfaces*. Cisco supports up to 300 interfaces on a router, which includes both physical and logical interfaces. Once you create a subinterface, a router will treat this logical interface just like a physical interface: you can assign layer-3 addressing to it, enable, it, disable it, and many other things.

To create a subinterface, use the following command:

```
Router(config)# interface type port_#.subinterface_#
                          [point|multipoint]
Router(config-subif)#
```

After entering the physical interface type and port identifier, follow this with a "." and a subinterface number. This number can range 0–4,294,967,295. The number that you use for the subinterface number is only for reference purposes within the IOS, and the only requirement is that when creating a subinterface, you use a unique number. Many administrators prefer to use the VLAN number that the subinterface will handle for the subinterface number; however, this is not a requirement.

exam Watch

Be familiar with how to create a subinterface with the `interface` command.

At the end of the statement, you must specify the type of connection *if* the interface is of type `serial`; otherwise, you can omit it. The **`point`** parameter is used for point-to-point serial connections, and **`multipoint`** is used for multipoint connections. The **`multipoint`** parameter is used for connections that have more than one device connected to them (physically or logically). Prior to IOS 12.0, if you omitted the connection type, it defaulted to **`multipoint`**. In 12.0 and higher, however, you must specify the type—there is no default. (This point is covered in more depth in Chapter 16.) For a router-on-a-stick configuration, *omit* the connection type, since it isn't used.

Interface Encapsulation

Once you create a subinterface, you'll notice that your CLI prompt has changed and that you are now in *Subinterface Configuration* mode. If you are routing between VLANs, you'll need an interface that supports trunking. There are some things configured on the major interface and some things configured on the subinterface. Configurations like duplexing and speed are done on the major (or physical) interface. Most other tasks are done on the subinterface, including which VLAN the subinterface belongs to and its IP addressing information.

When setting up your subinterface for a router-on-a-stick, one thing that you must configure is the type of trunking—ISL or 802.1Q—and the VLAN the subinterface is associated with, like this:

```
Router(config)# interface type port_#.subinterface_#
Router(config-subif)# encapsulation isl|dot1q VLAN_#
```

Use the **encapsulation** command to specify the trunk type and the VLAN associated with the subinterface. The VLAN number you specify here *must* correspond to the correct VLAN number in your switched network. You must also set up a trunk connection on the switch for the port that the router is connected to. Once you do this, the switch will send tagged frames to the router, and the router, using your encapsulation, will understand how to read the tags. The router will be able to see which VLAN the frame came from and match it up with the appropriate subinterface that will process it.

Example Configuration

Let's look at an example to see how a router-on-a-stick is configured. I'll use Figure 9-1 for this configuration. I'll assume that this is a 3600 router, that the Fast Ethernet interface is the first interface in the first slot, and that the switch is using ISL trunking.

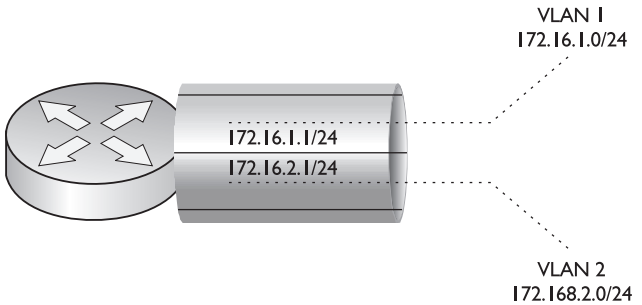
Here's the code example for this router:

```
Router(config)# interface fastethernet 0/0
Router(config-if)# duplex full
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface fastethernet 0/0.10
Router(config-subif)# encapsulation isl 1
Router(config-subif)# ip address 172.16.1.1 255.255.255.0
Router(config-subif)# exit
Router(config)# interface fastethernet 0/0.20
Router(config-subif)# encapsulation isl 2
Router(config-subif)# ip address 172.16.2.1 255.255.255.0
Router(config-subif)# exit
```

Notice in this example that the subinterface numbers (10 and 20) do not match the VLAN numbers in the encapsulation (1 and 2)—remember that the subinterface numbers are used by the IOS only to reference the particular subinterface and do not have to match any configuration on the subinterface.

FIGURE 9-1

Router-on-a-stick example



on the job

If you are configuring static routes and want to route traffic out of a particular subinterface, specify the major interface along with the subinterface number, like `fastethernet0/0.20`.

CertCam

9.02. The CD contains a multimedia demonstration of setting up a router-on-a-stick.

Dynamic Routing Protocols

Unlike static routes that require manual configuration to tell the router where destination networks are, *dynamic routing protocols* learn about destination networks from neighboring routers. Dynamic routing protocols fall under one of three categories: distance vector, link state, and hybrid. Each of these routing protocol types takes a different approach in sharing routing information with neighboring routers and choosing the best path to a destination.

Because of the differences between the various routing protocol types, each has advantages and disadvantages. One choice you'll have to make will be which routing protocol you'll run on the routers in your network. There are various factors that you'll have to examine when choosing a routing protocol:

- Routing metrics used to choose paths
- How routing information is shared
- Convergence speed of the routing protocol
- How routers process routing information
- Overhead of the routing protocol

Routing Metrics

As mentioned in the section "Administrative Distance," if your router has two types of routes, such as RIP and IGRP, for the same network number, the router uses the

administrative distance to choose the best one. However, a situation might arise where there are two paths to the destination network, and the same routing protocol, RIP, for instance, discovers these multiple paths to the destination network. If this is the case, a routing protocol will use a measurement called a *metric* to determine which path is the best path.

Table 9-3 lists some common metrics, the routing protocols that use them, and brief descriptions. As you can see from this table, some routing protocols use only a single metric. For instance, IP RIP uses hop count as a metric, and OSPF uses cost. Other routing protocols use multiple metric values to choose a best path to a destination. For instance, IP EIGRP and IGRP can use bandwidth, delay, reliability, load, and MTU when choosing a best path to a destination.

Distance Vector Protocols

Of the three types of routing protocols—distance vector, link state, and hybrid—distance vector protocols are the simplest. *Distance vector* routing protocols use the distance and direction (vector) to find paths to destinations. Most distance vector protocols use the Bellman-Ford algorithm for finding paths to networking destinations. Sometimes these protocols are referred to as *routing by rumor*, since the routers learn routing information from directly connected neighbors, and these neighbors might have learned these networks from other neighboring routers. Some examples of IP routing protocols that are distance vector are RIPv1 and IGRP. These protocols are discussed in depth in Chapter 10.

TABLE 9-3 Routing Protocol Metrics

Metric	Routing Protocols	Description
Bandwidth	IP EIGRP, IP IGRP	The capacity of the links in Kbps (T1=1,554)
Cost	IP OSPF, IPX NSLP	Measurement in the inverse of the bandwidth of the links
Delay	IP EIGRP, IP IGRP	Time it takes to reach the destination
Hop count	IP RIP, IPX RIP	How many routers away from the destination
Load	IP EIGRP, IP IGRP	The path with the least utilization
Maximum Transmission Unit (MTU)	IP EIGRP, IP IGRP	The path that supports the largest frame sizes
Reliability	IP EIGRP, IP IGRP	The path with the least amount of errors or down time
Ticks	IPX RIP	Measurement in delay (55 milliseconds)

Advertising Updates

One of the mechanisms of a routing protocol is to share information with neighboring routers. Some protocols use local broadcasts to disseminate information, some use multicasts, and some use unicasts. Distance vector protocols periodically use local broadcasts with a destination IP address of 255.255.255.255 to share routing information. These protocols do this religiously, whether or not something has changed: once their periodic timer expires, they broadcast their routing information to any devices connected to their interfaces. Note that distance vector protocols really don't care who listens to these updates, nor do they verify if neighboring routers received the broadcast update.

Routers running distance vector protocols learn who their neighbors are by listening for routing broadcasts on their interfaces. There is no formal handshaking process or hello process to discover who the neighboring routers are. Distance vector protocols assume that through the broadcast process, neighbors will be learned, and if a neighbor fails, the missed broadcasts from these neighbors will eventually be detected. And even if changes occur and your router misses an update from a neighbor, it is assumed that your router will learn about the change in the next broadcast update.

Processing Updates

When a distance vector protocol receives a routing update, it performs these steps:

1. Increment the metrics of the incoming routes in the advertisement (for IP RIP, add 1 to the hop count).
2. Compare the network numbers in the routing update from the neighbor to what the router has in its routing table.
3. If the neighbor's information is better, place it in the routing table and remove the old entry.
4. If the neighbor's information is worse, ignore it.
5. If the neighbor's information is exactly the same as the entry already in the table, reset the timer for the entry in the routing table (in other words, the router already learned about this route from the same neighbor).
6. If the neighbor's information is a different path to a known destination network, but with the same metric as the existing network in the routing table, the router will add it to the routing table along with the old one. This assumes you have not exceeded the maximum number of equal-cost paths for this destination network number. In this situation, your router is learning about the same network number from two different neighbors, and both neighbors are advertising the network number with the same metric.

exam Watch

Remember the advantages of distance vector protocols and how they process updates with the Bellman-Ford algorithm. Distance vector protocols use broadcasts to disseminate routing information and do not care if neighbors listen to their routing updates.

The six steps are generally referred to as the Bellman-Ford algorithm. As you can see from step 6, Cisco supports load balancing for equal-cost paths to a destination within a particular route type, such as IP RIP routes.

Since distance vector protocols are the simplest of the three, they are easy to set up and troubleshoot. They have very low overhead on the router, requiring few CPU cycles and memory to process updates: they receive an incoming update, increment the metrics, compare the results to the routes in the routing table, and update the routing table if necessary.

Link State Protocols

Link state protocols use an algorithm called the *Shortest Path First (SPF)* algorithm, invented by Dijkstra, to find the best path to a destination. Whereas distance vector protocols rely on *rumors* from other neighbors about remote routes, link state protocols will learn the complete topology of the network: which routers are connected to which networks. Because of the size of a network, this can create scalability problems. Therefore, link state protocols typically contain capabilities to limit the scope of their learning process, limiting a router's knowledge of the network topology to a smaller number of routers and routes.

Examples of link state protocols include IP's OSPF and IS-IS and IPX's NLSP. OSPF is covered in more depth in Chapter 11. IS-IS is an ISO link state protocol. It was originally developed by DEC as the DECnet Phase V routing protocol. It can route for both TCP/IP traffic and CLNP and CLNS traffic. IS-IS provides for more scalability than OSPF but is more complex to configure. Quite a few ISPs use IS-IS as the routing protocol for their own network. IS-IS is covered in Cisco's BSCI CCNP exam. IPX supports three routing protocols: RIP (IPX version), NSLP, and EIGRP. NSLP allows IPX networks to scale to very large sizes.

Advertising Updates

Whereas distance vector protocols use local broadcasts to disseminate routing information, link state protocols use multicasts. A distance protocol will send out its routing table religiously on its periodic interval whether there are changes or not. Link state protocols

are smarter. They multicast what is called a Link State Advertisement (LSA), which is a piece of routing information that contains who originated the advertisement and what the network number is.

LSAs are typically generated only when there are changes in the network, which is more friendly to your networking resources. In other words, periodic updates are rare occurrences. Whereas distance vector protocols use local broadcasts, which are processed by every machine on the segment, link state protocols use multicasts, which are processed only by other devices running the link state protocol. Plus, link state protocols send their updates reliably. A destination router, when receiving an LSA update, will respond to the source router with an acknowledgment. This process is different from distance vector protocols, which don't verify that a routing update was received from neighboring routers.

As a router learns routes from the LSAs of routers in the network, it builds a complete topology of the network—what routers are connected to other routers, and what the network numbers are. Whereas distance vector protocols are referred to as *routing by rumor,* link state protocols are referred to as *routing by propaganda,* since link state routers are learning which routers are sourcing (connected to) a network number. The LSAs gathered by a link state router are then stored in a local database. Anytime there is a change in the database, the router runs the SPF algorithm. The SPF algorithm builds an inverted tree, with the router itself at the top, and other routers and network segments beneath it. This algorithm is somewhat similar to the STP algorithm that layer-2 devices use to remove loops. Depending on the tree structure and the metrics used, the link state router then populates the routing table with the best (shortest) paths to the networks in the SPF tree.

Advantages of Link State Protocols

One advantage link state protocols have is that they use a hierarchical structure that helps limit the distance that an LSA travels. This reduces the likelihood that a change in the network will impact every router. This process is different from distance vector protocols, which use a flat topology. With distance vector protocols, a change in one part of the network will eventually impact every router in the network. Depending on the configuration of routers in a link state protocol, this is not necessarily true. For instance, OSPF uses areas to help contain changes; therefore, a change in one area won't necessarily impact other areas.

A second advantage of link state protocols is that they use multicasts to share routing information. Multicasts are sent to a group of devices, whereas broadcasts are sent to everyone. Only other routers running the link state protocol will process these LSA packets. Plus, link state routers send out only *incremental* updates. Incremental updates are updates sent out when there is a change in the state of the network. This is much

more advantageous than what distance vector protocols do: broadcast updates based on a periodic timer, which is typically either 30 or 60 seconds. Once all the link state routers are booted up and they learn the topology of the network, updates are sent out only when changes take place, which shouldn't be that often. The advantage of this process is that you are using your network's bandwidth and resources more efficiently than with distance vector protocols.

A third advantage that link state protocols have over distance vector protocols is that they support route classless routing. Classless routing allows you to summarize a large group of contiguous routes into a smaller number of routes. This process is called variable-length subnet masking (VLSM) and classless inter-domain routing (CIDR). These concepts are discussed in depth in Chapter 12.

By summarizing routes, you are making the routing process more efficient. First, you are advertising a smaller number of routes. And second, in order for the summarized route to fail, all of the subnets or networks in the summarization must fail. As an example, you might have a WAN link that is *flapping*. A flapping route is going up and down, up and down, over and over again. This can create serious performance problems for link state protocols.

When you perform summarization, if the specific route within a summarized route is flapping, this will not affect the status of the summarized route, and thus won't impact many of the routers in your network. Third, by summarizing routes, you reduce the size of your router's routing link state database, which will reduce the number of CPU cycles required to run the SPF algorithm and update the routing table, as well as reduce your router's memory requirements.

A fourth advantage is that with the use of the SPF algorithm, routing loops will not be included in the population of the routing table. Routing loops can create problems with distance vector protocols; they are discussed in the section "Problems with Distance Vector Protocols" section later in this chapter.

Disadvantages of Link State Protocols

Given the advantages of link state protocols, they do have disadvantages. For instance, even though link state protocols can scale a network to a much larger size than distance vector protocols, they come with their own set of problems. First, link state protocols are more CPU- and memory-intensive. Link state protocols have to maintain more tables in memory: a neighbor table, a link state database, and a routing table. When changes take place in the network, the routers must update the link state database, run the SPF algorithm, build the SPF tree, and then rebuild the routing table, which requires a lot more CPU cycles than a distance vector protocol's approach: increment the metrics of incoming routes and compare this to the current routes in the routing table.

As an example, a flapping route in a link state network can kill the processing on many routers, especially if the change is occurring every 10–15 seconds. The advantage that distance vector protocols have is that the only time the routers have to perform a function is when they receive the periodic updates, and then processing these updates is router-friendly.

exam Watch

Link state protocols use the SPF algorithm to choose the best path. They are more CPU- and memory-intensive than distance vector protocols. However, they are more network friendly in that they use multicasts to disseminate routing information and only advertise changes. Plus, with route summarization and hierarchical routing, link state protocols can scale to very large network sizes.

Hybrid Protocols

A *hybrid* protocol takes the advantages of both distance vector and link state protocols and merges them into a new protocol. Typically, hybrid protocols are based on a distance vector protocol but contain many of the features and advantages of link state protocols. Examples of hybrid protocols include RIPv2, EIGRP, and BGP. RIPv2 is covered in more depth in Chapter 10. and EIGRP is covered in Chapter 11. BGP is beyond the scope of this book but is heavily emphasized on the CCNP BSCI exam.

As an example, Cisco's EIGRP routing protocol reduces the CPU and memory overhead by acting like a distance vector protocol when it comes to process routing updates. Instead of sending out periodic updates like a distance vector protocol, EIGRP sends out incremental, reliable updates via multicast messages, providing a more network- and router-friendly network. EIGRP supports many other features of link state protocols, such as VLSM and route summarization.

BGP is also a hybrid protocol, drawing a lot of its functionality from distance vector protocols. It is based on a standard (RFC 1772) and is used as the de facto routing protocol to interconnect ISPs on the Internet. Unlike most of the other

exam Watch

The focus of the CCNA exam for routing configuration is primarily on RIPv1 and IGRP; however, you will need to be familiar with the basics of RIPv1 and v2, IGRP, EIGRP, and OSPF. The following chapters in this book spend more time on the operation of these protocols, as well as their configuration.

protocols that use multicasts or broadcasts for dissemination, BGP sets up a TCP connection (port 179) to a neighboring peer and uses TCP to reliable share connection information. Like EIGRP and OSPF, BGP supports route summarization. Unlike these protocols, BGP was meant to route between autonomous systems.

Problems with Distance Vector Protocols

The remainder of this chapter focuses on the problems that pertain to distance vector routing protocols: they converge slowly, and they are prone to routing (layer-3) loops. The next few sections cover these problems, as well as present solutions implemented by distance vector protocols to solve these problems.

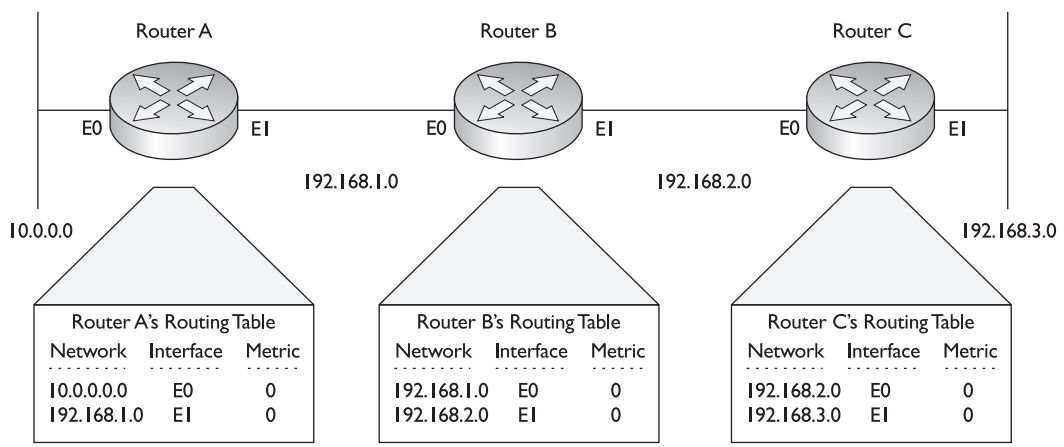
Problem: Convergence

The term *convergence*, in routing terms, refers to the time it takes for all of the routers to understand the current topology of the network. Link state protocols tend to converge very quickly, while distance vector protocols tent to converge slowly.

Convergence Example

To understand the issue that distance vector protocols have with convergence, look at an example. In this example, I'll assume that the periodic timer for the distance vector protocol is set to 60 seconds. I'll use the network shown in Figure 9-2. I'll also assume that the distance vector protocol is using hop count as a metric and that no special features are implemented in this example to solve convergence or routing loop problems.

FIGURE 9-2 Convergence example after routers turned on



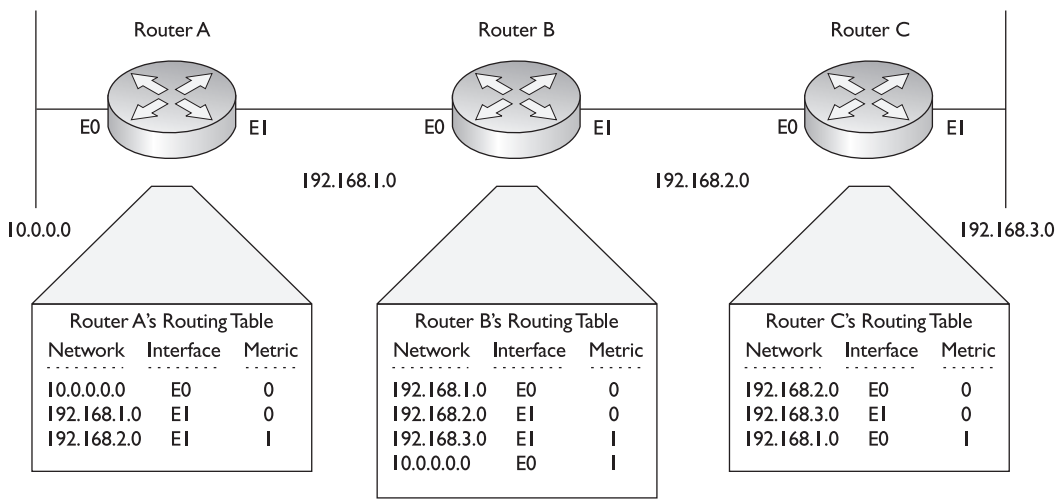
This example has three routers: RouterA, RouterB, and RouterC, where these routers were just turned on. As you can see from the routers' routing tables, the only routes these routers initially know about are the directly connected routes, which they learn by examining the status of their interfaces, making sure that they are *up* and *up*; they then take the network numbers (learned from the IP address and subnet mask) and put this information in their routing tables. Currently, each router contains two routes in its routing table. Also notice the metric: these routes have a hop count of 0, since they are directly connected.

Now that their interfaces are active and the routers have an initial routing table, they'll send out their first routing broadcast on these interfaces (they don't wait for their periodic timer in this instance). This broadcast contains the entries that they have in their routing tables. Let's assume that all routers are synchronized when they advertise their routing broadcasts, even though this would be highly unlikely in a production environment. This list shows which routers are advertising which routes on their active interfaces:

- **RouterA** Networks 10.0.0.0 and 192.168.1.0
- **RouterB** Networks 192.168.1.0 and 192.168.2.0
- **RouterC** Networks 192.168.2.0 and 192.168.3.0

After this first exchange of routing tables, each router will process its neighbor's received update and incorporate these changes, if necessary. Figure 9-3 displays the contents of the routing tables on the routers after this first exchange.

FIGURE 9-3 Convergence example after first routing update



Let's break this process down one router at a time, starting with RouterA:

1. Receives networks 192.168.1.0 and 192.168.2.0 from RouterB and increments the metric by one hop for each route
2. Compares the advertised routes from RouterB to what it has in its routing table
3. Adds 192.168.2.0 because it is not in the routing table
4. Ignores 192.168.1.0 from RouterB because RouterB has a hop count of 1, while the current routing table entry in the routing table has a hop count of 0

Let's look at RouterC next:

1. Receives networks 192.168.1.0 and 192.168.2.0 from RouterB and increments the metric by one hop
2. Compares the advertised routes from RouterB to what it has in its routing table
3. Adds 192.168.1.0 because it is not in the routing table
4. Ignores 192.168.2.0 from RouterB because RouterB has a hop count of 1 while the current routing table entry has a metric of 0

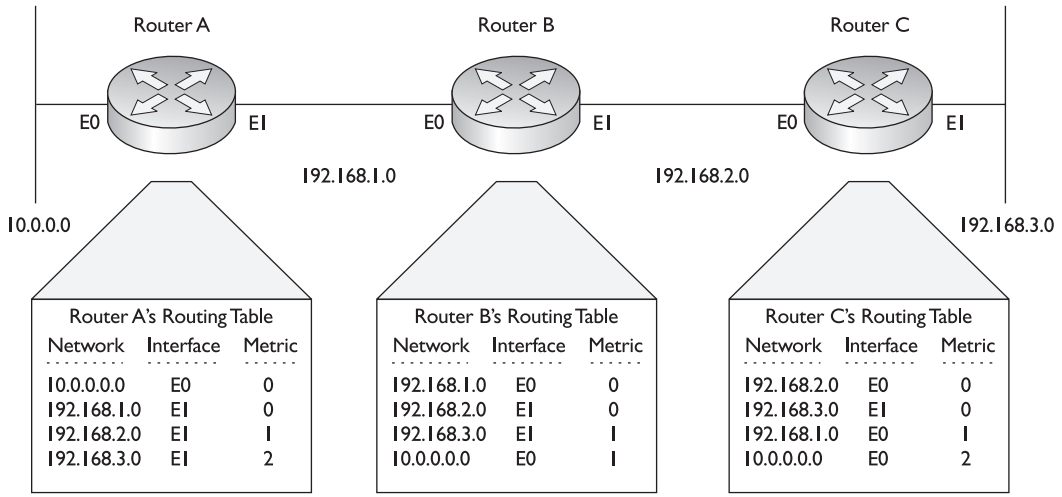
I've saved RouterB for last, since it presents a more complicated situation: it is receiving routes from both RouterA and RouterB. Here are the steps RouterB goes through:

1. Receives networks 10.0.0.0 and 192.168.1.0 from RouterA and 192.168.2.0 and 192.168.3.0 from RouterC and increments the metric by one hop
2. Compares the advertised routes from RouterA and RouterC to what it has in its routing table
3. Adds 10.0.0.0 and 192.168.3.0 because they are not currently in the routing table
4. Ignores 192.168.1.0 and 192.168.2.0 from RouterA and RouterC respectively because RouterA and RouterC have a metric of 1 for these routes, while the current routing table entries have a metric of 0

Looking at Figure 9-3, have the routers converged? Remember the definition of convergence: the routers understand the complete topology of the network. Given this definition, the routers have not yet converged. RouterA's routing table doesn't contain 192.168.3.0 and RouterC's routing table doesn't contain 10.0.0.0. Note, however, that RouterB has converged, but RouterA and RouterC still need additional routes.

After their periodic timers expire, the routers, again, generate local routing broadcast updates on each of their interfaces. Again, they broadcast their entire routing tables on these interfaces. Figure 9-4 shows the network after these routers

FIGURE 9-4 Convergence example after second routing update



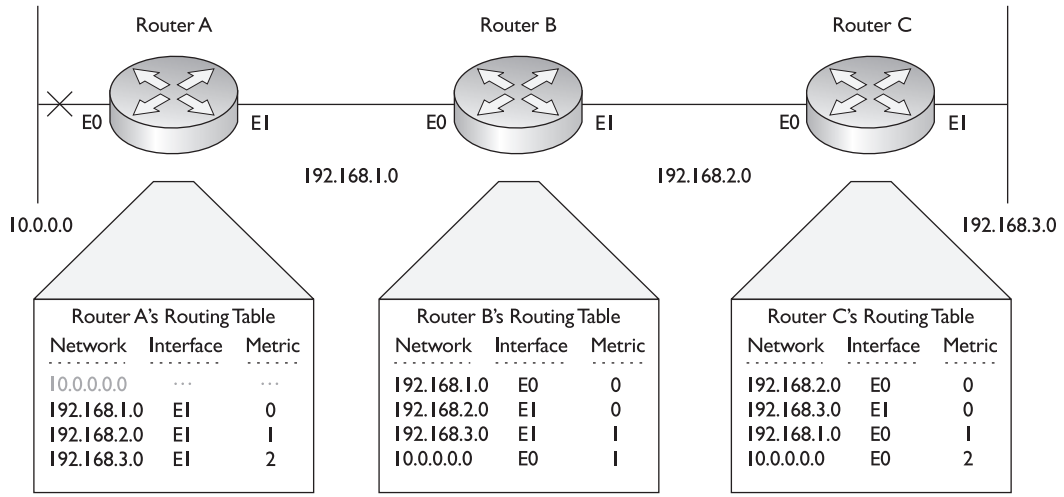
process these new updates. The routers in this network go through the same process again when receiving the updates. Notice that RouterA's routing table now contains 192.168.3.0, with a hop count of 2, while RouterC's routing table contains 10.0.0.0, with a hop count of 2. Both of these routers learned these networks via RouterB. And since these networks have a hop count of 1 on RouterB, when the edge routers receive the routing table from RouterB, they increment the hop count to 2 for these network numbers.

Given the routing tables shown in Figure 9-4, the routers have fully converged. The problem, however, is that convergence took place only after two updates. The first update took place as soon as the interface was active, and the second took place 60 seconds later. So in this example, it took over 60 seconds for convergence to take place. You can imagine that if you have a few hundred routers in your network, it might take many minutes before your network converges and each router knows about all of the destinations that are reachable.

Let's use the same network, but assume that RouterA's E0 interface has failed and that RouterA has lost its connection to network 10.0.0.0, as is shown in Figure 9-5. As you can see in this example, RouterA's routing table lists the network as unreachable. Unfortunately, RouterA cannot tell the rest of the network concerning the downed route *until* its periodic timer expires.

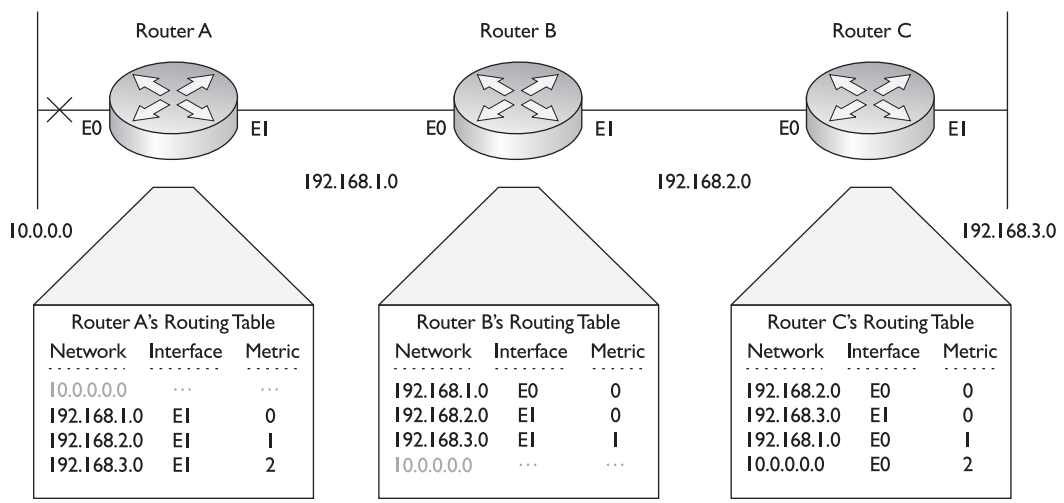
After the timer expires, RouterA advertises its routing table to RouterB, which is shown in Figure 9-6. After RouterB receives its update, it has converged. However,

FIGURE 9-5 RouterA's **E0** interface has failed.



RouterC is still lacking this information about the updated topology and must wait for RouterB's periodic timer to expire in order to receive RouterB's updated routing table.

FIGURE 9-6 RouterB receives the updated information.



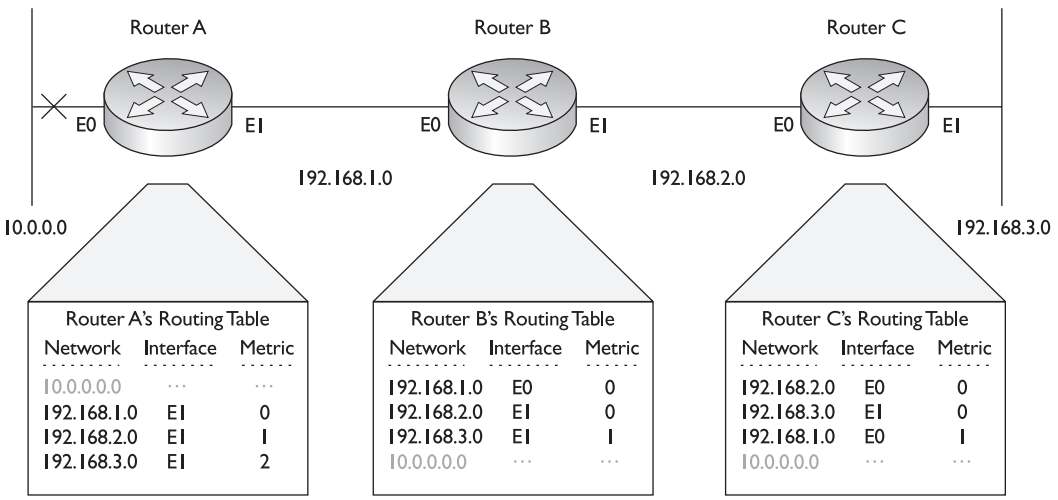
exam Watch

Convergence is when a router understands the current topology of the network. You should be able to figure out whether a distance vector protocol has converged or not by examining routing tables on routers.

After RouterB's periodic timer has expired, it shares its routing table with RouterC, as is shown in Figure 9-7. Up to this point, RouterC assumed that it had the most up-to-date routing information and would still send packets to 10.0.0.0, since the routing table indicated that 10.0.0.0 was reachable via RouterB. However, after receiving the routing update from RouterB, RouterC updates its routing table and knows that 10.0.0.0 is not reachable; it will now drop any packets being sent to 10.0.0.0.

Now all three routers have converged. Here are the three things that affected convergence in this example: the time it took for RouterA to discover that e0 failed (a few seconds); the periodic timer on RouterA to advertise this to RouterB (up to 60 seconds); and the periodic timer on RouterB to advertise this to RouterC (up to 60 seconds). Given these three items, it could take over two minutes to converge. As you can see from the past two examples, convergence, with distance vector protocols, is a slow process.

FIGURE 9-7 RouterC receives the updated information.



Solution: Triggered Updates

Now that you understand some of the problems associated with convergence in distance vector protocols, let's talk about a possible solution. Given the last three items I listed that affected convergence with the unreachable network (10.0.0.0), the two items that slowed down convergence were periodic timers. You actually have two solutions that you can use in order to speed convergence: change the periodic timer interval and/or use triggered updates.

One solution is to change the periodic timer interval. For instance, in our example the timer was set to 60 seconds. To speed up convergence, you might want to set the interval to 10 seconds. In our example, then, convergence would take only about 20 seconds. However, in today's networks, even waiting this amount of time still creates network disruptions. Also, by setting the timer to 10 seconds, you are creating six times the amount of routing broadcast traffic, which is not very efficient.

A second solution is to implement triggered updates. Triggered updates complement periodic updates. The distance vector routing protocol would still generate periodic updates; however, whenever a change takes place, the router will immediately generate an update without waiting for the periodic timer to expire. This can decrease convergence times, but it also creates a problem. If you have a flapping route, then an update will be triggered each time the route changes state, which creates a lot of unnecessary broadcast traffic in your network and could cause a broadcast storm. IGRP is an example of a distance vector protocol that implements triggered updates.

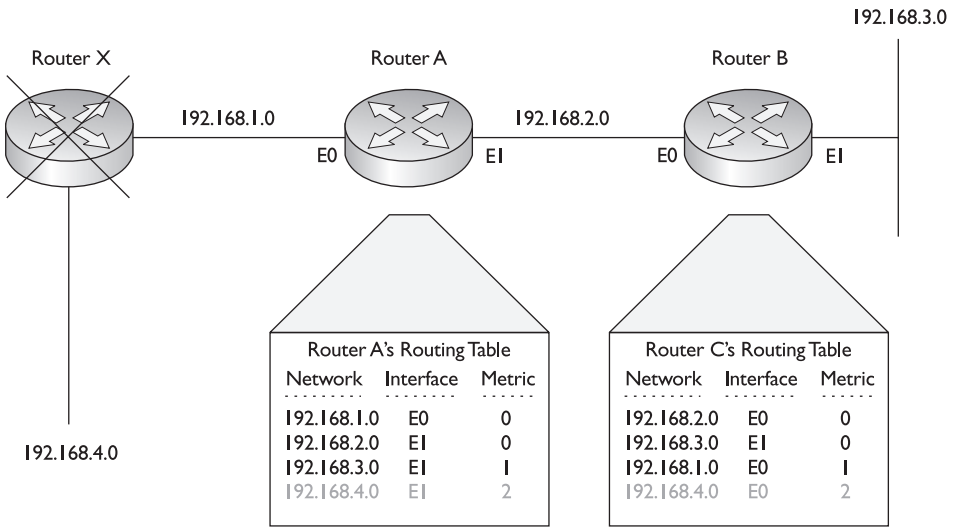
Problem: Routing Loops

The other main problem of distance vector protocols is that they are prone to routing loops. A *routing loop* is a layer-3 loop in the network. Basically, it is a disagreement about how to reach a destination network.

Routing Loop Example

Let's take a look at a simple example of what kind of problems routing loops can create. I'll use the network shown in Figure 9-8. In this example, I'll assume that RouterX was originally advertising 192.168.4.0 to RouterA, which passed this on to RouterB. RouterX, though, has failed and is no longer advertising 192.168.4.0. However, both RouterA and RouterB advertise 192.168.4.0 to each other, creating confusion about how to reach 192.168.4.0, if it can even be reached (which in this case, it can't). In this example, RouterA thinks that to reach 192.168.4.0, it should

FIGURE 9-8 Simple routing loop example



send these packets to RouterB. RouterB, on the other hand, thinks that to reach 192.168.4.0, it should use RouterA. This is a very simple example of a routing loop. Typically, routing loops are created because of confusion in the network related to the deficiencies of using periodic timers.

Distance vector protocols have mechanisms that they can use to deal with routing loop problems. However, these solutions slow down convergence. Link state and some hybrid protocols deal with routing loops with more intelligent methods that don't slow down convergence. The following sections cover the methods that a distance vector protocol might implement to solve routing loop problems.

Counting to Infinity Solution: Maximum Hop Count

One problem with a routing loop is the *counting to infinity* symptom. When a routing loops occurs and a packet or packets are caught in the loop, they continuously circle around the loop, wasting bandwidth on the segments and CPU cycles on the routers that are processing these packets.

To prevent packets from circling around the loop forever, distance vector protocols typically place a hop count limit as to how far a packet is legally allowed to travel. As a packet travels from router to router, a router keeps track of the hops in the TTL field

in the IP datagram header: for each hop a packet goes through, the packet's TTL field is decremented by one. If this value reaches 0, the packet is dropped by the router that decremented the value from 1 to 0. The function of the TTL field was covered in Chapter 3.

Placing a maximum hop count limitation on packets, however, doesn't solve routing loop problems—the loop still exists. This solution only prevents packets from getting stuck in the loop. Another issue with placing a hop count limit on packets is that in some instances, the destination that the packet is trying to reach exceeds the maximum hop count allowed. A router doesn't distinguish between valid destinations and routing loop destinations when examining the TTL field; if the maximum is reached, then the packet is dropped.

IP RIP and IPX RIP set a hop count limit of 15, by default, and IGRP allows a hop count of 100. When a packet comes into an interface of a router, it decrements the TTL field, and if the hop count falls to 0, the router immediately drops the packet. If you have a destination that is beyond these limits, you can change the maximum hop count for your routing protocol; however, you should do this on every router in your network.

exam Watch

Routing loops are a misunderstanding about the reachability of a destination and are a common problem with distance vector protocols. Different solutions are used to deal with routing loops and their problems. Counting to infinity is dealt with by assigning a hop count limit to limit how far a packet can travel.

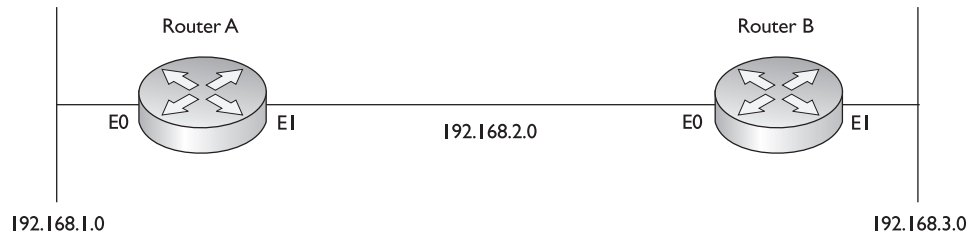
Solution: Split Horizon

Distance vector protocols implement a few solutions to deal with routing loops. *Split horizon* is used with small routing loops. Split horizon states that if a neighboring router sends a route to a router, the receiving router will *not* propagate this route back to the advertising router on the same interface.

Consider Figure 9-9 to see how split horizon functions. RouterA advertises 192.168.1.0 to RouterB out its E1 interface. Without split horizon in effect, RouterB could advertise this network right back to RouterA. Obviously, RouterA would ignore this, since the directly connected path is better than RouterB's advertised path. However, what would happen if RouterA's E0 interface failed and it received an update from RouterB stating that it had an *alternative* path to 192.168.1.0?

FIGURE 9-9

Split horizon example



exam Watch

Split horizon prevents a router from advertising a route back out the same interface where the router originally learned the route.

In this situation, the network obviously has connectivity problems. With split horizon, though, RouterB would never advertise 192.168.1.0 back to RouterA. Therefore, if RouterA's E0 interface would fail, both RouterA and RouterB would realize that there is no alternative path to reach this network until RouterA's E0 connection is fixed.

Solution: Route Poisoning

Whereas split horizon is used to solve small routing loop problems, distance vector protocols use two mechanisms to deal with large routing loop problems: *route poisoning* and *hold down timers*. Route poisoning is a derivative of split horizon. When a router detects that one of its connected routes has failed, the router will *poison* the route by assigning an infinite metric to it. In IP RIP, the route is assigned a hop count of 16 (15 is the maximum), thus making it an *unreachable* network.

exam Watch

A poisoned route has an infinite metric assigned to it. A poison reverse causes the router to break the split horizon rule and advertise the poisoned route out all interfaces.

When a router advertises a poised route to its neighbors, its neighbors break the rule of split horizon and send back to the originator the same poisoned route, called a *poison reverse*. This ensures that everyone received the original update of the poisoned route.

Hold-Down Timers

In order to give the routers enough time to propagate the poisoned route and to ensure that no routing loops occur while propagation is occurring, the routers implement a *hold-down* mechanism. During this period, the routers will freeze the poisoned route in their routing tables for the period of the hold-down timer, which is typically three times the interval of the routing broadcast update.

When hold-down timers are used, a poisoned route will remain in the routing table until the timer expires. However, if a router with a poisoned route receives a routing update from a neighboring router with a metric that is the same or better than the original route, the router will abort the hold-down period, remove the poisoned route, and put the new route in its table. However, if a router receives a worse route from a neighboring router, the router treats this as a suspect route and assumes that this route is probably part of a routing loop, ignoring the update. Of course, the worse-metric route really might be a valid alternative path to the network; however, the function of hold-down timers and poisoning routes prohibits the use of this route until the hold period expires. While in a hold-down state, a poisoned route in the routing table will appear as *possibly down*.

One of the problems of using hold-down timers is that they cause the distance vector routing protocol to converge slowly—if the hold-down period is 180 seconds, you can't use a valid alternative path with a worse metric until the hold-down period expires. Therefore, your users will lose their connections to this network for at least three minutes.

exam Watch

Hold-down timers are used to keep the poisoned route in the routing table long enough that the poisoned route has a chance to be propagated to all other routers in the network. One downside to hold-down timers is that they slow down convergence.

Example of Route Poisoning and Hold-Down Timers

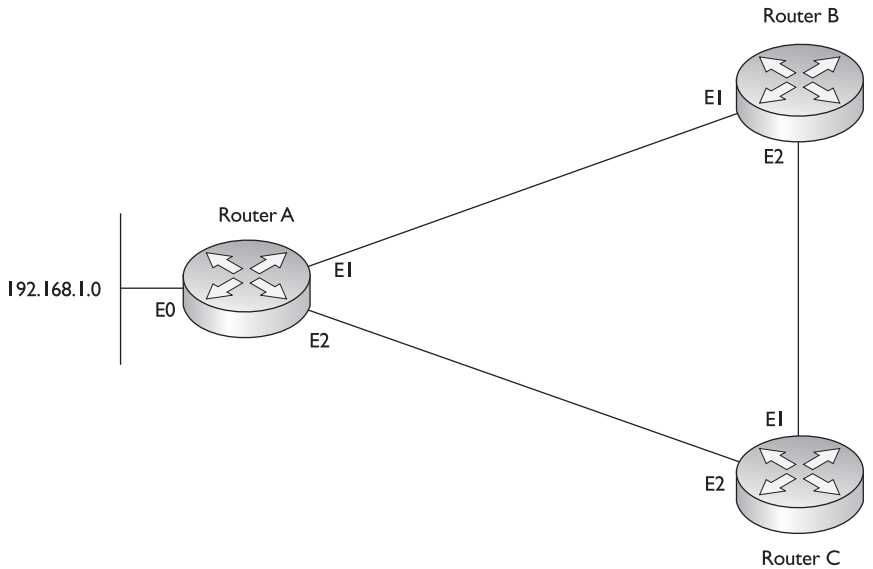
Understanding how poisoned routes and hold-down timers work can become complex. Let's take a look at an example to see how these two mechanisms work hand-in-hand to solve large routing loop problems. I'll use the network shown in Figure 9-10. In this example, I'll assume the routers are running IP RIPv1.

In this example, RouterA's E0 interface fails, causing it to lose its connection to 192.168.1.0. Since RIPv1 doesn't use triggered updates, as IGRP does, the routing protocol must wait for its periodic timer to expire before broadcasting its routing information to RouterB and RouterC. In RIPv1, the periodic update timer is set to 30 seconds. RouterA will poison the route (assign an infinite metric of 16 to 192.168.1.0) and send this to the other two routers when the periodic update timer expires.

When RouterB and RouterC receive the routing update with the poisoned route from RouterA, they will send back a poison reverse to RouterA. All routers will freeze

FIGURE 9-10

Route poisoning and hold-down timer example



the poisoned route in their routing table for the period of the hold-down timer. In RIPv1, this defaults to 180 seconds. RouterB and RouterC also advertise the poised route in their routing updates out any other active interfaces (once their periodic timers expire). As the propagation of the poisoned route is occurring, the routes that have already received it are counting down from their hold-down timer value.

If another router in the network advertises a worse path to 192.168.1.0 (this has to be a worse hop count than the route originally advertised from RouterA), the three routers shown in the network diagram won't use it, since they have frozen the poisoned route in their routing tables. The reason for this hold-down period is that someone else might be advertising 192.168.1.0, but it might not be a valid path. In other words, another router might be advertising reachability to 192.168.1.0, but it is assuming that this network is reachable via RouterA. In this situation, this rogue router hasn't received the poisoned route—the hold-down timer for the other routers, however, ensures that these rogue routers don't corrupt the routing tables by introducing incorrect or bad routing information, causing a routing loop.

During this process, if RouterA is able to fix is connection to 192.168.1.0, it will start advertising the reachability of the network to RouterB and RouterC. Since the metric RouterA is advertising is the same as the metric it had previously announced for this route, RouterB and RouterC will cancel their hold-down times and replace the poisoned route with the new information.

EXERCISE 9-2

Basic IP and Routing Troubleshooting

This chapter dealt with the basics of routers and routing. This exercise is a troubleshooting exercise and is different from the other exercises you have performed so far. In previous exercises, you were given a configuration task. In this exercise, the network is already configured; however, there are three problems in this network you'll need to find and fix in order for it to operate correctly. All of these problems deal with IP (layer-3) connectivity. You'll perform this exercise using Boson's NetSim™ simulator. You can find a picture of the network diagram for Boson's NetSim™ simulator in the Introduction of this book. The addressing scheme is the same. After starting up the simulator, click on the *LabNavigator* button. Next, double-click on *Exercise 9-2* and click on the *Load Lab* button. This will load the lab configuration based on Chapter 5's exercises.

Lets' start with your problem: Host1 cannot ping Host3. Your task is to figure out what the problems are and fix them: there are three problems. I would recommend that you try this troubleshooting process on your own first; and if you have problems, come back to the steps and solutions provided below.

1. Test connectivity from Host1 to Host3 with ping as well as from Host1 to its default gateway.

   At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host1*. On Host1, ping Host3: **`ping`** `192.168.3.2`. Note that the ping fails. Examine the IP configuration on Host1 by executing: **`winipcfg`**. Make sure the IP addressing information is correct: IP address of 192.168.1.10, subnet mask of 255.255.255.0, and default gateway address of 192.168.1.1. Click on the *Cancel* button to close **`winipcfg`**. Ping the default gateway address: **`ping`** `192.168.1.1`. The ping should be successful, indicating that at least layer-3 is functioning between Host1 and the 2600.

2. Test connectivity from Host3 to its default gateway.

   At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host3*. Examine the IP configuration on Host3 by executing: **`winipcfg`**. Make sure the IP addressing information is correct: IP address of 192.168.3.2, subnet mask of 255.255.255.0, and default gateway address of 192.168.3.1. Click on the *Cancel* button to close **`winipcfg`**. Ping the default gateway address: **`ping`** `192.168.3.1`. The ping should fail, indicating that there is a problem between Host3 and the 2500. In this example, layer-2 is functioning correctly; therefore, the problem must lie with the 2500.

3. Check the 2500's IP configuration.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2500*. From the 2500, ping Host3: **ping** `192.168.3.2`. The ping should fail. Examine the interface on the 2500: **show** `interface e0`. The interface is disabled, but has the correct IP address: 192.168.3.1. Enable the interface: **configure** `terminal`, **interface** `e0`, **no** `shutdown`, **end**. The interface should come up. Retry the ping test: **ping** `192.168.3.2`. The ping should be successful.

4. Access Host1 and retry pinging Host3.

   At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host1*. Test connectivity to Host3: **ping** `192.168.3.2`. The ping should still fail. So far, at least there is connectivity within 192.168.1.0 and 192.168.3.0; but there is still a problem between these two networks.

5. Check the interface statuses on the 2600 and verify connectivity to the 2500.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600*. Check the status of the interfaces: **show** `ip interface brief`. Notice that the `fa0/0` and `s0` are *up and up*. Try pinging the 2500's `serial0` interface: **ping** `192.168.2.2`. The ping fails. Examine the CDP information that the 2600 has learned about the 2500: **show** `cdp entry 2500`. Notice that the 2500 has the wrong IP address (192.168.22.2).

6. Fix the IP addressing problem on the 2500 and retest connectivity across the serial connection. Test connectivity from the 2500 to Host1.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2500*. Fix the IP address: **configure** `terminal`, **interface** `serial0`, **ip** `address 192.168.2.2`, **end**. Retest the connection to the 2600: **ping** `192.168.2.1`. The ping should be successful.

7. Test connectivity from the 2500 to Host1. Examine the routing table.

   Test the connection to Host1: **ping** `192.168.1.10`. The ping should be successful. This indicates that you have full layer-3 connectivity between the 2500 and Host1 and the 2500 and Host3; however, there is an issue getting traffic through the 2500. Examine the routing table: **show** `ip route`. As you can see, 192.168.1.0 shows up as a static route and points to 192.168.2.1.

8. Access Host3 and try connectivity between its default gateway and the 2600 router.

   At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host3*. Test the connection to the 2500: **ping** `192.168.3.11`. The ping should be successful, considering that we already tested it. Test

connectivity to the 2600: **ping** `192.168.2.1`. The ping should fail. This presents an interesting problem Host1 and ping the 2600. The 2600 can ping the 2500. Host3 can ping the 2500.So, on a hop-by-hop basis, we have IP connectivity. The 2500 can even ping Host1, indicating that some routing functioning is working.

9. Access the 2600 router and examine its routing table. Fix the problem.

   At the top of the simulator in the menu bar, click on the *eRouters* icon and choose *2600*. Examine the routing table: **show** `ip route`. Does the 2600 know how to reach 192.168.3.0/24?It does not. The 2500 router could ping Host1 since the 2600 router is directly connected to these segments; but any traffic from the 2600 to 192.168.3.0/24 will fail since the router doesn't have a path. Add a static route to 192.168.3.0/24: **ip** `route 192.168.3.0 255.255.255.0 192.168.2.2`. Test connectivity to Host3: **ping** `192.168.3.2`. The ping should be successful.

10. Now test connectivity between Host1 and Host3.

    At the top of the simulator in the menu bar, click on the *eStations* icon and choose *Host1*. Test connectivity to Host3: **ping** `192.168.3.2`. The ping should be successful.

CERTIFICATION SUMMARY

Routers find layer-3 paths of destination networks and switch packets from one interface to another to get the packets to their respective destinations. Routers learn about neighboring routers, find locations to destination locations, choose the best paths, and maintain up-to-date routing information. A routed protocol is a layer-3 protocol, like IP or IPX. A routing protocol defines how to find destinations for a routed protocol.

Some routing protocols, such as IGRP and EIGRP, use autonomous systems, which group networks under a single administrative control. Administrative distance is used by a Cisco router to choose among multiple routing protocols to put a destination in the routing table. The routing protocol with the lowest administrative distance with a path to the destination is placed in the routing table.

There are two types of routing protocols: static and dynamic. To create a static route, use the **ip** `route` command. For a default route, use `0.0.0.0/0` as the network number and subnet mask. To view your router's routing table, use the **show** `ip route` command.

A router-on-stick uses a single trunk connection from a router to a switch to route among multiple VLANs. You must create a subinterface on your router for each VLAN. Each subinterface requires the **`encapsulation`** `isl|dot1q` command and a layer-3 address or addresses.

When choosing a dynamic routing protocol, you should consider routing metrics, how routing information is shared, convergence time, how routing information is processed, and routing overhead. Routing metrics define the method used to calculate a cost to a destination. For instance, IP RIP uses hop count.

Distance vector protocols use broadcasts to share routing information and don't verify if neighbors receive routing updates. They use the Bellman-Ford algorithm to process updates, which requires very little CPU processing and memory: They receive an update, increment the metrics, compare the results to the routing table, and update the routing table if necessary.

Link state protocols use the SPF algorithm to build the routing table, providing a loop-free topology. They use multicasts to share routing information incrementally and verify that neighbors received this information. Link state protocols support classless routing and allow you to summarize networking information in your routing table. The main downside of these protocols is that they require more CPU cycles and memory to process and store routing information. They are also prone to flapping route problems.

Hybrid protocols are based on the simplicity of a distance vector protocol but borrow on many features of link state protocols to make them more efficient and scalable. RIPv2, EIGRP, and BGP are examples of hybrid protocols.

Distance vector protocols have problems with convergence and routing loops. Convergence is the amount of time it takes for all of the routers in the network to understand the current topology. Triggered updates can be used to speed up convergence. A routing loop is basically a disagreement about how to reach a particular network. Counting to infinity is resolved by placing a hop count limit to prevent packets from circling around the loop forever. Split horizon is used to prevent the creation of small routing loops: It prevents a router from advertising a route out the same interface from which the route was learned.

Route poisoning, poison reverse, and hold-down timers are used to prevent large routing loops. A route is poisoned if a network connected to a router goes down. Poison reverse has a router advertise a poisoned route out all interfaces, including the interface from which it was learned. Hold-down timers keep the poisoned route in the routing table to ensure the poisoned route is propagated to all routers before any (worse) alternative paths are chosen.

TWO-MINUTE DRILL

Types of Routes

- Routers learn about connected routers, find locations of destination networks, choose the best paths to each destination, and maintain their routing tables.
- A static route is a manually configured route. A connected route is a network that the router is directly connected to on an interface.
- An autonomous system (AS) is a group of networks under a single administrative control, which could be your company, a division within your company, or a group of companies.
- Administrative distance is a Cisco-proprietary mechanism used to rank IP routing protocols and helps the router populate the routing table. If you are running more than one routing protocol, this is used to determine which routing protocol to use when populating the routing table. The lower the administrative distance number, the more preferred it is.

Static Routes

- Use the **`ip`** `route` command to configure a static route.
- After the subnet mask parameter, you have two ways of specifying how to reach the destination network: you can tell the router either the next hop neighbor's IP address or the interface the router should exit to reach the destination network. The former has an administrative distance of 1 and the later, 0 (a directly connected route).
- Use the **`ip`** `classless` command to allow a routing protocol to accept a route with a nonconforming subnet mask value.

Router on a Stick

- A *router-on-a-stick* is a router with a single trunk connection to a switch; it routes between the VLANs on this trunk connection.
- To route between VLANs with a router-on-a-stick, use subinterfaces and specify the VLAN with the **`encapsulation`** `isl|dot1q` command on the subinterface.

Dynamic Routing Protocols

- Internally, a routing protocol will use a metric to choose a best path to reach a destination.
- Distance vector protocols, such as RIPv1 and IGRP, use distance and direction to find paths to a destination and are referred to as routing by rumor. They generate periodic updates as broadcasts and build no formal relationship with other routers. They require little processing and memory, since they only need to increment metrics and compare these to routes in the routing table.
- Link state protocols, such as OSPF, use the SPF algorithm and understand the complete topology of the network (routing by propaganda). They multicast LSAs, which are specific routes, when changes occur in the network. The LSAs are stored in a link state database. These protocols converge fast, since they use incremental updates, but require more memory and processing power. They also support a hierarchical structure and route summarization.
- Hybrid protocols, such as RIPv2 and EIGRP, take the advantages of both distance vector and link state protocols and merge them together.

Problems with Distance Vector Protocols

- Distance vector protocols converge slowly because of periodic updates. IGRP overcomes this by using triggered updates.
- Distance vector protocols are prone to routing loops. To solve this, they use these mechanisms: hop count limits, split horizon, poisoned routes, and hold-down timers.
- To prevent packets from circling around the loop forever, distance vector protocols typically place a hop count limit as to how far a packet is legally allowed to travel, referred to as maximum hop count, or time-to-live (TTL).
- Split horizon states that if a neighboring router sends a route to a router, the receiving router will not propagate this route back to the advertising router on the same interface.
- With route poisoning, when a router detects that one of its connected routes has failed, the router will *poison* the route by assigning an infinite metric to it and advertising it to neighbors. When a router advertises a poised route to its neighbors, its neighbors break the rule of split horizon and send back to the originator the same poisoned route, called a *poison reverse*.
- In order to give the routers enough time to propagate the poisoned route and to ensure that no routing loops occur while propagation is occurring, the routers implement a hold-down mechanism.

SELF TEST

The following Self Test questions will help you measure your understanding of the material presented in this chapter. Read all the choices carefully, as there may be more than one correct answer. Choose all correct answers for each question.

Types of Routes

1. _____ is/are a routed protocol.
   - A. RIP
   - B. OSPF
   - C. Both RIP and OSPF
   - D. Neither RIP or OSPF
2. A _____ routes between different autonomous systems.
   - A. BPG
   - B. EGP
   - C. IPP
   - D. IGP
3. Your router is running RIP and OSPF and both routing protocols are learning 192.168.1.0/24. Which routing protocol will your router use for this route?
   - A. RIP
   - B. OSPF

Static Routes

4. Enter the command to set up a static route to 192.168.1.0/24, where the next hop address is 192.168.2.2: _____.
5. What subnet mask would you use to set up a default route?
   - A. 0.0.0.0
   - B. 255.255.255.255
   - C. Depends on the type of network number
   - D. None of these answers
6. Enter the command to allow your routing protocol to accept nonconforming subnet masks, like a default route: _____.

Router-on-a-Stick

7. When configuring a router-on-a-stick, the configuration is done on _____.

A. Physical interfaces
B. Major subinterfaces
C. Subinterfaces

8. Which router-on-a-stick command defines the VLAN for the interface?

A. `vlan`
B. `encapsulation`
C. `trunk`
D. `frame-type`

Dynamic Routing Protocols

9. When choosing a dynamic routing protocol, which of the following should not be considered?

A. Metrics used
B. How routing information is shared
C. How routing information is processed
D. Number of PCs in the network

10. A routing protocol will use a _____ to determine which path is the best path.

A. Administrative distance
B. Metric
C. Hop count
D. Cost

11. Distance vector protocols use _____ to disseminate routing information.

A. Unicast
B. Multicast
C. Broadcast

12. Which type of routing protocol uses the Shortest Path First algorithm?

A. Distance vector
B. Link state
C. Hybrid

13. Which is an example of a hybrid protocol?

A. IGRP
B. EIGRP
C. RIPv1
D. OSPF

Problems with Distance Vector Protocols

14. What would you use to prevent a packet from traveling around a routing loop forever?

A. Split horizon
B. Poison reverse
C. Hold-down timer
D. TTL

15. _____ states that if a neighboring router sends a route to a router, the receiving router will not propagate this route back to the advertising router on the same interface.

A. Split horizon
B. Poison reverse
C. Hold-down timer
D. Hop count limit

SELF TEST ANSWERS

Types of Routes

1. ☑ **D.** RIP and OSPF are *routing* protocols. Routed protocols are IP, IPX, AppleTalk, etcetera.
   ☒ **A**, **B**, and **C** are routing protocols.
2. ☑ **B.** An Exterior Gateway Protocol (EGP) routes between autonomous systems.
   ☒ **D** routes within an AS. **A** and **C** are nonexistent routing protocols.
3. ☑ **B.** OSPF has a lower administrative distance: 110. The lower one is given preference.
   ☒ **A** has a higher administrative distance: 120.

Static Routes

4. **SYMBOL 254 \f "Wingdings" \s 11** `ip route 192.168.1.0 255.255.255.0 192.168.2.2`
5. ☑ **A.** A default route is set up with an IP address and mask of 0.0.0.0 0.0.0.0.
   ☒ **B** indicates that the complete IP address is a network number. **C** is dependent on the mask, not the network number. And since there is a correct answer, **D** is incorrect.
6. **SYMBOL 254 \f "Wingdings" \s 11** `ip classless`

Router-on-a-Stick

7. ☑ **C.** When configuring a router-on-a-stick, the configuration is done on subinterfaces.
   ☒ **A** is used for access-link connections, not trunk connections. **B** is non-existent.
8. ☑ **B.** Use the **`encapsulation`** command to specify the trunking encapsulation and the VLAN number for the subinterface.
   ☒ **A**, **C**, and **D** are nonexistent router commands.

Dynamic Routing Protocols

9. ☑ **D.** It's not the number of devices, but the number of networks, that might affect the routing protocol that you choose.
   ☒ **A**, **B**, and **C** are incorrect because they are things you should consider when choosing a routing protocol.

10. ☑ **B.** A routing protocol will use a metric to determine which path is the best path.
☒ **A** is used to choose between different routing protocols, not within a routing protocol. **C** and **D** are types of metrics.

11. ☑ **C.** Distance vector protocols use broadcasts to share routing information.
☒ **A** is used by BGP, which is a hybrid protocol. **B** is used by link state and hybrid protocols.

12. ☑ **B.** Link state protocols use the SPF algorithm, developed by Dijkstra, in order to choose the best path to a destination.
☒ **A** uses distance and direction when choosing best paths. **C** protocols typically use methods based on distance vector protocols.

13. ☑ **B.** EIGRP is a hybrid protocol, along with RIPv2.
☒ **D** is a link state protocol. **A** and **C** are distance vector protocols.

Problems with Distance Vector Protocols

14. ☑**D.** TTL, which implements a hop count limit, prevents an IP packet from traveling around a routing loop forever.
☒ **A** is used to prevent small routing loops, preventing the advertisement of a route out the same interface it was learned on. **B** and **C** are used to prevent large routing loops: they allow network stabilization by waiting until every router learns about the downed route before accepting an alternative path.

15. ☑**A.** Split horizon is used to prevent small routing loops, preventing the advertisement of a route out the same interface it was learned on.
☒ **B** assigns an infinite metric to the route, sends it to a neighboring router, and has the neighbor advertise this back to you. **C** sets a timer that a poisoned route is held in the routing table. **D** is used to prevent a packet from traveling around a routing loop forever.