CHAPTER 2

Routing Information Protocol (RIP)

In this chapter:

RIP is the first in a family of dynamic routing protocols that we will look at closely. Dynamic routing protocols *automatically* compute routing tables, freeing the network administrator from the task of specifying routes to every network using static routes. Indeed, given the complexity of and number of routes in most networks, static routing usually is not even an option.

In addition to computing the "shortest" paths to all destination networks, dynamic routing protocols discover alternative (second-best) paths when a primary path fails and balance traffic over multiple paths (load balancing).

Most dynamic routing protocols are based on one of two distributed algorithms: Distance Vector or Link State. RIP, upon which Cisco's IGRP was based, is a classic example of a DV protocol. Link State protocols include OSPF, which we will look at in a later chapter. The following section gets us started with configuring RIP.

Getting RIP Running

Throughout this book, we'll be using a fictional network called TraderMary to illustrate the concepts with which we're working. TraderMary is a distributed network with nodes in New York, Chicago, and Ames, Iowa, as shown in Figure 2-1.

As a distributed process, RIP needs to be configured on every router in the network:

```
hostname NewYork
...
interface Ethernet0
```

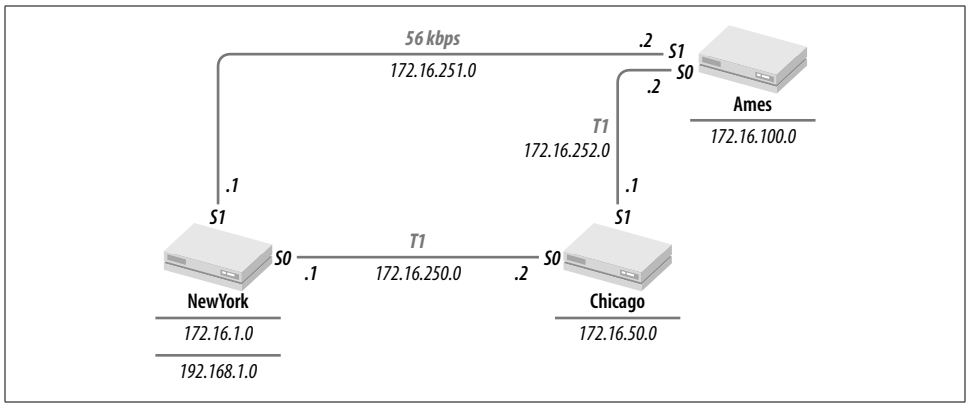


Figure 2-1. TraderMary's network

```
ip address 172.16.1.1 255.255.255.0
!
interface Ethernet1
ip address 192.168.1.1 255.255.255.0
!
interface Serial0
ip address 172.16.250.1 255.255.255.0
!
interface Serial1
ip address 172.16.251.1 255.255.255.0
...
router rip
network 172.16.0.0


hostname Chicago
...
interface Ethernet0
ip address 172.16.50.1 255.255.255.0
!
interface Serial0
ip address 172.16.250.2 255.255.255.0
!
interface Serial1
ip address 172.16.252.1 255.255.255.0
...

router rip
network 172.16.0.0


hostname Ames
...
interface Ethernet0
ip address 172.16.100.1 255.255.255.0
```

```
!
interface Serial0
ip address 172.16.252.2 255.255.255.0
!
interface Serial1
ip address 172.16.251.2 255.255.255.0
...

router rip
network 172.16.0.0
```

Notice that all that is required of a network administrator to start RIP on a router is to issue the following command:

```
router rip
```

in global configuration mode and to list the networks that will be participating in the RIP process:

```
network 172.16.0.0
```

What does it mean to list the network numbers participating in RIP?

1. Router *NewYork* will include directly connected `172.16.0.0` subnets in its updates to neighboring routers. For example, `172.16.1.0` will now be included in updates to the routers *Chicago* and *Ames*.
2. *NewYork* will receive and process RIP updates on its `172.16.0.0` interfaces from other routers running RIP. For example, *NewYork* will receive RIP updates from *Chicago* and *Ames*.
3. By exclusion, network `192.168.1.0`, connected to *NewYork*, will not be advertised to *Chicago* or *Ames*, and *NewYork* will not process any RIP updates received on *Ethernet0* (if there is another router on that segment).

Next, let's verify that all the routers are seeing all the `172.16.0.0` subnets:

```
NewYork>sh 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

Gateway of last resort is not set

C       192.168.1.0 is directly connected, Ethernet1
     172.16.0.0/16 is subnetted, 6 subnets
C       172.16.1.0 is directly connected, Ethernet0
C       172.16.250.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
R       172.16.50.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
R       172.16.100.0 [120/1] via 172.16.251.2, 0:00:19, Serial1
R       172.16.252.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
                     [120/1] via 172.16.251.2, 0:00:19, Serial1
```

```
Chicago>sh 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

Gateway of last resort is not set

     172.16.0.0/16 is subnetted, 6 subnets
C       172.16.50.0 is directly connected, Ethernet0
C       172.16.250.0 is directly connected, Serial0
C       172.16.252.0 is directly connected, Serial1
R       172.16.1.0 [120/1] via 172.16.250.1, 0:00:01, Serial0
R       172.16.100.0 [120/1] via 172.16.252.2, 0:00:10, Serial1
R       172.16.251.0 [120/1] via 172.16.250.1, 0:00:01, Serial0
                     [120/1] via 172.16.252.2, 0:00:10, Serial1


Ames>sh 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

Gateway of last resort is not set

1      172.16.0.0/16 is subnetted, 6 subnets
C       172.16.100.0 is directly connected, Ethernet0
C       172.16.252.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
R       172.16.50.0 [120/1] via 172.16.252.1, 0:00:21, Serial0
R       172.16.1.0 [120/1] via 172.16.251.1, 0:00:09, Serial1
R       172.16.250.0 [120/1] via 172.16.252.1, 0:00:21, Serial0
                     [120/1] via 172.16.251.1, 0:00:09, Serial1
```

The left margin in the output of the routing tables shows how the route was derived. "C" indicates a directly connected network; "R" indicates RIP. Further note that there is some indentation in the output. The subnets of `172.16.0.0` are indented under line 1, which gives us the number of subnets (6) in `172.16.0.0` and the subnet mask that is associated with this network (`/16`). The routing table provides this information for every major network number it knows, indenting the subnets below the major network number.

Configuring RIP is fairly straightforward. We'll examine how RIP works in more detail in the next section.

How RIP Finds Shortest Paths

All DV protocols essentially operate the same way: routers exchange routing updates with neighboring (directly connected) routers; the routing updates contain a list of

network numbers along with the distance (metric, in routing terminology) to the networks. Each router chooses the shortest path to a destination network by comparing the distance (or metric) information it receives from its various neighbors. Let's look at this in more detail in the context of RIP.

Let's imagine that the network is cold-started—i.e., all three routers are powered up at the same time. The first thing that happens after IOS has finished loading is that the router checks for its connected interfaces and determines which ones are up. Next, these directly connected networks are installed in each router's routing table. So, right after IOS has been loaded and before any routing updates have been exchanged, the routing table would look like this:

```
NewYork>sh 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

Gateway of last resort is not set

C       171.16.1.0 is directly connected, Ethernet0
C       171.16.250.0 is directly connected, Serial0
C       171.16.251.0 is directly connected, Serial1


Chicago>sh 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

Gateway of last resort is not set

C       171.16.50.0 is directly connected, Ethernet0
C       171.16.250.0 is directly connected, Serial0
C       171.16.252.0 is directly connected, Serial1


Ames>sh 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

Gateway of last resort is not set

C       171.16.100.0 is directly connected, Ethernet0
C       171.16.250.0 is directly connected, Serial0
C       171.16.252.0 is directly connected, Serial1
```

The routers are now ready to update their neighbors with these routes.

RIP Update

RIP updates are encapsulated in UDP. The well-known port number for RIP updates is 520. The format of a RIP packet is shown in Figure 2-2.

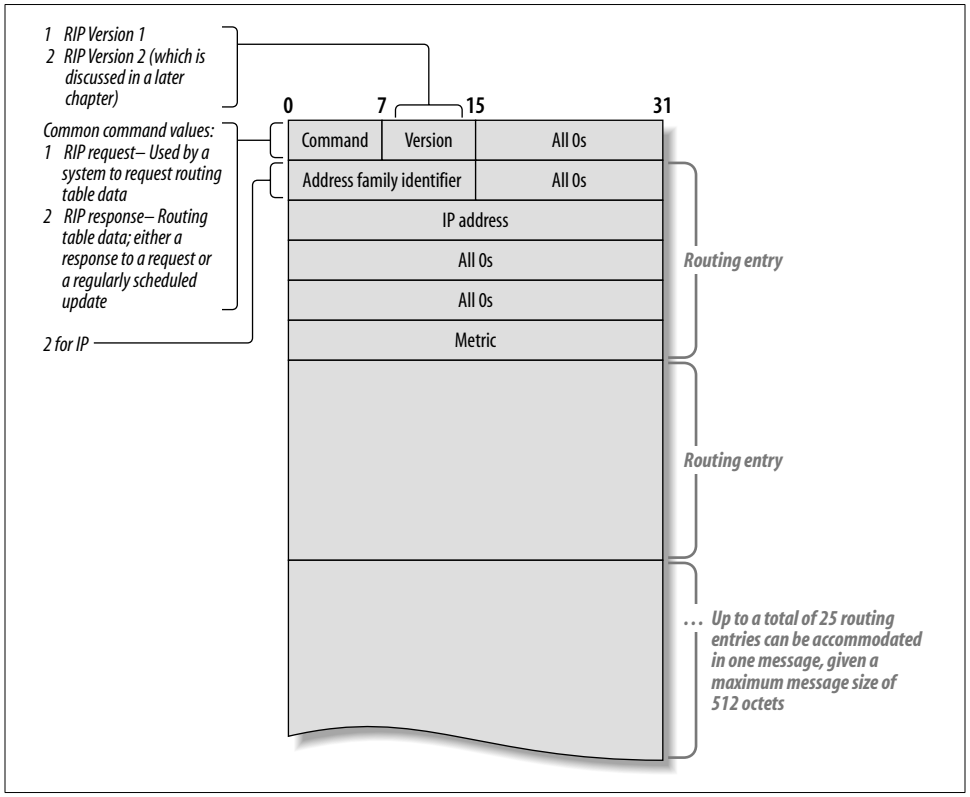


Figure 2-2. Format of RIP update packet

Note that RIP allows a station to request routes, so a machine that has just booted up can request the routing table from its neighbors instead of waiting until the next cycle of updates.

The destination IP address in RIP updates is `255.255.255.255`. The source IP address is the IP address of the interface from which the update is issued.

If you look closely at the update you will see that a key piece of information is missing: the subnet mask. Let's say that an update was received with the network number `172.31.0.0`. Should this be interpreted as `172.31.0.0/16` or `172.31.0.0/24` or `172.31.0.0/26` or ...? This question is addressed later, in the "Subnet Masks" section.

RIP Metric

The RIP metric is simply a measure of the number of hops to a destination network. `172.16.100.0`, which is directly connected to *Ames*, is zero hops from *Ames* but one hop from *NewYork* and *Chicago*. You can see RIP metrics in the routing table:

```
NewYork>sh ip route
...
Gateway of last resort is not set

C       192.168.1.0 is directly connected, Ethernet1
     172.16.0.0/16 is subnetted, 6 subnets
C       172.16.1.0 is directly connected, Ethernet0
C       172.16.250.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
R       172.16.50.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
R       172.16.100.0 [120/1] via 172.16.251.2, 0:00:19, Serial1
R       172.16.252.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
                     [120/1] via 172.16.251.2, 0:00:19, Serial1
```

This routing table shows the [distance/metric] tuple in bold. Every hop between two routers adds 1 to the RIP metric. Thus, *NewYork* sees the *Ames* segment (`172.16.100.0`) as one hop via the direct 56-kbps link and two hops via the T-1 to *Chicago*. *NewYork* will prefer the direct one-hop path to *Ames*.

The simplicity of the RIP metric is an asset in small, homogenous networks but a liability in networks with heterogeneous media. Consider the following comparison: the transmission delay for a 1,000-octet packet is 143 ms over a 56-kbps link and 5 ms over a T-1 link. Neglecting buffering and processing delays, two T-1 hops will cost 10 ms in comparison to 143 ms via the 56-kbps link. Thus, the two-hop T-1 path between *NewYork* and *Ames* is quicker; indeed, the designers of TraderMary's network may have put in the 56-kbps link only for backup purposes. However, RIP does not account for line speed, delay, or reliability. For this, we will look to the next DV protocol—IGRP.

Let's look at one more example of RIP metrics for TraderMary's network. Let's say that the T-1 link between *NewYork* and *Chicago* fails. As soon as *NewYork* (or *Chicago*) detects a failure in the link, all routes associated with that link are purged from the routing table, and, upon receipt of the next update, *NewYork* (*Chicago*) will learn the routes to *Chicago* (*NewYork*) via *Ames*. *NewYork*'s routing table would look like this:

```
NewYork>sh ip route
...
Gateway of last resort is not set

C       192.168.1.0 is directly connected, Ethernet1
     172.16.0.0/16 is subnetted, 6 subnets
C       172.16.1.0 is directly connected, Ethernet0
C       172.16.251.0 is directly connected, Serial1
```

```
R       172.16.50.0 [120/2] via 172.16.251.2, 0:00:23, Serial1
R       172.16.100.0 [120/1] via 172.16.251.2, 0:00:23, Serial1
R       172.16.252.0 [120/1] via 172.16.251.2, 0:00:23, Serial1
```

As we discussed in the previous chapter, the distance value associated with RIP is 120. Note that directly connected routes do not show a distance or metric value. Directly connected routes have a distance value of 0 and thus show the most preferred route to a destination, no matter how low the metric value of a route to the same network may be through another routing source (such as RIP).

The RIP metrics we saw in the previous examples were 1 or 2. It turns out that a RIP metric of 16 signals infinity (or unreachability). Why is it necessary to choose a maximum value for the RIP metric? Without a maximum hop count, a route can propagate indefinitely during certain failure scenarios, resulting in indefinitely long convergence times. This is discussed further in the "Convergence" section under "Counting to infinity."

Processing RIP Updates

The following rules summarize the steps a router takes when it receives a RIP update:

1. If the destination network number is unknown to the router, install the route using the source IP address of the update (provided the hop count is less than 16).
2. If the destination network number is known to the router but the update contains a smaller metric, modify the routing table entry with the new next hop and metric.
3. If the destination network number is known to the router but the update contains a larger metric, ignore the update.
4. If the destination network number is known to the router and the update contains a higher metric that is from the same next hop as in the table, update the metric.
5. If the destination network number is known to the router and the update contains the same metric from a different next hop, RFC 1058 calls for this update to be ignored, in general. However, Cisco differs from the standard here and installs up to four parallel paths to the same destination. These parallel paths are then used for load balancing.

Thus, when the first update from *Ames* reaches *NewYork* with the network `172.16.100.0`, *NewYork* installs the route with a hop count of 1 using rule 1. *NewYork* will also receive `172.16.100.0` in a subsequent update from *Chicago* (after *Chicago* itself has learned the route from *Ames*), but *NewYork* will discard this route because of rule 3.

Steady State

It is important for you as the network administrator to be familiar with the state of the network during normal conditions. Deviations from this state will be your clue to troubleshooting the network during times of network outage.

The following output will show you the values of the RIP timers. Note that RIP updates are sent every 30 seconds and the next update is due in 24 seconds, which means that an update was issued about 6 seconds ago. We will discuss the invalid, hold-down, and flush timers later, in the "Convergence" section.

```
NewYork>sh ip protocol

Routing Protocol is "rip"
Sending updates every 30 seconds, next due in 24 seconds
Invalid after 90 seconds, hold down 90, flushed after 180
```

One key area to look at in the routing table is the timer values. The format Cisco uses for timers is *hh:mm:ss* (hours:minutes:seconds). You would expect the time against each route to be between 0 and 30 seconds. If a route was received more than 30 seconds ago, that indicates a problem in the network. You should begin by checking to see if the next hop for the route is reachable. As an example, in line 1, `172.16.50.0` was learned 11 seconds ago from `172.16.250.2` (on *Serial0*).

```
NewYork>sh ip route
...
Gateway of last resort is not set

C       192.168.1.0 is directly connected, Ethernet1
     172.16.0.0/16 is subnetted, 6 subnets
C       172.16.1.9 is directly connected, Ethernet0
C       172.16.250.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
2 R     172.16.50.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
R       172.16.100.0 [120/1] via 172.16.251.2, 0:00:19, Serial1
R       172.16.252.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
                     [120/1] via 172.16.251.2, 0:00:19, Serial1
```

Parallel Paths

There are two equal-cost paths to network `172.16.252.0` from *NewYork*—one advertised by *Ames* and the other by *Chicago*. *NewYork* will install both routes in its routing table:

```
NewYork>sh ip route
...
Gateway of last resort is not set

C       192.168.1.0 is directly connected, Ethernet1
     172.16.0.0/16 is subnetted, 6 subnets
C       172.16.1.9 is directly connected, Ethernet0
C       172.16.250.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
R       172.16.50.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
R       172.16.100.0 [120/1] via 172.16.251.2, 0:00:19, Serial1
R       172.16.252.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
                     [120/1] via 172.16.251.2, 0:00:19, Serial1
```

Both paths are utilized to forward packets. How is traffic split over the two links? The answer depends on the switching mode configured on the Cisco router. Two common switching modes are process switching and fast switching.

Process Switching

Process switching results in packet-by-packet load balancing—one packet travels out on *serial0* and the next packet travels out on *serial1*. Packet-by-packet load balancing is possible while process switching because in this switching mode the router examines its routing table for every packet it receives.

Process switching is configured as follows:

```
NewYork#-config#interface serial0
NewYork#-config-if#no ip route-cache
```

Packet switching is very CPU-intensive, as every packet causes a routing table lookup.

Fast Switching

In this mode, only the first packet for a given destination is looked up in the routing table, and, as this packet is forwarded, its next hop (say, *serial0*) is placed in a cache. Subsequent packets for the same destination are looked up in the cache, not in the routing table. This implies that all packets for this destination will follow the same path (*serial0*).

Now, if another packet arrives that matches the routing entry `204.148.185.192`, it will be cached with a next hop of *serial1*. Henceforth, all packets to this second destination will follow *serial1*.

Fast switching thus load-balances destination-by-destination (or session-by-session). Fast switching is configured as follows:

```
NewYork#-config#interface serial0
NewYork#-config-if#ip route-cache
```

In fast switching, the first packet for a new destination causes a routing table lookup and the generation of a new entry in the route cache. Subsequent packets consult the route cache but not the routing table.

Convergence

Changes—planned and unplanned—are normal in any network:

- A serial link breaks
- A new serial link is added to a network
- A router or hub loses power or malfunctions
- A new LAN segment is added to a network

All routers in the routing domain will not reflect these changes right away. This is because RIP routers rely on their direct neighbors for routing updates, which in turn rely on another set of neighbors. The routing process that is set into motion from the time of a network change (such as the failure of a link) until all routers correctly reflect the change is referred to as convergence. During convergence, routing connectivity between some parts of the network may be lost and, hence, an important question that is frequently asked is "How long will the network take to converge after such-and-such failure in the network?" The answer depends on a number of factors, including the network topology and the timers that have been defined for the routing protocol.

The following list defines the four timers that are key to the operation of any DV protocol, including RIP:

Update timer (default value: 30 seconds)
: After sending a routing update, RIP sets the update timer to 0. When the timer expires, RIP issues another routing update. Thus, RIP updates are sent every 30 seconds.

Invalid timer (default value: 180 seconds)
: Every time a router receives an update for a route, it sets the invalid timer to 0. The expiration of the invalid timer indicates that six consecutive updates were missed—at this time, the source of the routing information is considered suspect. Even though the route is declared invalid, packets are still forwarded to the next hop specified in the routing table. Note that prior to the expiration of the invalid timer RIP would process any updates received by updating the route's timers.

Hold-down timer (default value: 180 seconds)
: When the invalid timer expires, the route automatically enters the hold-down phase. During hold-down, all updates regarding the route are disregarded—it is assumed that the network may not have converged and that there may be bad routing information circulating in the network. The hold-down timer is started when the invalid timer expires. Thus, a route goes into hold-down state when the invalid timer expires. A route may also go into hold-down state when an update is received indicating that the route has become unreachable—this is discussed further later in this section.

Flush timer (default value: 240 seconds)
: The flush timer is set to 0 when an update is received. When the flush timer expires, the route is removed from the routing table and the router is ready to receive an update with this route. Note that the flush timer overrides the hold-down timer.

Let's consider Figure 2-3. Here is a snapshot of *A*'s routing table (when all entities are up):

```
A>sh ip route
...

C       192.168.1.0 is directly connected, Ethernet1
     172.17.0.0/16 is subnetted, 6 subnets
```

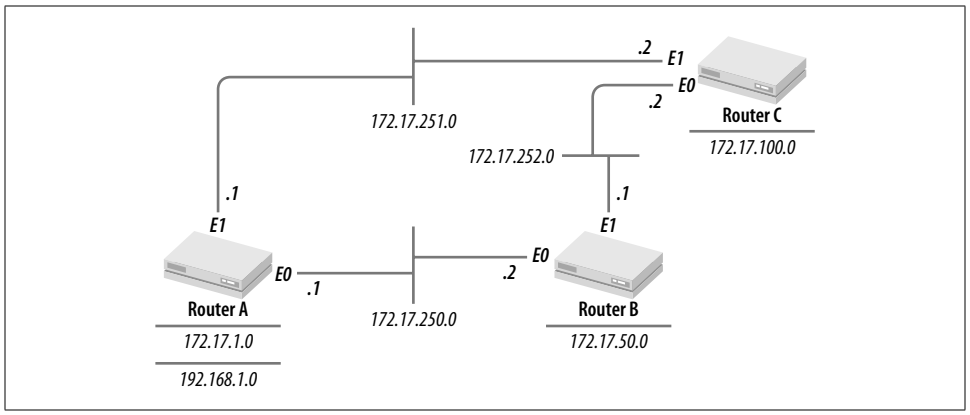


Figure 2-3. Three routers connected using Ethernet segments

```
C       172.17.1.9 is directly connected, Ethernet0
C       172.17.250.0 is directly connected, Ethernet1
C       172.17.251.0 is directly connected, Ethernet2
R       172.17.50.0 [120/1] via 172.17.250.2, 0:00:11, Ethernet1
R       172.17.100.0 [120/1] via 172.17.251.2, 0:00:19, Ethernet2
R       172.17.252.0 [120/1] via 172.17.250.2, 0:00:11, Ethernet1
                     [120/1] via 172.17.251.2, 0:00:19, Ethernet2
```

This table shows that 11 seconds ago *A* received an update for `172.17.50.0` from `172.17.250.2` (*B*). The update and invalid timers for a route are reset (set to 0) every time a valid update is received for the route. At the moment this routing-table snapshot was taken, *A*'s invalid timer for `172.16.50.0` and *B*'s update timer for `172.16.50.0` would both be 11 seconds.

Let's say that at this very time, *B* was disconnected from its LAN attachment to *A*. *A* would now stop receiving updates from *B*. 30 seconds after the cut, the routing table would look like this:

```
A>sh ip route
...

C       192.168.1.0 is directly connected, Ethernet1
     172.17.0.0/16 is subnetted, 6 subnets
C       172.17.1.9 is directly connected, Ethernet0
C       172.17.250.0 is directly connected, Serial0
C       172.17.251.0 is directly connected, Serial1
R       172.17.50.0 [120/1] via 172.17.250.2, 0:00:41, Serial0
R       172.17.100.0 [120/1] via 172.17.251.2, 0:00:19, Serial1
R       172.17.252.0 [120/1] via 172.17.250.2, 0:00:41, Serial0
                     [120/1] via 172.17.251.2, 0:00:19, Serial1
```

The invalid timer for `172.16.50.0` is now at 41 seconds. *A* would still continue to forward traffic for `172.17.50.0` via *Ethernet0*. The assumption RIP makes is that an update was lost or damaged in transit from *B* to *A*, even though the route is still good. This assumption holds good until the invalid timer expires (180 seconds or 6

update intervals from the last update). Before the invalid timer expires, *A* will receive and process any updates received regarding `172.16.50.0`. Once the invalid timer expires, the route is placed in hold-down and subsequent updates about `172.16.0.0` are suppressed under the assumption that the route has gone bad and that bad routing information may be circulating in the network. The route will go into hold-down 180 seconds from the last update, or 169 seconds after the cut. At this time, the routing table would look like this:

```
A>sh ip route
...

C       192.168.1.0 is directly connected, Ethernet1
     172.17.0.0/16 is subnetted, 6 subnets
C       172.17.1.9 is directly connected, Ethernet0
C       172.17.250.0 is directly connected, Serial0
C       172.17.251.0 is directly connected, Serial1
R       172.17.50.0 is possibly down,
          routing via 172.17.250.2, Serial0
R       172.17.100.0 [120/1] via 172.17.251.2, 0:00:19, Serial1
R       172.17.252.0 [120/1] is possibly down,
          routing via 172.16.250.2, Ethernet1
                    [120/1] via 172.17.251.2, 0:00:19, Serial1
```

The route remains in hold-down until the hold-down timer expires or until the route is flushed, whichever happens first. Using default timers, the flush timer would go off first, 229 seconds after the cut. Router *A* would then learn the route to `172.17.50.0` when the next update arrived from *C*, which could be between 0 and 30 seconds after the route has been flushed, or 229 to 259 seconds from the cut.

The events just described are illustrated in Figure 2-4.

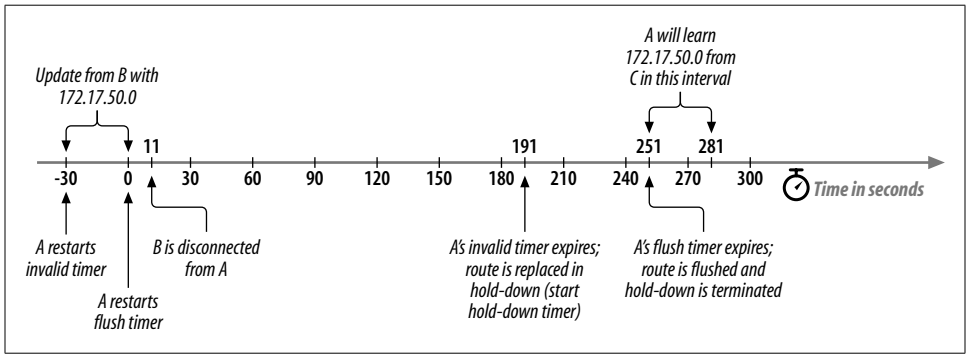


Figure 2-4. Route convergence after a failure

Speeding Up Convergence

When a router detects that an interface is down, it immediately flushes all routes it knows via that interface. This speeds up convergence, avoiding the invalid, hold-down, and flush timers.

Can you now guess the reason why the case study used earlier (routers *A*, *B*, and *C* connected via Ethernet segments) differs slightly from TraderMary's network in New York, Chicago, and Ames?

We couldn't illustrate the details of the invalid, hold-down, and flush timers in TraderMary's network because if a serial link is detected in the down state, all routes that point through that interface are immediately flushed from the routing table. In our case study, we were able to pull *B* off its Ethernet connection to *A* while keeping *A* up on all its interfaces.

Split horizon

Consider a simple network with two routers connected to each other (Figure 2-5).

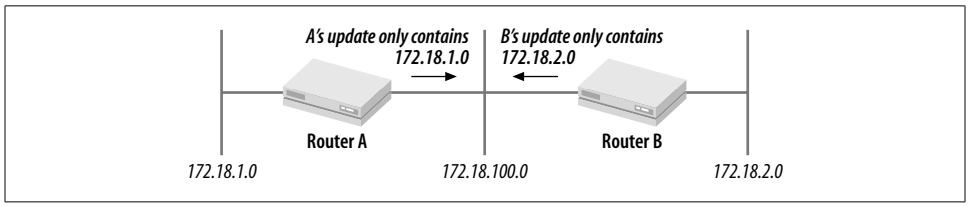


Figure 2-5. Split horizon

Let's say that router *A* lost its connection to `172.18.1.0`, but before it could update *B* about this change, *B* sent *A* its full routing table, including `172.18.1.0` at one hop. Router *A* now assumes that *B* has a connection to `172.18.1.0` at one hop, so *A* installs a route to `172.18.1.0` at two hops via *B*. *A*'s next update to *B* announces `172.18.1.0` at two hops, so *B* adjusts its route `172.18.1.0` to three hops via *A*! This cycle continues until the route metric reaches 16, at which stage the route update is discarded.

Split horizon solves this problem by proposing a simple solution: when a router sends an update through an interface, it does not include in its update any routes that it learned via that interface. Using this rule, the only network that *A* would send to *B* in its update would be `172.18.1.0`, and the only network that *B* would send to *A* would be `172.18.2.0`. *B* would never send `172.18.1.0` to *A*, so the previously described loop would be impossible.

Counting to infinity

Split horizon works well for two routers directly connected to each other. However, consider the following network (shown in Figure 2-6).

Let's say that router *A* stopped advertising network *X* to its neighbors *B* and *E*. Routers *B*, *D*, and *E* will finally purge the route to *X*, but router *C* may still advertise *X* to *D* (without violating split horizon). *D*, in turn, will advertise *X* to *E*, and *E* will advertise *X* to *A*. Thus, the router (*C*) that did not purge *X* from its table can propagate a bad route.

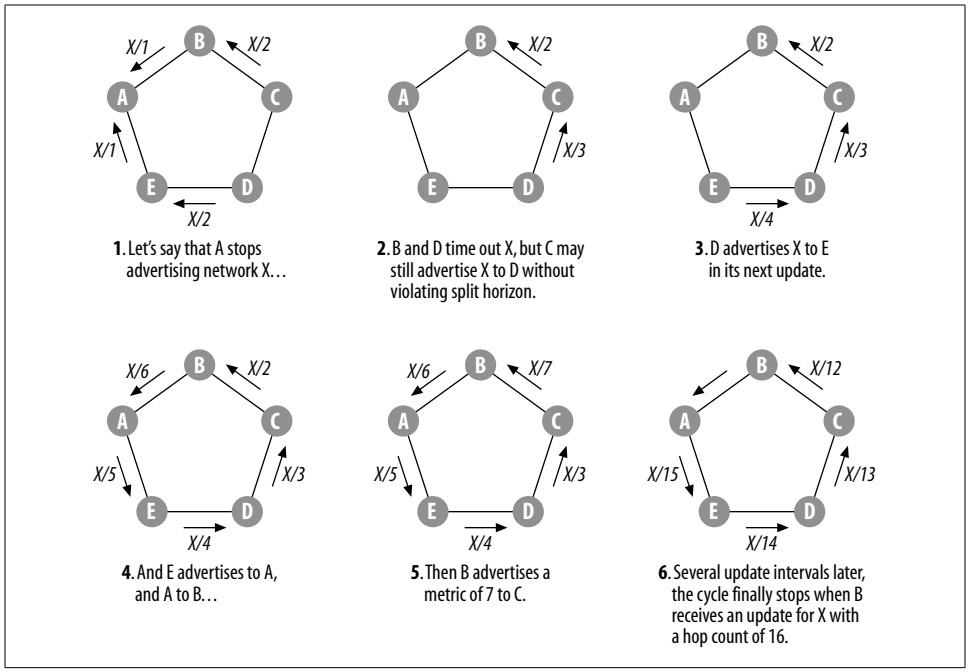


Figure 2-6. Counting to infinity

This problem is solved by equating a hop count of 16 to infinity and hence disregarding any advertisement for a route with this metric.

In Figure 2-6, when *B* finally receives an advertisement for *X* with a metric of 16, it will consider *X* to be unreachable and will disregard the advertisement. The choice of 16 as infinity limits RIP networks to a maximum diameter of 15 hops between nodes. Note that the choice of 16 as infinity is a compromise between convergence time and network diameter—if a higher number were chosen, the network would take longer to converge after a failure; if a lower number were chosen, the network would converge faster but the maximum possible diameter of a RIP network would be smaller.

Triggered updates

When a router detects a change in the metric for a route and sends an update to its neighbors right away (without waiting for its next update cycle), the update is referred to as a *triggered update*. The triggered update speeds convergence between two neighbors by as much as 30 seconds. A triggered update does not include the entire routing table, but only the route that has changed.

Poison reverse

When a router detects that a link is down, its next update for that route will contain a metric of 16. This is called *poisoning* the route. Downstream routers that receive

this update will immediately place the route in hold-down (without going through the invalid period).

Poison reverse and triggered updates can be combined. When a router detects that a link has been lost or the metric for a route has changed to 16, it will immediately issue a poison reverse with triggered update to all its neighbors.

Neighbors that receive unreachability information about a route via a poison reverse with triggered update will place the route in hold-down if their next hop is via the router issuing the poison reverse. The hold-down state ensures that bad information about the route (say from a neighbor that may have lost its copy of the triggered update or may have issued a regular update just before it received the triggered update) does not propagate in the network.

Triggered updates and hold-downs can handle the loss of a route, preventing bad routing information. Why, then, do we need the count-to-infinity limits? Triggered updates may be dropped, lost, or corrupted. Some routers may not ever receive the unreachability information and may inject a path for a route into the network even when that path has been lost. Count to infinity would take care of these situations.

Setting timers

The value of RIP timers on a Cisco router can be seen in the following example:

```
Chicago>sh ip protocol

Routing Protocol is "rip"
Sending updates every 30 seconds, next due in 24 seconds
Invalid after 90 seconds, hold down 90, flushed after 180
```

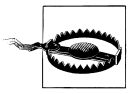
These timers could be modified to allow faster convergence. The following command:

```
timers basic 10 25 30 40
```

would send RIP updates every 10 seconds instead of every 30 seconds. The other three timers specify the invalid, hold-down, and flush timers, respectively. These timers can be configured as follows:

```
NewYork#config
NewYork-config#router rip
NewYork-config#timers basic 10 25 30 40
```

However, RIP timers should not be modified without a detailed understanding of how RIP works. Potential problems with decreasing the timer values are that updates will be issued more frequently and can cause congestion on low-bandwidth networks, and that congestion in the network is more likely to cause routes to go into hold-down; this, in turn, can cause route flapping.



Do not modify RIP timers unless absolutely necessary. If you modify RIP timers, make sure that all routers have the same timers.

If an interface on a router goes down, the router sends a RIP request out to the other, up interfaces. This speeds up convergence if any of the other neighbors can reach the destinations that were missed in the first request.

Subnet Masks

Looking closely at Figure 2-2, we see that there is no field for subnet masks in RIP. Let's say that router *SantaFe* received an update with the following routes in the IP address field:

```
192.100.1.48
192.100.1.64
192.100.2.0
10.0.0.0
```

And let's say that *SantaFe* has the following configuration:

```
hostname SantaFe
!
interface Ethernet 0
ip address 192.100.1.17 255.255.255.240
!
interface Ethernet 1
ip address 192.100.1.33 255.255.255.240
!
router rip
network 192.100.1.0
network 192.100.2.0
```

How would the router associate subnet masks with these routes?

- If the router has an interface on a network number received in an update, it would associate the same mask with the update as it does with its own interface. Consequently, RIP does not permit Variable Length Subnet Masks (VLSM).
- If the router does not have an interface on the network number received in an update, it would assume a natural mask for the network number.

SantaFe's routing table would look like this:

```
SantaFe>sh ip route
...
Gateway of last resort is not set

R       10.0.0.0 [120/1] via 192.100.1.18, 0:00:11, Ethernet0
R       192.100.2.0 [120/1] via 192.100.1.18, 0:00:11, Ethernet0
     192.100.1.0/16 is subnetted, 4 subnets
C       192.100.1.16 is directly connected, Ethernet0
C       192.100.1.32 is directly connected, Ethernet1
R       192.100.1.48 [120/1] via 192.100.1.18, 0:00:11, Ethernet0
R       192.100.1.64 [120/1] via 192.100.1.18, 0:00:11, Ethernet0
```

SantaFe represents `192.100.1.48` and `192.100.1.64` with a 28-bit mask even though the subnet mask was not conveyed in the RIP update. *SantaFe* was able to deduce the

28-bit mask because it has direct interfaces on 192.100.1.0 networks. This assumption is key to why RIP does not support VLSM.

SantaFe represents 192.100.2.0 and 10.0.0.0 with their natural 24-bit and 8-bit masks, respectively, because it has no interfaces on those networks. Chapter 5 covers RIP-2, an extension of RIP that supports VLSM.

Route Summarization

Consider the router *Phoenix*, which connects to *SantaFe* and sends the RIP updates shown earlier:

```
192.100.1.48
192.100.1.64
192.100.2.0
10.0.0.0
```

Phoenix may have been configured as follows (see Figure 2-8, later in this chapter):

```
hostname Phoenix
ip subnet-zero
!
interface Ethernet 0
ip address 192.100.1.18 255.255.255.240
!
interface Ethernet 1
ip address 192.100.1.49 255.255.255.240
!
interface Ethernet 2
ip address 192.100.1.65 255.255.255.240
!
interface Ethernet 3
ip address 192.100.2.1 255.255.255.240
!
interface Ethernet 4
ip address 192.100.2.17 255.255.255.240
!
interface Ethernet 5
ip address 10.1.0.1 255.255.0.0
!
interface Ethernet 6
ip address 10.2.0.1 255.255.0.0
!
router rip
network 192.100.1.0
network 192.100.2.0
network 10.0.0.0
```

Phoenix did not send detailed routes for 192.100.2.0 or 10.0.0.0 when advertising to *SantaFe* because *Phoenix* summarized those routes. As I stated earlier, since *Phoenix* did not have interfaces on those networks, it couldn't have made sense of those routes anyway.

Default Route

A routing table need not contain all routes in the network to reach all destinations. This simplification is arrived at through the use of a *default route*. When a router does not have an explicit route to a destination IP address, it looks to see if it has a default route in its routing table and, if so, forwards packets for this destination via the default route.

In RIP, the default route is represented as the IP address `0.0.0.0`. This is convenient because `0.0.0.0` cannot be confused with any Class A, B, or C IP address.

One situation in which default routes can be employed in an intranet is in a core network that has branch offices hanging off it (Figure 2-7).

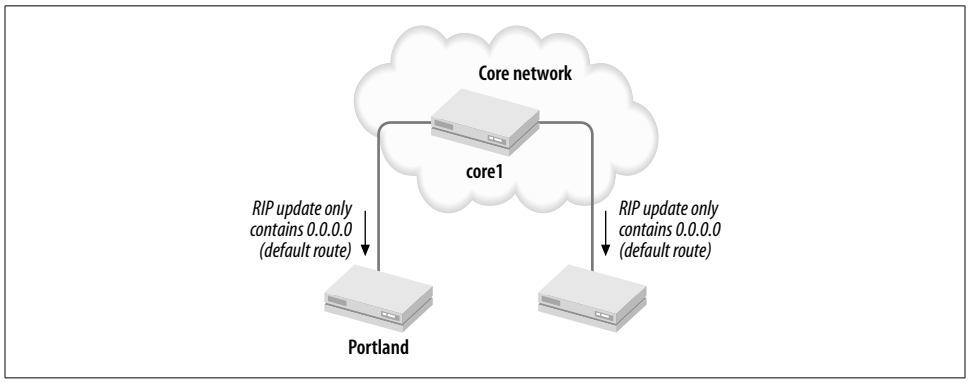


Figure 2-7. Branch offices only need a default route

Consider the topology of this figure. Since the branch offices have only one connection (to the core), all routes to the core network and to other branches can be replaced with a single default route pointing toward the core network. This implies that the size of the routing table in the branch offices is just the number of directly connected networks plus the default route.

So, router *Portland* may be configured as follows:

```
hostname Portland
...
interface Ethernet 0
ip address 192.100.1.17 255.255.255.240
!
interface Serial 0
ip address 192.100.1.33 255.255.255.240
!
router rip
network 192.100.1.0
```

An examination of *Portland*'s routing table would show the following:

```
Portland>sh ip route
...
Gateway of last resort is not set

     192.100.1.0/28 is subnetted, 2 subnets
C       192.100.1.16 is directly connected, Ethernet0
C       192.100.1.32 is directly connected, Serial0
R    0.0.0.0 [120/1] via 192.199.1.34, 0:00:21, Serial0
```

The default route may be sourced from router *core1* as follows:

```
hostname core1
...
interface Serial 0
ip address 192.100.1.34 255.255.255.240
!
router rip
network 192.100.1.0
!
ip route 0.0.0.0 0.0.0.0 null0
```

Note that the default route `0.0.0.0` is automatically carried by RIP—it is not listed in a network number statement under *router rip*.

The advantage of using a default in place of hundreds or thousands of more specific routes is obvious—network bandwidth and router CPU are not tied up in routing updates. The disadvantage of using a default is that packets for destinations that are down or not even defined in the network are still forwarded to the core network.

Default routes are tremendously useful in Internet connectivity—where all (thousands and thousands of) Internet routes may be represented by a single default route.

Yet another use of default routes is in maintaining reachability between a routing domain running RIP and another routing domain with VLSM. Since VLSM cannot be imported into RIP, a default route pointing to the second domain may be defined in the RIP network.

Routes to hosts

Some host machines listen to RIP updates in "quiet" or "silent" mode (Figure 2-8). These hosts do not respond to requests for RIP routes or issue regular RIP updates. Listening to RIP provides redundancy to the hosts in a scenario in which multiple routers are connected to a segment. If the routers have similar routing tables, it may make sense to send only the default route (`0.0.0.0`) to hosts.

Fine-Tuning RIP

We saw in the section on RIP metrics that the preferred path between *NewYork* and *Ames* would be the two-hop path via *Chicago* rather than the one-hop 56-kbps path

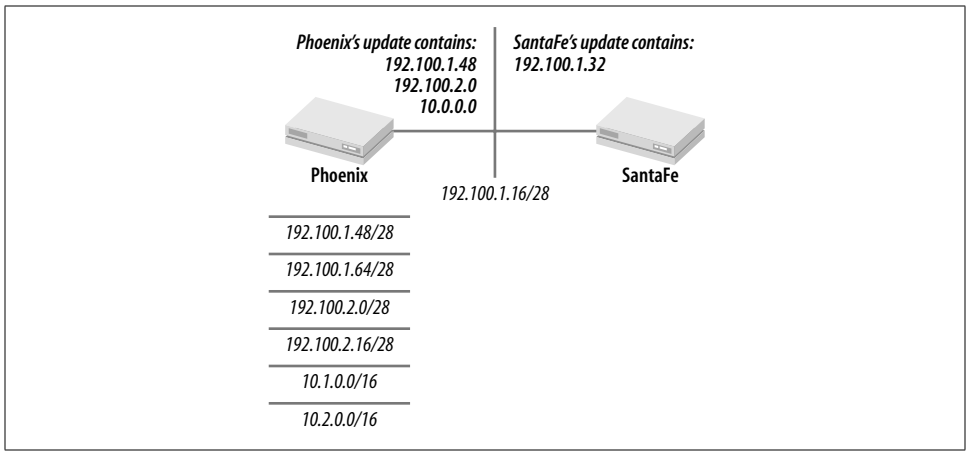


Figure 2-8. RIP routes to hosts

that RIP selects. The RIP metrics can be manipulated to disfavor the one-hop path through the use of offset lists:

```
hostname NewYork
...
router rip
network 172.16.0.0
offset-list 10 in 2 serial1
...
access-list 10 permit 172.16.100.0 0.0.0.0

hostname Chicago
...
router rip
network 172.16.0.0

Ames#config terminal
router rip
network 172.16.0.0
offset-list 20 in 2 serial1
...
access-list 20 permit 172.16.1.0 0.0.0.0
```

NewYork adds 2 to the metric for the routes specified in access list 10 when learned via *serial1*, and *Ames* adds 2 to the metric for the routes specified in access list 20 when learned via *serial1*. The direct route over the 56-kbps link thus has a metric of 3, and the route via *Chicago* has a metric of 2. The new routing tables look like this:

```
NewYork>sh ip route
...
Gateway of last resort is not set

C       192.168.1.0 is directly connected, Ethernet1
     172.16.0.0/16 is subnetted, 6 subnets
C       172.16.1.0 is directly connected, Ethernet0
```

```
C       172.16.250.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
R       172.16.50.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
R       172.16.100.0 [120/2] via 172.16.250.2, 0:00:19, Serial0
R       172.16.252.0 [120/1] via 172.16.250.2, 0:00:11, Serial0
                     [120/1] via 172.16.251.2, 0:00:19, Serial1

Ames>sh ip route
...
Gateway of last resort is not set

     172.16.0.0/16 is subnetted, 6 subnets
C       172.16.100.0 is directly connected, Ethernet0
C       172.16.252.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
R       172.16.50.0 [120/1] via 172.16.252.1, 0:00:21, Serial0
R       172.16.1.0 [120/2] via 172.16.251.1, 0:00:09, Serial1
R       172.16.250.0 [120/1] via 172.16.252.1, 0:00:21, Serial0
                     [120/1] via 172.16.251.1, 0:00:09, Serial1
```

The syntax for offset lists is as follows:

```
offset-list {access-list} {in | out} offset [type number]
```

The offset list specifies the offset to add to the RIP metric on routes of interface *type* (Ethernet, serial, etc.) and *number* (interface number) that are being learned (*in*) or advertised (*out*).

An offset list can also be applied to default routes. Thus, in Figure 2-7, let's consider the scenario where *Portland* is given a second connection to a backup router *core2*. *core2* may originate a default with a higher metric:

```
hostname core2
...
interface Serial 0
ip address 192.100.2.34 255.255.255.240
!
router rip
network 192.100.2.0
offset-list 30 out 3 serial0
!
ip route 0.0.0.0 0.0.0.0 null0
!
access-list 30 permit 0.0.0.0 0.0.0.0
```

Portland would prefer the default via *core1* because the metric from *core1* would be lower by 3. *Portland* would use the default from *core2* if *core1* or the link to *core1* went down.

Summing Up

RIP is a relatively simple protocol, easy to configure and very reliable. The robustness of RIP is evident from the fact that various implementations of RIP differ in

details and yet work well together. A standard for RIP wasn't put forth until 1988 (by Charles Hedrick, in RFC 1058). Small, homogeneous networks are a good match for RIP. However, as networks grow, other routing protocols may look more attractive for several reasons:

- The RIP metric does not account for link bandwidth or delay.
- The exchange of full routing updates every 30 seconds does not scale for large networks—the overhead of generating and processing all routes can be high.
- RIP convergence times can be too long.
- Subnet mask information is not exchanged in RIP updates, so Variable Length Subnet Masks are not supported.
- The RIP metric restricts the network diameter to 15 hops.