In this chapter:

CHAPTER 4

Enhanced Interior Gateway Routing Protocol (EIGRP)

The Enhanced Interior Gateway Routing Protocol (EIGRP), referred to as an advanced Distance Vector protocol, offers radical improvements over IGRP. Traditional DV protocols such as RIP and IGRP exchange periodic routing updates with all their neighbors, saving the best distance (or metric) and the vector (or next hop) for each destination. EIGRP differs in that it saves not only the best (least-cost) route but all routes, allowing convergence to be much quicker. Further, EIGRP updates are sent only upon a network topology change; updates are not periodic.

Getting EIGRP running is not much more difficult than getting IGRP running, as we will see in the section "Getting EIGRP Running."

Even though EIGRP offers radical improvements over IGRP, there are similarities between the protocols. Like IGRP, EIGRP bases its metric on bandwidth, delay, reliability, load, and MTU (see the "EIGRP Metric" section).

The fast convergence feature in EIGRP is due to the Diffusing Update Algorithm (DUAL), discussed in "How EIGRP Works."

EIGRP updates carry subnet mask information. This allows EIGRP to summarize routes on arbitrary bit boundaries, support classless route lookups, and allow the support of Variable Length Subnet Masks (VLSM). This is discussed in "Variable Length Subnet Masks" and "Route Summarization."

Setting up default routes in EIGRP is discussed in "Default Routes."

Troubleshooting EIGRP can be tricky. This chapter ends with some troubleshooting tips in "Troubleshooting EIGRP."

EIGRP is a Cisco proprietary protocol; other router vendors do not support EIGRP. Keep this in mind if you are planning a multivendor router environment.

This chapter focuses on EIGRP's enhancements over IGRP: the use of DUAL; and the use of subnet masks in updates, which in turn allow VLSM and route summarization at arbitrary bit boundaries. This chapter does not cover router metrics in detail or the concept of parallel paths. Those concepts have not changed much in EIGRP. I assume that the reader is familiar with IGRP.

Getting EIGRP Running

TraderMary's network, shown in Figure 4-1, can be configured to run EIGRP as follows.

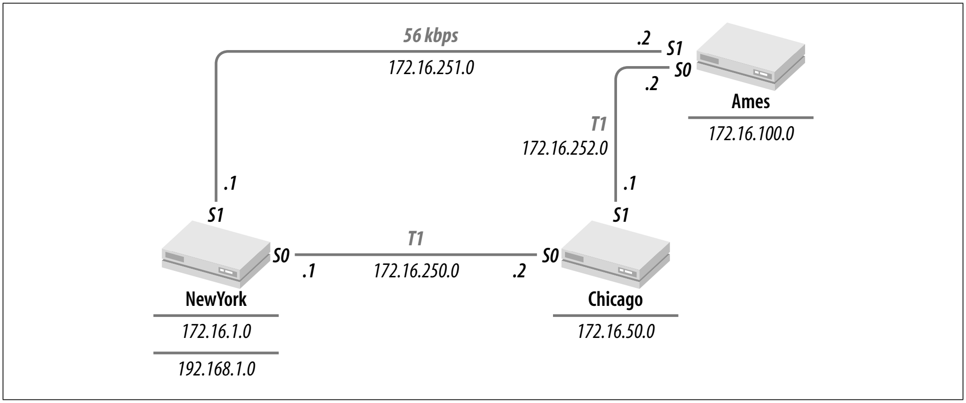

Figure 4-1. TraderMary's network

Just like RIP and IGRP, EIGRP is a distributed protocol that needs to be configured on every router in the network:

```
hostname NewYork
...
interface Ethernet0
ip address 172.16.1.1 255.255.255.0
!
interface Ethernet1
ip address 192.168.1.1 255.255.255.0
!
interface Serial0
description New York to Chicago link
ip address 172.16.250.1 255.255.255.0
!
interface Serial1
description New York to Ames link
bandwidth 56
ip address 172.16.251.1 255.255.255.0
```

```
...
router eigrp 10
network 172.16.0.0


hostname Chicago
...
interface Ethernet0
ip address 172.16.50.1 255.255.255.0
!
interface Serial0
description Chicago to New York link
ip address 172.16.250.2 255.255.255.0
!
interface Serial1
description Chicago to Ames link
ip address 172.16.252.1 255.255.255.0
...

router eigrp 10
network 172.16.0.0


hostname Ames
...
interface Ethernet0
ip address 172.16.100.1 255.255.255.0
!
interface Serial0
description Ames to Chicago link
ip address 172.16.252.2 255.255.255.0
!
interface Serial1
description Ames to New York link
bandwidth 56
ip address 172.16.251.2 255.255.255.0
...

router eigrp 10
network 172.16.0.0
```

The syntax of the EIGRP command is:

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

in global configuration mode. The networks that will be participating in the EIGRP process are then listed:

```
network 172.16.0.0
```

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

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 EIGRP updates on its 172.16.0.0 interfaces from other routers running EIGRP 10. For example, *NewYork* will receive EIGRP 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 EIGRP updates received on *Ethernet0* (if there is another router on that segment).

The routing tables for *NewYork*, *Chicago*, and *Ames* will show all 172.16.0.0 subnets. Here is *NewYork*'s table:

```
    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

1        172.16.0.0/24 is subnetted, 6 subnets
    D       172.16.252.0 [90/2681856] via 172.16.250.2, 00:18:54, Ethernet0/0
    C       172.16.250.0 is directly connected, Ethernet0/0
    C       172.16.251.0 is directly connected, Ethernet0/1
    D       172.16.50.0 [90/2195456] via 172.16.250.2, 00:18:54, Ethernet0/0
    C       172.16.1.0 is directly connected, Loopback0
    D       172.16.100.0 [90/2707456] via 172.16.250.2, 00:18:54, Ethernet0/0
    C    192.168.1.0/24 is directly connected, Loopback1
```

The EIGRP-derived routes in this table are labeled with a "D" in the left margin. Note that the routing table provides summary information (as in line 1). Line 1 contains subnet mask information (24 bits, or 255.255.255.0) and the number of subnets in 172.16.0.0 (6).

In addition to the routing table, EIGRP builds another table called the *topology table*:

```
    NewYork#sh ip eigrp topology
    IP-EIGRP Topology Table for process 10

    Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
           r - Reply status

    P 172.16.252.0/24, 1 successors, FD is 2681856
             via 172.16.250.2 (2681856/2169856), Serial0
             via 172.16.251.2 (46738176/2169856), Serial1
    P 172.16.250.0/24, 1 successors, FD is 2169856
             via Connected, Serial0
    P 172.16.251.0/24, 1 successors, FD is 46226176
             via Connected, Serial1
    P 172.16.50.0/24, 1 successors, FD is 2195456
             via 172.16.250.2 (2195456/281600), Serial0
    P 172.16.1.0/24, 1 successors, FD is 128256
             via Connected, Ethernet0
2   P 172.16.100.0/24, 1 successors, FD is 2707456
```

```
3          via 172.16.250.2 (2707456/2195456), Serial0
4          via 172.16.251.2 (46251776/281600), Serial1
```

This topology table shows two entries for *Ames*'s subnet, `172.16.100.0` (line 2). Only the lower-cost route (line 3) is installed in the routing table, but the second entry in the topology table (line 4) allows *NewYork* to quickly converge on the less preferred path if the primary path fails.

Note that network `192.168.1.0`, defined on *NewYork* interface *Ethernet1*, did not appear in the routing tables of *Chicago* and *Ames*. To be propagated, `192.168.1.0` would have to be defined in a network statement under the EIGRP configuration on *NewYork*:

```
hostname NewYork
...
router eigrp 10
network 172.16.0.0
network 192.168.1.0
```

Each EIGRP process is identified by an autonomous system (AS) number, just like IGRP processes. Routers with the *same* AS numbers will exchange routing information with each other, resulting in a *routing domain*. Routers with dissimilar AS numbers will not exchange any routing information by default. However, routes from one routing domain can be leaked into another domain through the redistribution commands—this is covered in Chapter 8.

Compare the routing table in this section with the corresponding table for IGRP in Chapter 3. The essential contents are identical: the same routes with the same next hops. However, the route metrics look much bigger and the route update times are very high. IGRP routes would have timed out a while ago.

EIGRP metrics are essentially derived from IGRP metrics. The following section provides a quick summary.

EIGRP Metric

The EIGRP composite metric is computed exactly as the IGRP metric is and then multiplied by 256. Thus, the default expression for the EIGRP composite metric is:

$$Metric = [BandW + Delay] \times 256$$

where *BandW* and *Delay* are computed exactly as for IGRP (see the section "IGRP Metric" in Chapter 3). In summary, *BandW* is computed by taking the smallest bandwidth (expressed in kbits/s) from all outgoing interfaces to the destination (including the destination) and dividing 10,000,000 by this number (the smallest bandwidth), and *Delay* is the sum of all the delay values to the destination network (expressed in tens of microseconds).

Further, note that the total delay (line 6), minimum bandwidth (line 6), reliability (line 7), minimum MTU (line 7), and load (line 8) for a path, which are used to compute the composite metric (line 5), are shown as output of the *show ip route destination-network-number* command:

```
NewYork#sh ip route 172.16.50.0
Routing entry for 172.16.50.0 255.255.255.0
  Known via "eigrp 10", distance 90, metric 2195456, type internal
  Redistributing via eigrp 10
  Last update from 172.16.250.2 on Serial0, 00:00:21 ago
  Routing Descriptor Blocks:
  * 172.16.50.0, from 172.16.250.2, 00:00:21 ago, via Serial0
5       Route metric is 2195456, traffic share count is 1
6       Total delay is 21000 microseconds, minimum bandwidth is 1544 Kbit
7       Reliability 255/255, minimum MTU 1500 bytes
8       Loading 1/255, Hops 1
```

Converting route metrics between EIGRP and IGRP is very straightforward: EIGRP metrics are 256 times larger than IGRP metrics. This easy conversion becomes important when a network is running both IGRP and EIGRP, such as during a migration from IGRP to EIGRP.

Just like IGRP, EIGRP can be made to use load and reliability in its metric by modifying the parameters k1, k2, k3, k4, and k5 (see the "IGRP Metric" section in the previous chapter).

The constants k1, k2, k3, k4, and k5 can be modified with the following command:

```
metric weights tos k1 k2 k3 k4 k5
```

Cisco strongly recommends *not* modifying the k1, k2, k3, k4, and k5 values for EIGRP.

How EIGRP Works

Unlike traditional DV protocols such as RIP and IGRP, EIGRP does not rely on *periodic* updates: routing updates are sent only when there is a change. Remember that RIP and IGRP reset the invalid and flush timers upon receiving a route update. When a route is lost, the updates stop; the invalid and flush timers grow and grow (the timers are not reset), and, ultimately, the route is flushed from the routing table. This process of convergence assumes periodic updates. EIGRP's approach has the advantage that network resources are not consumed by periodic updates. However, if a router dies, taking away all its downstream routes, how would EIGRP detect the loss of these routes? EIGRP relies on small *hello packets* to establish neighbor relationships and to detect the loss of a neighbor. Neighbor relationships are discussed in detail in the next section.

RIP and IGRP suffer from a major flaw: *routing loops*. Routing loops happen when information about the loss of a route does not reach all routers in the network because an update packet gets dropped or corrupted. These routers (that have not received the information about the loss of the route) inject bad routing information back into the network by telling their neighbors about the route they know. EIGRP uses *reliable* transmission for all updates between neighbors. Neighbors acknowledge the receipt of updates, and if an acknowledgment is not received, EIGRP retransmits the update.

RIP and IGRP employ a battery of techniques to reduce the likelihood of routing loops: split horizon, hold-down timers, and poison reverse. These techniques do not guarantee that loops will not occur and, in any case, result in long convergence times. EIGRP uses the Diffusing Update Algorithm (DUAL) for all route computations. DUAL's convergence times are an order of magnitude lower than those of traditional DV algorithms. DUAL is able to achieve such low convergence times by maintaining a table of loop-free paths to every destination, in addition to the least-cost path. DUAL is described in more detail later in this chapter.

DUAL can support IP, IPX, and AppleTalk. A protocol-dependent module encapsulates DUAL messages and handles interactions with the routing table. In summary, DUAL requires:

1. A method for the discovery of new neighbors and their loss (see the next section, "Neighbor Relationship").
2. Reliable transmission of update packets between neighbors (see the later section "Reliable Transport Protocol").
3. Protocol-dependent modules that can encapsulate DUAL traffic in IP, IPX, or AppleTalk. This text will deal only with EIGRP in IP networks (see the later section "Protocol-Dependent Module").

I'll end this section with a discussion of EIGRP packet formats.

Neighbor Relationship

A router discovers a neighbor when it receives its first hello packet on a directly connected network. The router requests DUAL to send a full route update to the new neighbor. In response, the neighbor sends its full route update. Thus, a new neighbor relationship is established in the following steps:

1. When a router *A* receives a hello packet from a new neighbor *B*, *A* sends its topology table to router *B* in unicast updates with the *initialization bit* turned on.
2. When router *B* receives a packet with the initialization bit on, it sends its topology table to router *A*.

The interval between hello packets from any EIGRP-speaking router on a network is five seconds (by default) on most media types. Each hello packet advertises *hold-time*—the length of time the neighbor should consider the sender up. The default hold-time is 15 seconds. If no hellos are received for the duration of the hold-time, DUAL is informed that the neighbor is down. Thus, in addition to detecting a new neighbor, hello packets are also used to detect the loss of a neighbor.

The hello-interval can be changed with the following command in interface configuration mode:

```
ip hello-interval eigrp autonomous-system-number seconds
```

Lengthening the hello-interval will also lengthen the route convergence time. However, a longer hello-interval may be desirable on a congested network with many EIGRP routers.

If the hello-interval is changed, the hold-time should also be modified. A rule of thumb is to keep the hold-time at three times the hello-interval.

```
ip hold-time eigrp autonomous-system-number seconds
```

Note that the hello-interval and hold-time need *not* be the same for all routers on a network. Each router advertises its own hold-time, which is recorded in the neighbor's neighbor table.

The default hello-interval is 60 seconds (with a hold-time of 180 seconds) on multipoint interfaces (such as ATM, Frame Relay, and X.25) with link speeds of T-1 or less. Hello packets are multicast; no acknowledgments are expected.

The following output shows *NewYork*'s neighbors. The first column—labeled H—is the order in which the neighbors were learned. The hold-time for `172.16.251.2` (*Ames*) is 10 seconds, from which we can deduce that the last hello was received 5 seconds ago. The hold-time for `172.16.250.2` (*Chicago*) is 13 seconds, from which we can deduce that the last hello was received 2 seconds ago. The hold-time for a neighbor should not exceed 15 seconds or fall below 10 seconds (if the hold-time fell below 10 s, that would indicate the loss of one or more hello packets).

```
NewYork#sh ip eigrp neighbor
IP-EIGRP neighbors for process 10
H   Address                 Interface   Hold Uptime   SRTT   RTO  Q  Seq
                                        (sec)         (ms)       Cnt Num
1   172.16.251.2            Se0/1         10 00:17:08   28  2604  0  7
0   172.16.250.2            Se0/0         13 00:24:43   12  2604  0  14.
```

After a neighbor relationship has been established between A and B the only EIGRP overhead is the exchange of hello packets, unless there is a topological change in the network.

Reliable Transport Protocol

The EIGRP transport mechanism uses a mix of multicast and unicast packets, using reliable delivery when necessary. All transmissions use IP with the protocol type field set to 88. The IP multicast address used is `224.0.0.10`.

DUAL requires guaranteed and sequenced delivery for some transmissions. This is achieved using acknowledgments and sequence numbers. So, for example, *update packets* (containing routing table data) are delivered reliably (with sequence numbers) to all neighbors using multicast. *Acknowledgment packets*—with the correct sequence number—are expected from every neighbor. If the correct acknowledgment number is not received from a neighbor, the update is retransmitted as a unicast.

The sequence number (seq num) in the last packet from the neighbor is recorded to ensure that packets are received in sequence. The number of packets in the queue that might need retransmission is shown as a queue count (QCnt), and the smoothed round trip time (SRTT) is used to estimate how long to wait before retransmitting to the neighbor. The retransmission timeout (RTO) is the time the router will wait for an acknowledgment before retransmitting the packet in the queue.

Some transmissions do not require reliable delivery. For example, hello packets are multicast to all neighbors on an Ethernet segment, whereas acknowledgments are unicast. Neither hellos nor acknowledgments are sent reliably.

EIGRP also uses *queries* and *replies* as part of DUAL. Queries are multicast or unicast using reliable delivery, whereas replies are always reliably unicast. Query and reply packets are discussed in more detail in the next section.

Diffusing Update Algorithm (DUAL)

All route computations in EIGRP are handled by DUAL. One of DUAL's tasks is maintaining a table of loop-free paths to every destination. This table is referred to as the *topology table*. Unlike traditional DV protocols that save only the best (least-cost) path for every destination, DUAL saves all paths in the topology table. The least-cost path(s) is copied from the topology table to the routing table. In the event of a failure, the topology table allows for very quick convergence if another loop-free path is available. If a loop-free path is not found in the topology table, a route recomputation must occur, during which DUAL queries its neighbors, who, in turn, may query their neighbors, and so on... hence the name "Diffusing" Update Algorithm.

These processes are described in detail in the following sections.

Reported distance

Just like RIP and IGRP, EIGRP calculates the lowest cost to reach a destination based on updates[*] from neighbors. An update from a router *R* contains the cost to reach the destination network *N* from *R*. This cost is referred to as the *reported distance* (RD). *NewYork* receives an update from *Ames* with a cost of 281,600, which is *Ames*'s cost to reach `172.16.100.0`. In other words, the RD for *Ames* to reach `172.160.100.0` as reported to *NewYork* is 281,600. Just like *Ames*, *Chicago* will report its cost to reach `172.16.100.0`. *Chicago*'s RD is 2,195,456 (see Figure 4-2).

[*] Unlike RIP and IGRP, EIGRP updates are *not* periodic. EIGRP updates are sent only when there is a topological change in the network.

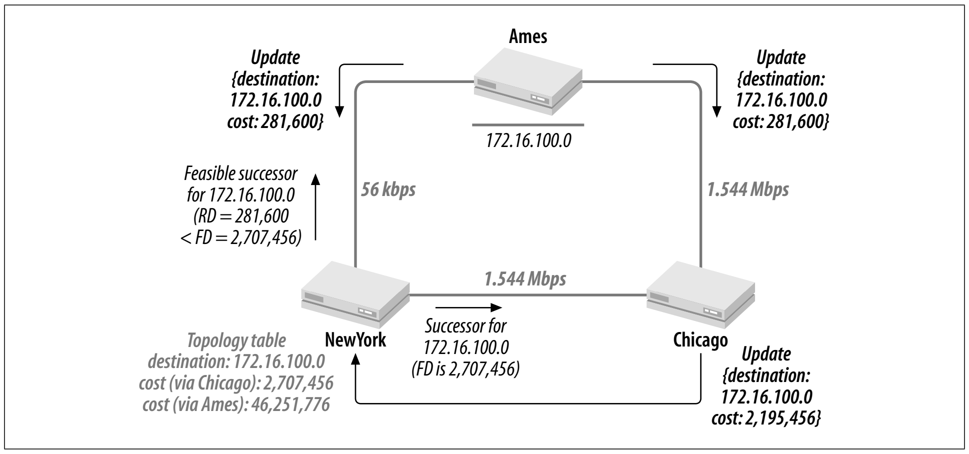
Ames
Update {destination: 172.16.100.0 cost: 281,600}
Update {destination: 172.16.100.0 cost: 281,600}
172.16.100.0
Feasible successor for 172.16.100.0 (RD = 281,600 < FD = 2,707,456)
56 kbps
1.544 Mbps
1.544 Mbps
Topology table destination: 172.16.100.0 cost (via Chicago): 2,707,456 cost (via Ames): 46,251,776
NewYork
Successor for 172.16.100.0 (FD is 2,707,456)
Chicago
Update {destination: 172.16.100.0 cost: 2,195,456}

Figure 4-2. Ames is a feasible successor for 172.16.100.0

Feasible distance and successor

NewYork will compute its cost to reach `172.16.100.0` via *Ames* and *Chicago*. *NewYork* will then compare the metrics for the two paths. *NewYork*'s cost via *Ames* is 46,251,776. *NewYork*'s cost via *Chicago* is 2,707,456. The lowest cost to reach a destination is referred to as the *feasible distance* (FD) for that destination. *NewYork*'s FD to `172.16.100.0` is 2,707,456 (*BandW* = 1,544 and *Delay* = 4,100). The next-hop router in the lowest-cost path to the destination is referred to as the *successor*. *NewYork*'s successor for `172.16.100.0` is `172.16.50.1` (*Chicago*).

Feasibility condition and feasible successor

If a reported distance for a destination is less than the feasible distance for the same destination, the router that advertised the RD is said to satisfy the *feasibility condition* (FC) and is referred to as a *feasible successor* (FS). *NewYork* sees an RD of 281,600 via *Ames*, which is lower than *NewYork*'s FD of 2,707,456. *Ames* satisfies the FC. *Ames* is an FS for *NewYork* to reach `172.16.100.0`.

Loop freedom

The feasibility condition is a test for *loop freedom*: if the FC is met, the router advertising the RD must have a path to the destination not through the router checking the FC—if it did, the RD would have been higher than the FD.

Let's illustrate this concept with another example. Consider the network in Figure 4-3. The metric values used in this example have been simplified to small numbers to make it easier to follow the concept.

Router *A*'s best route to network *N* is via router *B*, and the cost of this path is 100 (*A*'s FD to *N* is 100). Router *X* also knows how to get to network *N*; *X* advertises *N*

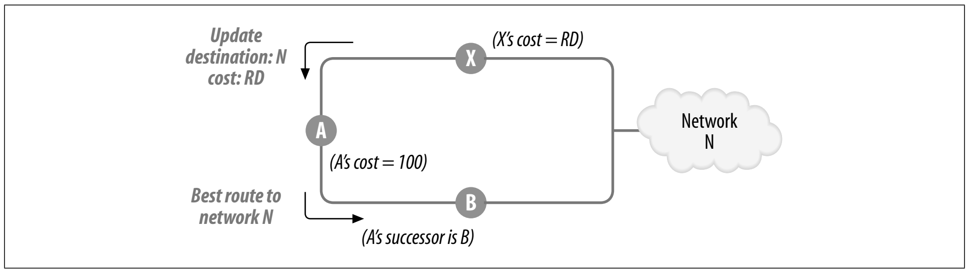

Figure 4-3. Loop freedom

to *A* in an update packet (*A* copies this information into its topology table). In the event that *A*'s link to *B* fails, *A* can use the route to *N* via *X* if *X* does not use *A* to get to *N* (in other words, if the path is loop-free). Thus, the key question for *A* to answer is whether or not the path that *X* advertises is loop-free.

Here is how *A* answers this question. Let's say that *X* advertises *N* with a metric of 90 (*X*'s RD for *N*). *A* compares 90 (RD) with 100 (FD). Is RD < FD? This comparison is the FC check. Since *A*'s FD is 100, *X*'s path to *N* must not be via *A* (and is loop-free). If *X* advertises *N* with a metric of 110, *X*'s path to *N* could be via *A* (the RD is not less than the FD, so the FC check fails)—110 could be *A*'s cost added to the metric of the link between *A* and *X* (and, hence, is not guaranteed to be free of a loop).

Topology table

All destinations advertised by neighbors are copied into the topology table. Each destination is listed along with the neighbors that advertised the destination, the RD, and the metric to reach the destination via that neighbor. Let's look at *NewYork*'s topology table and zoom in on destination `172.16.100.0`. There are two neighbors that sent updates with this destination: *Chicago* (`172.16.250.2`) and *Ames* (`172.16.251.2`), as shown on lines 9 and 10, respectively:

```
    NewYork#sh ip eigrp topology
    IP-EIGRP Topology Table for process 10

    Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
           r - Reply status

    ...
    P 172.16.100.0/24, 1 successors, FD is 2,707,456
 9           via 172.16.250.2 (2,707,456/2,195,456), Serial0
10           via 172.16.251.2 (46,251,776/281,600), Serial1
```

Chicago sent an update with an RD of 2,195,456, and *Ames* sent an update with an RD *of* 281,600. *NewYork* computes its own metric to `172.16.100.0`: 2,707,456 and 46,251,776 via *Chicago* and *Ames*, respectively. *NewYork* uses the lower-cost path via *Chicago*. *NewYork*'s FD to `172.16.100.0` is thus 2,707,456, and *Chicago* is the successor. Next *NewYork* checks to see if *Ames* qualifies as a feasible successor. *Ames*'s RD is 281,600. This is checked against the FD. Since the RD < FD (281,600 < 2,707,456), *Ames* is a feasible successor (see Figure 4-2).

Note that not all loop-free paths satisfy the FC. Thus, *NewYork*'s topology table does not contain the alternate path to `172.16.50.0` (via *Ames*). The FC guarantees that the paths that satisfy the condition are loop-free; however, not all loop-free paths satisfy the FC.

Let's take a closer look at `172.16.50.0` (*Chicago*) in *NewYork*'s topology table:

```
NewYork#sh ip eigrp topology
IP-EIGRP Topology Table for process 10

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status

...
P 172.16.50.0/24, 1 successors, FD is 2195456
         via 172.16.250.2 (2195456/281600), Serial0
```

Notice that *Ames* (`172.16.251.2`) did not become a feasible successor, even though *Ames* offers a valid loop-free path. The condition that *Ames* would have to satisfy to become a feasible successor is for its RD to be less than *NewYork*'s FD to `172.16.50.0`. *Ames*'s RD can be seen from *Ames's* routing table:

```
   Ames#sh ip route
   ...
        172.16.0.0/24 is subnetted, 6 subnets
   C       172.16.252.0 is directly connected, Serial0
   D       172.16.250.0 [90/2681856] via 172.16.252.1, 00:21:10, Serial0
   C       172.16.251.0 is directly connected, Serial1
11 D       172.16.50.0 [90/2195456] via 172.16.252.1, 00:21:10, Serial0
   D       172.16.1.0 [90/2707456] via 172.16.252.1, 00:15:36, Serial0
   C       172.16.100.0 is directly connected, Ethernet0
```

Ames's metric to `172.16.50.0` is 2,195,456 (line 11). This will be the metric that *Ames* reports to *NewYork*. The RD is thus 2,195,456. *NewYork*'s FD to `172.16.50.0` is 2,195,456. The RD and the FD are equal, which is not surprising given the topology: both *NewYork* and *Ames* have identical paths to `172.16.50.0`—a T-1 link, a router, and the destination Ethernet segment. Since the condition for feasible successor is that RD < FD, *Ames* is not an FS for `172.16.50.0` (see Figure 4-4).

The output of *show ip eigrp topology* shows only feasible successors. The output of *show ip eigrp topology all-links* shows all neighbors, whether feasible successors or not.

Note the "P" for "passive state" in the left margin of each route entry in *NewYork*'s topology table. *Passive state* indicates that the route is in quiescent mode, implying that the route is known to be good and that no activities are taking place with respect to the route.

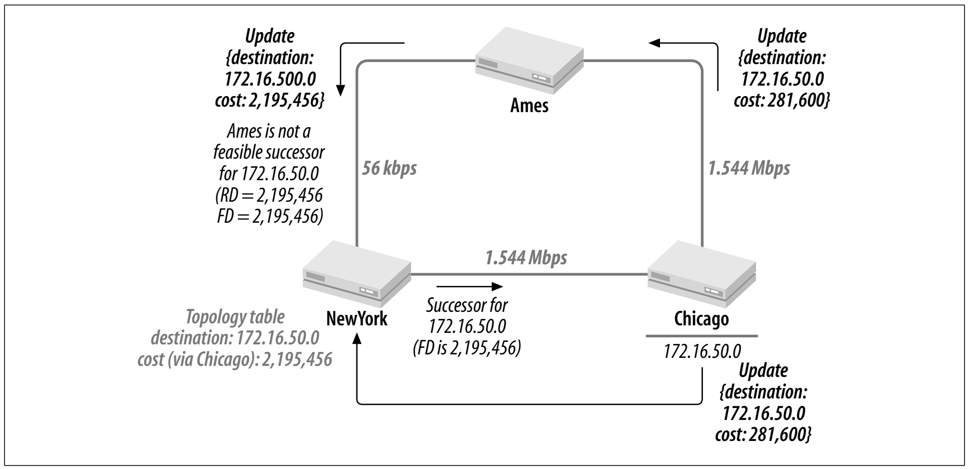

Figure 4-4. Ames is not a feasible successor for 172.16.50.0

Any of the following events can cause DUAL to reevaluate its feasible successors:

- The transition in the state of a directly connected link
- A change in the metric of a directly connected link
- An update from a neighbor

If DUAL finds a feasible successor in its own topology table after one of these events, the route remains in passive state. If DUAL cannot find a feasible successor in its topology table, it will send a query to all its neighbors and the route will transition to *active state*.

The next section contains two examples of DUAL reevaluating its topology table. In the first example, the route remains passive; in the second example, the route becomes active before returning to the passive state.

Convergence in DUAL— local computation

Let's say that the *NewYork* → *Chicago* link fails (Figure 4-5).

NewYork's routing table shows that `172.16.100.0` and `172.16.50.0` are learned via this link (*Serial0*):

```
NewYork#sh ip route
...
     172.16.0.0/24 is subnetted, 6 subnets
...
D       172.16.50.0 [90/2195456] via 172.16.250.2, 00:18:54, Serial0
D       172.16.100.0 [90/2707456] via 172.16.250.2, 00:18:54, Serial0
...
```

These routes become invalid. DUAL attempts to find new successors for both destinations—`172.16.50.0` and `172.16.100.0`.

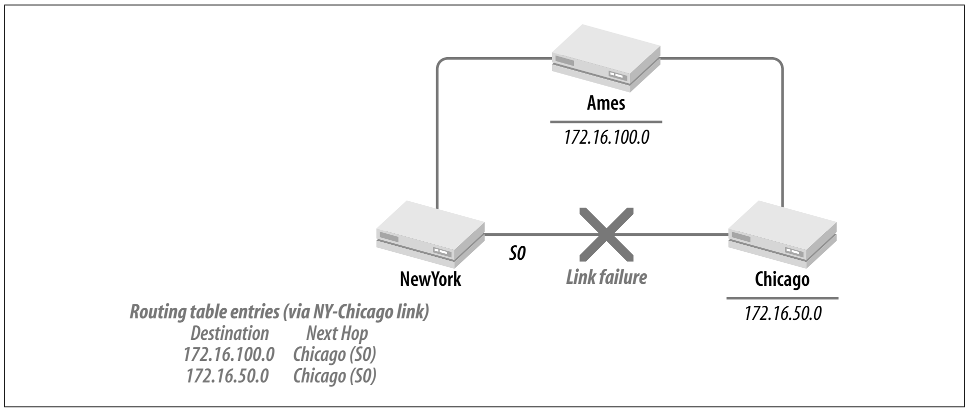

Figure 4-5. Link failure

Let's start with `172.16.100.0`. DUAL checks the topology table for `172.16.100.0`:

```
    NewYork#sh ip eigrp topology
    ...
    P 172.16.100.0/24, 1 successors, FD is 2707456
12           via 172.16.250.2 (2707456/2195456), Serial0
13           via 172.16.251.2 (46251776/281600), Serial1
```

Since *Serial0* is down, the only feasible successor is `172.16.251.2` (*Ames*). Let's review how *Ames* qualifies as an FS. The FS check is:

- RD < FD.
- RD=281,600 (line 13).
- FD=2,707,456 (line 12).
- Since 281,600 < 2,707,456, *Ames* qualifies as an FS.

In plain words, this implies that the path available to *NewYork* via *Ames* (the FS) is independent of the primary path that just failed. DUAL installs *Ames* as the new successor for `172.16.100.0`.

In our case study, only one FS was available. In general, multiple FSs may be available, all of which satisfy the condition that their RD < FD, where FD is the cost of the route to the destination via the successor that was just lost.

DUAL will compute its metric to reach the destination via each FS. Since DUAL is searching for the successor(s) for this destination, it will choose the minimum from this set of metrics via each FS. Let the lowest metric be *Dmin*. If only one FS yields this metric of *Dmin*, that FS becomes the new successor. If multiple FSs yield metrics equal to *Dmin*, they all become successors (subject to the limitation in the maximum number of parallel paths allowed—four or six, depending on the IOS version number). Since the new successor(s) is found locally (without querying any other router), the route stays in passive state. After DUAL has installed the new successor, it sends an update to all its neighbors regarding this change.

How long does this computation take? We simulated the failure of the *NewYork* → *Chicago* link in our laboratory. To measure how long EIGRP would take to converge after the failure of the link, we started a long ping test just before failing the *NewYork* → *Chicago* link:

```
NewYork#ping
Protocol [ip]:
Target IP address: 172.16.100.1
Repeat count [5]: 1000
...
Sending 1000, 100-byte ICMP Echos to 172.16.100.1, timeout is 2 seconds:

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Success rate is 99 percent (999/1000), round-trip min/avg/max = 1/3/92 ms
```

Note that only one ping packet was lost during this computation, implying that the convergence time (including the time to detect the failure of the link) was in the range of two to four seconds.

Convergence in DUAL—diffusing computation

Let's next follow the steps that DUAL would take for `172.16.50.0`. Notice that this is a different case in that when *Serial0* is down, *NewYork* has no feasible successors in its topology table (see line 14).

```
    NewYork#sh ip eigrp topology
    ...
    P 172.16.50.0/24, 1 successors, FD is 2195456
14          via 172.16.250.2 (2195456/281600), Serial0
    ...
```

DUAL knows of no feasible successors, but *NewYork* has a neighbor that may know of a feasible successor. DUAL places the route in active state (see line 15) and sends a query to all its neighbors:

```
    NewYork#sh ip eigrp topology
    IP-EIGRP Topology Table for process 10

    Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
           r - Reply status

    ...
15  A 172.16.50.0/24, 0 successors, FD is 2195456, Q
        1 replies, active 00:00:06, query-origin: Local origin
        Remaining replies:
16          via 172.16.251.2, r, Serial1
```

which in this case is only `172.16.251.2` (*Ames*, as in line 16). *NewYork* sets the reply flag on (line 16), which indicates that *NewYork* expects a reply to the query. *Ames* receives the query and marks its topology table entry for `172.16.50.0` via *NewYork* as down. Next, *Ames* checks its topology table for a feasible successor:

```
    Ames#sh ip eigrp topology
    IP-EIGRP Topology Table for process 10

    Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
           r - Reply status

    ...
17  P 172.16.50.0/24, 1 successors, FD is 2195456
             via 172.16.252.1 (2195456/281600), Serial0
    ...
```

and finds that it has a successor (`172.16.252.1`). *Ames* sends a reply packet to *NewYork* with an RD of 2,195,456 (line 17). *NewYork* marks the route as passive and installs a route for `172.16.50.0` via `172.16.251.2` (*Ames*).

In general, if DUAL does not find a feasible successor, it forwards the query to its neighbors. The query thus propagates ("diffuses") until a reply is received. Routers that did not find a feasible successor would return an unreachable message. So, if *Ames* did not have a feasible successor in its topology table, it would mark the route as active and propagate the query to its neighbor, if it had another neighbor. If *Ames* had no other neighbor (and no feasible successor) it would return an unreachable message to *NewYork* and mark the route as unreachable in its own table.

When DUAL marks a route as active and sets the *r* flag on, it sets a timer for how long it will wait for a reply. The default value of the timer is three minutes. DUAL waits for a reply from all the neighbors it queries. If a neighbor does not respond to a query, the route is marked as *stuck-in-active* and DUAL deletes all routes in its topology table that point to the unresponsive neighbor as a feasible successor.

Protocol-Dependent Module

The successors in the DUAL topology table are eligible for installation in the routing table. Successors represent the best path to the destination known to DUAL. However, whether the successor is copied into the routing table is another matter. The router may be aware of a route to the same destination from another source (such as another routing protocol or via a static route) with a lower *distance*. The IP protocol-dependent module (PDM) handles this task. The PDM may also carry information in the reverse direction—from the routing table to the topology table. This will occur if routes are being redistributed into EIGRP from another protocol.

The PDM is also responsible for encapsulating EIGRP messages in IP packets.

EIGRP Packet Format

EIGRP packets are encapsulated directly in IP with the protocol field set to 88. The destination IP address in EIGRP depends on the packet type—some packets are sent as multicast (with an address of `224.0.0.10`) and others are sent as unicast (see the earlier section "Reliable Transport Protocol" for more details). The source IP address is the IP address of the interface from which the packet is issued.

Following the IP header is an EIGRP header. Key fields in the EIGRP header are as follows, and are also shown in Figure 4-6:

- The *opcode* field specifies the EIGRP packet type (update, query, reply, hello).
- The *checksum* applies to the entire EIGRP packet, excluding the IP header.
- The rightmost bit in the *flags* field is the initialization bit and is used in establishing a new neighbor relationship (see "Neighbor Relationship" earlier in this chapter).
- The *sequence* and *ack* fields are used to send messages reliably (see "Reliable Transport Protocol" earlier in this chapter).
- The *AS number* identifies the EIGRP process issuing the packet. The EIGRP process receiving the packet will process the packet only if the receiving EIGRP process has the same AS number; otherwise, the packet will be discarded.

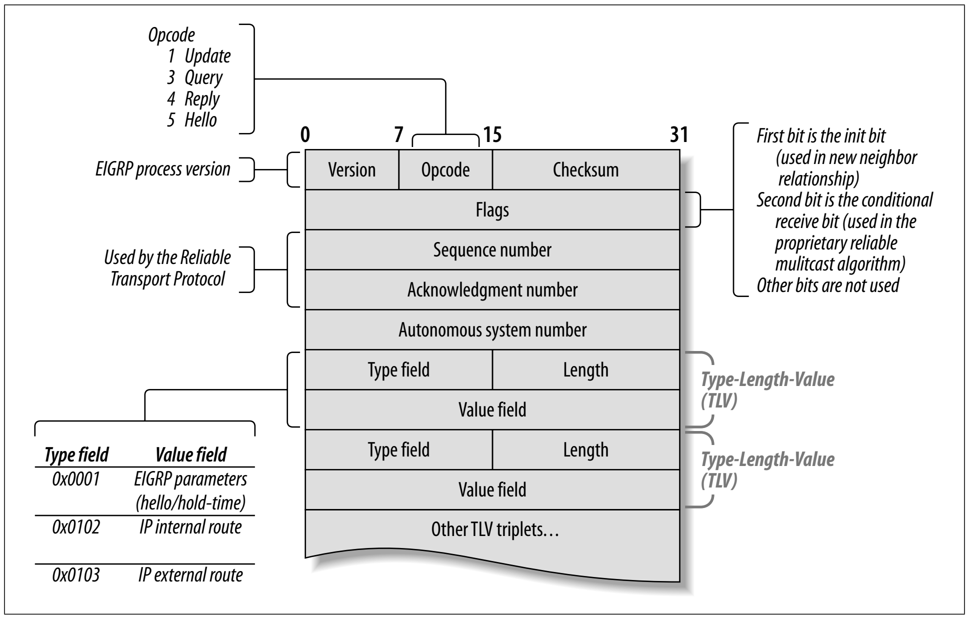

Figure 4-6. Format of EIGRP packets

The fields following the EIGRP header depend on the opcode field. Of particular interest to routing engineers is the information in updates. We will ignore the other types of EIGRP messages and focus on IP internal route updates and IP external route updates.

Internal routes contain destination network numbers learned within this EIGRP AS. For example, *NewYork* learns `172.16.50.0` from EIGRP 10 on *Chicago* as an internal route.

External routes contain destination network numbers that were not learned within this EIGRP AS but rather derived from another routing process and redistributed into this EIGRP AS.

Internal and external routes are represented differently in the EIGRP update.

Internal routes

Internal routes have a *type* field of 0x0102. The metric information contained with the route is much like IGRP's (see Chapter 3). However, there are two new fields: *next hop* and *prefix length*. Figure 4-7 shows the value field for the IP internal route.

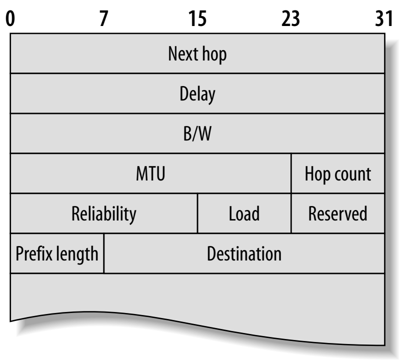

Figure 4-7. EIGRP internal route

The next hop identifies the router to send packets destined for *destination*, the network number of the destination. In general, the next hop field for internal routes will be the IP address of the router on the interface on which it is issuing the update.

The prefix length field signifies the subnet mask to be associated with the network number specified in the destination field. Thus, if an EIGRP router is configured as follows:

```
    ip address 172.16.1.1 255.255.255.0
```

it will advertise `172.16.1.0` with a prefix length of 24.

Likewise, if the router is configured as follows:

```
ip address 172.16.250.1 255.255.255.252
```

it will advertise `172.16.250.0` with a prefix length of 30.

External routes

Additional fields are required to represent the source from which external routes are derived, as shown in Figure 4-8.

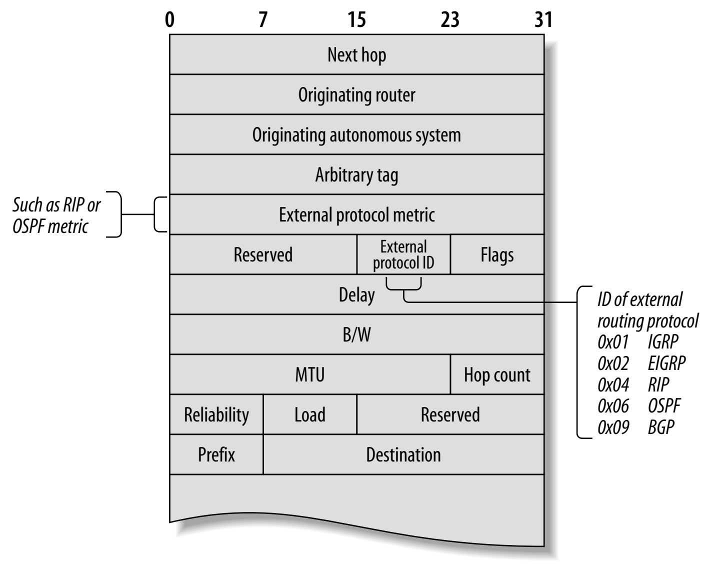

Figure 4-8. EIGRP external route

The next hop field identifies the router to send packets destined for *destination*, the network number of the destination. This field was absent in the IGRP update. Let's look at what this field signifies.

In IGRP, if router *X* sends an update to router *A* with a destination network number of *N*, router *A*'s next hop for packets to *N* will be *X*. In EIGRP, router *X* can send an update to router *A* with a destination network number of *N* and a next hop field of *Y*. This is useful, say, in a scenario where *X* and *Y* are running RIP and *X* is redistributing routes from RIP to IGRP. When *X* sends an update to its neighbors on a shared network, *X* can tell them to send traffic for network *N* directly to *Y* and not to *X*. This saves *X* from having to accept traffic on a shared network and then reroute it to *Y*.*

* You may ask why this cannot be handled by ICMP redirects. Cisco does not support redirects between routers.

The *originating router, originating AS, external protocol metric*, and *external protocol ID* fields specify information about the router and the routing process from which this route was derived. The external protocol ID specifies the routing protocol from which this route was derived. Here is a partial list of external protocol IDs: IGRP—0x01; EIGRP—0x02; RIP—0x04; OSPF—0x06; BGP—0x09. Thus, if a route was learned from RIP with a hop count of 3 and redistributed into EIGRP, the originating router field would contain the address of the RIP router, the originating AS field would be empty, the external protocol metric would be 3, and the external protocol ID would be 0x04.

The *arbitrary tag* field is used to carry route maps.

Candidate default routes are marked by setting the flags field to 0x02. A flags field of 0x01 indicates an external route (but not a candidate default route).

The other parameters in the external route packet are similar to those in IGRP.

Variable Length Subnet Masks

Unlike RIP and IGRP, EIGRP updates carry subnet mask information. The network architect now has the responsibility of using addresses wisely. Reviewing TraderMary's configuration, a mask of `255.255.255.0` on the serial links is wasteful: there are only two devices on the link, so a 24-bit mask will waste 252 addresses. A 30-bit mask (`255.255.255.252`) allows two usable IP addresses in each subnet, which fits a serial line exactly.

Let's say that the network architect decided to subdivide `172.16.250.0` using a 30-bit mask for use on up to 64 possible subnets. The subnets that thus become available are:

1. `172.16.250.0`
2. `172.16.250.4`
3. `172.16.250.8`
4. ...

64. `172.16.250.252`

The serial links in TraderMary's network can be readdressed using these subnets:

```
hostname NewYork
...
interface Ethernet0
ip address 172.16.1.1 255.255.255.0
!
interface Ethernet1
ip address 192.168.1.1 255.255.255.0
!
interface Serial0
description New York to Chicago link
```

```
ip address 172.16.250.1 255.255.255.252
!
interface Serial1
description New York to Ames link
bandwidth 56
ip address 172.16.250.5 255.255.255.252
...
router eigrp 10
network 172.16.0.0


hostname Chicago
...
interface Ethernet0
ip address 172.16.50.1 255.255.255.0
!
interface Serial0
description Chicago to New York link
ip address 172.16.250.2 255.255.255.252
!
interface Serial1
description Chicago to Ames link
ip address 172.16.250.9 255.255.255.0
...

router eigrp 10
network 172.16.0.0


hostname Ames
...
interface Ethernet0
ip address 172.16.100.1 255.255.255.0
!
interface Serial0
description Ames to Chicago link
ip address 172.16.250.10 255.255.255.0
!
interface Serial1
description Ames to New York link
bandwidth 56
ip address 172.16.250.6 255.255.255.0
...

router eigrp 10
network 172.16.0.0
```

NewYork's routing table now looks like this:

```
NewYork#sh ip route
...
     172.16.0.0/16 is variably subnetted, 6 subnets, 2 masks
D       172.16.250.8/30 [90/2681856] via 172.16.250.2, 00:18:54, Serial0
C       172.16.250.0/30 is directly connected, Serial0
```

```
C       172.16.250.4/30 is directly connected, Serial1
D       172.16.50.0/24 [90/2195456] via 172.16.250.2, 00:18:54, Serial00
C       172.16.1.0/24 is directly connected, Ethernet0
D       172.16.100.0/24 [90/2707456] via 172.16.250.2, 00:18:54, Serial0
C    192.168.1.0/24 is directly connected, Ethernet1
```

Note that each route is now accompanied by its mask. When `172.16.0.0` had uniform masking, the routing table did not show the mask.

Further, let's say that Casablanca is a small office with only a dozen people on the staff. We may safely assign Casablanca a mask of `255.255.255.192` (a limit of 62 usable addresses). Forward-thinking is important when assigning addresses. When running IGRP, the network architect may have had the foresight to assign addresses from the beginning of the range. Excess addresses should not be squandered, such as by randomly choosing addresses for hosts. A general rule is to start assigning addresses from the beginning or the bottom of an address range. When a site is shrinking, again keep all addresses at one end.

Using subnet masks that reflect the size of the host population conserves addresses. Put on your plate only as much as you will eat.

Route Summarization

The default behavior of EIGRP is to summarize on network-number boundaries. This is similar to RIP and IGRP and is a prudent way for a routing protocol to reduce the number of routes that are propagated between routers. However, there are some enhancements in the way EIGRP summarizes routes that merit a closer look.

Automatic Summarization

Say TraderMary's network expands again, this time with a node in Shannon. Shannon gets connected to the London office via a 56-kbps link, as shown in Figure 4-9.

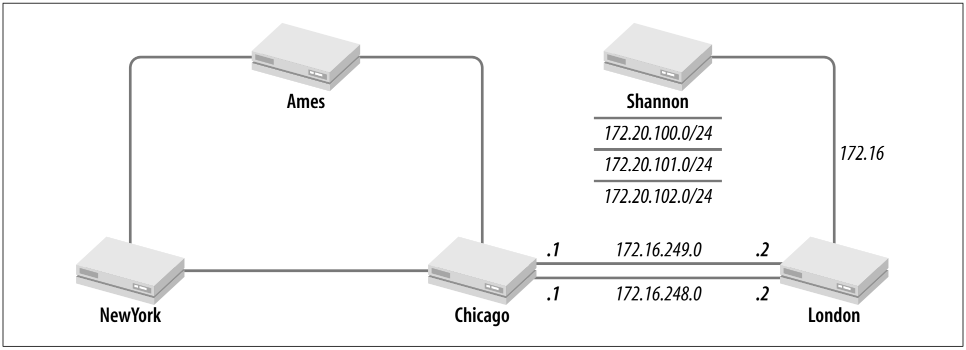

Figure 4-9. Route summarization

Shannon has three Ethernet segments with an IP subnet on each: 172.20.100.0/24, 172.20.101.0/24, and 172.20.102.0/24. The routers in London and Shannon are configured to run EIGRP 10 in keeping with the routing protocol in use in the U.S. *Shannon* will advertise 172.20.0.0/16 to *London* because the serial link from *London* to *Shannon* represents a network-number boundary (172.20.0.0/172.16.0.0). *Shannon* itself will see all 172.16.0.0 subnets (without summarization) because it has a directly connected 172.16.0.0 network.

In EIGRP, the router doing the summarization will build a route to *null0* (line 18) for the summarized address. Let's check *Shannon*'s routing table:

```
   Shannon#sh ip route 172.20.0.0
   ...
        172.20.0.0/16 is subnetted, 6 subnets
   C       172.20.100.0/24 is directly connected, Ethernet0
   C       172.20.101.0/24 is directly connected, Ethernet1
18 D       172.20.0.0/16 is a summary, 00:12:11, Null0
   C       172.20.102.0/24 is directly connected, Ethernet2
```

The route to *null0* ensures that if *Shannon* receives a packet for which it has no route (e.g., 172.20.1.1), it will route the packet using the null interface, thereby dropping the packet, rather than using some other route for the packet (such as a default route).

Now, let's muddy the picture up a bit. TraderMary acquires a small company in Ottawa which also happens to use a 172.20.0.0 subnet—172.20.1.0! The new picture looks something like Figure 4-10.

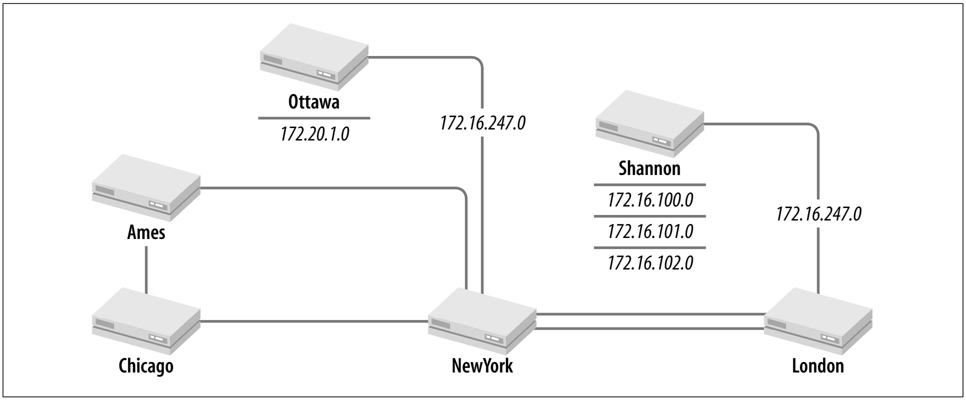

Figure 4-10. TraderMary's networks in Shannon and Ottawa

Ottawa is also configured to run EIGRP 10 with a link from *NewYork*. Since the IP address on the link is 172.16.0.0, *Ottawa* will send a summary update of 172.20.0.0 to *NewYork*.

We have a problem now. There are two sources advertising 172.20.0.0, and depending on where we are in the network, we will be able to route only to *Ottawa* or

Shannon. Thus, *NewYork* will install `172.20.0.0` only via *Ottawa*, and *London* will install `172.20.0.0` only via *Shannon*.

Unlike RIP and IGRP, EIGRP provides the option of disabling route summarization. Thus, *Shannon* and *Ottawa* can be configured as follows:

```
hostname Shannon
...
router eigrp 10
network 172.16.0.0
network 172.20.0.0
no auto-summary


hostname Ottawa
...
router eigrp 10
network 172.16.0.0
network 172.20.0.0
no auto-summary
```

When *no auto-summary* is turned on, *Shannon* and *Ottawa* will advertise their subnets to the rest of the network. The subnets happen to be unique, so any router will be able to route to any destination in the network.[*]

Note that *no auto-summary* was required only on the *Shannon* and *Ottawa* routers. *NewYork* and *London* and other routers will pass these subnets through (without summarizing them). Summarization happens only at a border between major network numbers, not at other routers.

The moral of this story is that EIGRP networks do not have to be contiguous with respect to major network numbers. However, I do not recommend deliberately building discontiguous networks. Summarizing on network-number boundaries is an easy way to reduce the size of routing tables and the complexity of the network. Disabling route summarization should be undertaken only when necessary.

Manual Summarization

EIGRP allows for the summarization of (external or internal) routes on any bit boundary. Manual summarization can be used to reduce the size of routing tables.

In our example, the network architect may decide to allocate blocks of addresses to *NewYork*, *Ames*, *Chicago*, etc. *NewYork* is allocated the block `172.16.1.0` through `172.16.15.0`. This may also be represented as `172.16.0.0/20`, signifying that the first four bits of the third octet in this range are all zeros, as is true for `172.16.1.0` through `172.16.15.0`.

[*] If the subnets overlapped, disabling route summarization would not do us any good. There are other methods to tackle duplicate address problems, such as Network Address Translation (NAT).

```
    hostname NewYork
    ...
19  interface Ethernet0
    ip address 172.16.1.1 255.255.255.0
    !
    interface Ethernet1
    ip address 192.168.1.1 255.255.255.0
    !
20  interface Ethernet2
    ip address 172.16.2.1 255.255.255.0
    !
    interface Serial0
    description New York to Chicago link
    ip address 172.16.250.1 255.255.255.0
    ip summary-address eigrp 10 172.16.0.0 255.255.240.0
    !
    interface Serial1
    description New York to Ames link
    bandwidth 56
    ip address 172.16.251.1 255.255.255.0
21  ip summary-address eigrp 10 172.16.0.0 255.255.240.0
    ...
    router eigrp 10
    network 172.16.0.0
```

NewYork now has two Ethernet segments (lines 19 and 20) from this block and has also been configured to send a summary route for this block (line 21) to its neighbors. The configuration of these routers is as shown in Figure 4-1. Here's *NewYork*'s routing table:

```
    NewYork#sh ip route
    ...
         172.16.0.0/16 is variably subnetted, 8 subnets, 2 masks
    D       172.16.252.0/24 [90/2681856] via 172.16.250.2, 00:01:44, Serial0
    C       172.16.250.0/24 is directly connected, Serial0
    C       172.16.251.0/24 is directly connected, Serial1
22  D       172.16.0.0/20 is a summary, 00:03:22, Null0
    C       172.16.1.0/24 is directly connected, Ethernet0
    C       172.16.2.0/24 is directly connected, Ethernet2
    D       172.16.50.0/20 [90/2195456] via 172.16.250.2, 00:01:45, Serial0
    D       172.16.100.0/20 [90/2707456] via 172.16.250.2, 00:01:45, Serial0
    C    192.168.1.0/24 is directly connected, Ethernet1
```

Note that *NewYork* installs a route to the null interface for the summarized address (`172.16.0.0/20`, as in line 22). Further, routers *Ames* and *Chicago* install this aggregated route (line 23) and not the individual `172.16.1.0/24` and `172.16.2.0/24` routes:

```
    Chicago#sh ip route
    ...
         172.16.0.0/16 is variably subnetted, 8 subnets, 2 masks
    C       172.16.252.0/24 is directly connected, Serial1
    C       172.16.250.0/24 is directly connected, Serial0
    D       172.16.251.0/24 [90/2681856] via 172.16.250.1, 00:02:30, Serial0
                            [90/2681856] via 172.16.252.2, 00:02:30, Serial1
    C       172.16.50.0/24 is directly connected, Ethernet0
23  D       172.16.0.0/20 [90/2195456] via 172.16.250.1, 00:02:12, Serial0
    D       172.16.100.0/20 [90/2195456] via 172.16.252.2, 00:02:10, Serial1
```

The address aggregation commands on *NewYork* reduce the routing-table size in the rest of the network. Note that address aggregation plans need to be laid out ahead of time so that network numbers can be allocated accordingly. Thus, in the previous example, *NewYork* was allocated a block of 16 subnets:

`172.16.96.0` through `172.16.15.0`

Continuing this scheme, *Ames* may be allocated a block of 16 addresses that envelop the network number it is currently using (`172.16.100.0`):

`172.16.96.0` through `172.16.111.0`

and *Chicago* may be allocated a block of 16 addresses that envelop the network number it is currently using (`172.16.50.0`):

`172.16.48.0` through `172.16.63.0`

Ames could now be configured to summarize its block using the statement on its serial interfaces:

```
ip summary-address eigrp 10 172.16.0.0 255.255.240.0
```

and *Chicago* could be configured to summarize its block using the statement on its serial interfaces:

```
ip summary-address eigrp 10 172.16.0.0 255.255.240.0
```

Default Routes

EIGRP tracks default routes in the external section of its routing updates. Candidate default routes are marked by setting the flags field to 0x02.

Default routes are most often used to support branch offices that have only one or two connections to the core network (see Figure 4-11).

The core router is configured as follows:

```
    hostname core1
    !
    interface Ethernet0
     ip address 192.168.1.1 255.255.255.0
    ...
    interface Serial0
    ip address 172.16.245.1 255.255.255.0
    ...
    router eigrp 10
24   redistribute static metric 56 100 255 1 255
     network 172.16.0.0
    !
    ip classless
25  ip route 0.0.0.0 0.0.0.0 Null0
```

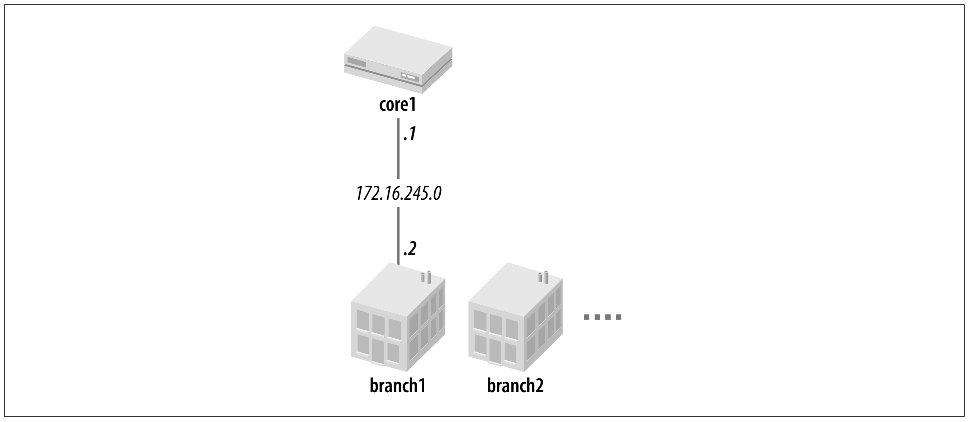

Figure 4-11. Branch offices only need a default route

The branch router is configured as follows:

```
    hostname branch1
    ...
    interface Serial0
    ip address 172.16.245.2 255.255.255.0
    ...
26  router eigrp 10
    network 172.16.0.0
```

An examination of *branch1*'s routing table would show:

```
    branch1#sh ip route
    ...
    Gateway of last resort is 172.16.251.1 to network 0.0.0.0

         172.16.0.0/24 is subnetted, 6 subnets
    C       172.16.245.0 is directly connected, Serial0
    ...
27  D*EX 0.0.0.0/0 [170/46251776] via 172.16.245.1, 00:01:47, Serial0
```

Since the default route is an external route, it is tagged with a distance of 170 (line 27).

The following steps were followed in the creation of this default route:

1. Network `0.0.0.0` was defined as a static route on *core1* (see line 25).
2. Network `0.0.0.0` was redistributed into EIGRP 10 (see line 24).
3. A default metric was attached to the redistribution (line 24).
4. EIGRP 10 was turned on in *branch1* (line 26).

To increase the reliability of the connection to branches, each branch may be connected to two core routers. *branch1* will now receive two default routes. One router (say, *core1*) may be set up as the primary, and the second router (*core2*) as backup. To do this, set up the default from *core2* with a *worse* metric, as we did for IGRP in Chapter 3.

Troubleshooting EIGRP

EIGRP can be difficult to troubleshoot because of its complexity. As a reminder, the best preparation for troubleshooting a network is to be familiar with the network and its state during normal (trouble-free) conditions. Become familiar with the routing tables, their sizes, the summarization points, routing timers, etc. Also, plan ahead with "what-if" scenarios. What if router *X* failed or link *Y* dropped? How would connectivity recover? Will all the routes still be in every router's table? Will the routes still be summarized?

Perhaps the second-best preparation for troubleshooting a network is the ability to track network implementations and changes. If network implementations/changes are made in a haphazard way with no central control, an implementation team may walk away from a change (unaware that their change caused an outage) and it may take the troubleshooting team hours, or even days, to unravel the events that led to the outage. Besides making the network more vulnerable, such loose methods of network operation create bad relationships between teams.

The following sections are a partial list of network states/conditions to check when looking for clues to routing problems in EIGRP.

Verifying Neighbor Relationships

If a router is unable to establish a stable relationship with its neighbors, it cannot exchange routes with those neighbors. The neighbor table can help check the integrity of neighbor relationships. Here is a sample of *NewYork*'s neighbor table:

```
NewYork#sh ip eigrp neighbor
IP-EIGRP neighbors for process 10
H   Address                 Interface   Hold Uptime   SRTT   RTO  Q  Seq
                                        (sec)         (ms)       Cnt Num
1   172.16.251.2            Se0/1         10 00:17:08   28  2604  0  7
0   172.16.250.2            Se0/0         13 00:24:43   12  2604  0  14
```

First, check that the neighbor count matches the number of EIGRP speakers. If routers *A*, *B*, and *C* share an Ethernet segment and run EIGRP 10, all four routers should see each other in their neighbor tables. If router *C* is consistently missing from *A* and *B*'s tables, there may be a physical problem with *C* or *C* may be misconfigured (check *C*'s IP address and EIGRP configuration). Next, look for one-way neighbor relationships. Is *C* in *A* and *B*'s tables, but are *A* and *B* not in *C*'s table? This could indicate a physical problem with *C*'s connection or a filter that is blocking EIGRP packets.

If the hold-time exceeds 15 seconds (or the configured hold-time), the network may be congested and losing hellos. Increasing the hello-interval/hold-time may be a quick fix to the problem.

The uptime should reflect the duration that the routers have been up. A low uptime indicates that the neighbor relationship is being lost and reestablished.

The QCnt should be 0 (or at least should not exceed 0 on a consistent basis).

In summary, if a problem is found in the neighbor relationship, you should do the following:

1. Check for bad physical infrastructure.
2. Ensure that router ports are plugged into the correct hubs.
3. Check for filters blocking EIGRP packets.
4. Verify router configurations—check IP addresses, masks, EIGRP AS numbers, and the network numbers defined under EIGRP.
5. Increase the hello-interval/hold-time on congested networks.

The command to clear and reestablish neighbor relationships is:

```
clear ip eigrp neighbors [ip address | interface]
```

Repeatedly clearing all neighbor relationships causes the loss of routes (and the loss of packets to those routes). Besides, repeatedly issuing *clear* commands usually does not fix the problem.

Stuck-in-Active

A route is regarded as stuck-in-active (SIA) when DUAL does not receive a response to a query from a neighbor for three minutes, which is the default value of the active timer. DUAL then deletes all routes from that neighbor, acting as if the neighbor had responded with an unreachable message for all routes.

Routers propagate queries through the network if feasible successors are not found, so it can be difficult to catch the culprit router (i.e., the router that is not responding to the query in time). The culprit may be running high on CPU utilization or may be connected via low-bandwidth links. Going back to TraderMary's network, when *NewYork* queries *Ames* for `172.16.50.0`, it marks the route as active and lists the neighbor from which it is expecting a reply (line 28):

```
NewYork#sh ip eigrp topology
IP-EIGRP Topology Table for process 10

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status

...
A 172.16.50.0/24, 0 successors, FD is 2195456, Q
    1 replies, active 00:00:06, query-origin: Local origin
    Remaining replies:
28         via 172.16.251.2, r, Serial1
```

If this route were to become SIA, the network engineer should trace the path of the queries to see which router has been queried, has no outstanding queries itself, and yet is taking a long time to answer.

Starting from *NewYork*, the next router to check for SIA routes would be `172.16.251.2` (line 28). Finding the culprit router in large networks is a difficult task, because queries fan out to a large number of routers. Checking the router logs would give a clue as to which router(s) had the SIA condition.

Increase active timer

Another option is to increase the active timer. The default value of the active timer is three minutes. If you think the SIA condition is occurring because the network diameter is too large, with several slow-speed links (such as Frame Relay PVCs), it is possible that increasing the active timer will allow enough time for responses to return. The following command shows how to increase the active timer:

```
router eigrp 10
timers active-time minutes
```

For the change to be effective, the active timer must be modified on every router in the path of the query.

EIGRP Bandwidth on Low-Speed Links

EIGRP limits itself to using no more than 50% of the configured bandwidth on router interfaces. There are two reasons for this:

1. Generating more traffic than the interface can handle would cause drops, thereby impairing EIGRP performance.
2. Generating a lot of EIGRP traffic would result in little bandwidth remaining for user data.

EIGRP uses the bandwidth that is configured on an interface to decide how much EIGRP traffic to generate. If the bandwidth configured on an interface does not match the physical bandwidth (the network architect may have put in an artificially low or high bandwidth value to influence routing decisions), EIGRP may be generating too little or too much traffic. In either case, EIGRP can encounter problems as a result of this. If it is difficult to change the *bandwidth* command on an interface because of such constraints, allocate a higher or lower percentage to EIGRP with the following command in interface configuration mode:

```
ip bandwidth percent eigrp AS-number percentage
```

Network Logs

Check the output of the *show logging* command for EIGRP/DUAL messages. For example, the following message:

```
%DUAL-3-SIA: Route XXX stuck-in-active state in IP-EIGRP
```

indicates that the route *XXX* was SIA.

IOS Version Check, Bug Lists

The EIGRP implementation was enhanced in IOS Releases 10.3(11), 11.0(8), and 11.1(3) with respect to its performance on Frame Relay and other low-speed networks. In the event of chronic network problems, check the IOS versions in use in your network. Also use the bug navigation tools available on the Cisco web site.

Debug Commands

As always, use *debug* commands in a production network only after careful thought. Having to resort to rebooting the router can be very unappetizing. The following is a list of EIGRP *debug* commands:

- *debug eigrp neighbors* (for neighbor-relationship activity)
- *debug eigrp packet* (all EIGRP packets)
- *debug eigrp ip neighbor* (if the previous two commands are used together, only EIGRP packets for the specified neighbor are shown)

Summing Up

EIGRP offers the following radical improvements over RIP and IGRP:

- Fast convergence—convergence is almost instantaneous when a feasible successor is available.
- Variable Length Subnet Masks are supported—subnet mask information is exchanged in EIGRP updates. This allows for efficient use of the address space, as well as support for discontiguous networks.
- Route summarization at arbitrary bit boundaries, reducing routing-table size.
- No regular routing updates—network bandwidth and router CPU resources are not tied up in periodic routing updates, leading to improved network manageability.
- Ease of configuration—EIGRP can be configured with almost the same ease as IGRP. However, troubleshooting DUAL can be difficult.

These EIGRP benefits come at the price of higher memory requirements (in addition to the routing table, EIGRP requires memory for the topology table and the neighbor table). DUAL is complex and can be very CPU-intensive, especially during periods of network instability when CPU resources are already scarce. Also, don't forget that the EIGRP is a Cisco proprietary protocol.

EIGRP is in use today in several mid-sized networks.