In this chapter:
- Getting IGRP Running
- How IGRP Works
- Speeding Up Convergence
- Route Summarization
- Default Routes
- Classful Route Lookups
- Summing Up

CHAPTER 3

Interior Gateway Routing Protocol (IGRP)

The second Distance Vector protocol that we will examine is the Interior Gateway Routing Protocol, or IGRP. IGRP and RIP are close cousins: both are based on the Bellman-Ford Distance Vector (DV) algorithms. DV algorithms propagate routing information from neighbor to neighbor; if a router receives the same route from multiple neighbors, it chooses the route with the lowest metric. All DV protocols need robust strategies to cope with *bad* routing information. Bad routes can linger in a network when information about the loss of a route does not reach some router (for instance, because of the loss of a route update packet), which then inserts the bad route back into the network. IGRP uses the same convergence strategies as RIP: triggered updates, route hold-downs, split horizon, and poison reverse.

IGRP has been widely deployed in small to mid-sized networks because it can be configured with the same ease as RIP, but its metric represents bandwidth and delay, in addition to hop count. The ability to discriminate between paths based on bandwidth and delay is a major improvement over RIP.

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

The following section gets us started with configuring IGRP.

Getting IGRP Running

TraderMary's network, shown in Figure 3-1, can be configured to run IGRP as follows.

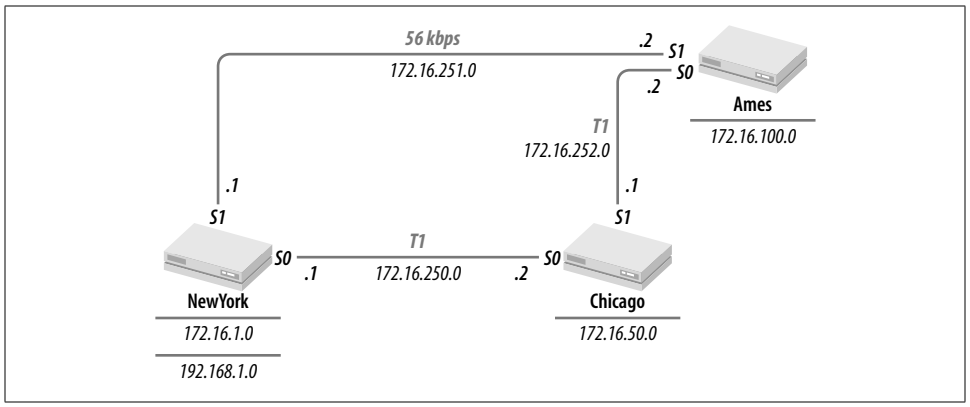


Figure 3-1. TraderMary's network

Like RIP, IGRP 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
ip address 172.16.251.1 255.255.255.0
...
router igrp 10
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 igrp 10
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 igrp 10
network 172.16.0.0
```

The syntax of the IGRP command is:

```
router igrp {process-id | autonomous-system-number}
```

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

```
network 172.16.0.0
```

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

1. *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 IGRP updates on its `172.16.0.0` interfaces from other routers running IGRP 10. For example, *NewYork* will receive IGRP 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 IGRP 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. Here is *NewYork*'s routing table:

```
NewYork#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set
```

```
     172.16.0.0/24 is subnetted, 6 subnets
I       172.16.252.0 [100/10476] via 172.16.251.2, 00:00:26, Serial1
                     [100/10476] via 172.16.250.2, 00:00:37, Serial0
C       172.16.250.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
I       172.16.50.0 [100/8576] via 172.16.250.2, 00:00:37, Serial0
C       172.16.1.0 is directly connected, Ethernet0
I       172.16.100.0 [100/8576] via 172.16.251.2, 00:00:26, Serial1
C    192.168.1.0/24 is directly connected, Ethernet1
```

Here is *Chicago*'s table:

```
Chicago#sh ip route
...
Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 6 subnets
C       172.16.252.0 is directly connected, Serial1
C       172.16.250.0 is directly connected, Serial0
I       172.16.251.0 [100/10476] via 172.16.250.1, 00:01:22, Serial0
                     [100/10476] via 172.16.252.2, 00:00:17, Serial1
C       172.16.50.0 is directly connected, Ethernet0
I       172.16.1.0 [100/8576] via 172.16.250.1, 00:01:22, Serial0
I       172.16.100.0 [100/8576] via 172.16.252.2, 00:00:17, Serial1
```

And here is *Ames*'s table:

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

     172.16.0.0/24 is subnetted, 6 subnets
C       172.16.252.0 is directly connected, Serial0
I       172.16.250.0 [100/10476] via 172.16.251.1, 00:01:11, Serial1
                     [100/10476] via 172.16.252.1, 00:00:21, Serial0
C       172.16.251.0 is directly connected, Serial1
I       172.16.50.0 [100/8576] via 172.16.252.1, 00:00:21, Serial0
I       172.16.1.0 [100/8576] via 172.16.251.1, 00:01:11, Serial1
C       172.16.100.0 is directly connected, Ethernet0
```

The IGRP-derived routes in these tables are labeled with an "I" in the left margin. The first line in each router's table contains summary information:

```
172.16.0.0/24 is subnetted, 6 subnets
```

Note that all three routers show the same summary information—*NewYork*, *Chicago*, and *Ames* show all six subnets.

Note also 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 IGRP configuration on *NewYork*:

```
hostname NewYork
...
```

```
router igrp 10
network 172.16.0.0
network 192.168.1.0
```

Getting IGRP started is fairly straightforward. However, if you compare the routing tables in this section to those in the previous chapter on RIP, there is no difference in the next-hop information. More importantly, the route from *NewYork* to network `172.16.100.0` is still over the direct 56-kbps path rather than the two-hop T-1 path. The two-hop T-1 path is better than the one-hop 56-kbps link. As an example, take a 512-byte packet; it would take 73 ms to copy this packet over a 56-kbits/s link versus 5 ms over two T-1 links. Our expectation is that IGRP should install this two-hop T-1 path, since IGRP has been touted for its metric that includes link bandwidth and delay. The later section "IGRP Metric" explains why IGRP installs the slower path. The "Modifying IGRP metrics" section leads us through the configuration changes required to make IGRP install the faster path.

A key difference in this configuration is that, unlike in RIP, each IGRP process is identified by an autonomous system (AS) number. AS numbers are described in detail in the next section.

How IGRP Works

Since IGRP is such a close cousin of RIP, we will not repeat the details of how DV algorithms work, how updates are sent, and how route convergence is achieved. However, because IGRP employs a much more comprehensive metric, I'll discuss the IGRP metric in detail. I'll begin this discussion with AS numbers.

IGRP Autonomous System Number

Each IGRP process requires an autonomous system number:

```
router igrp autonomous-system-number
```

The AS number allows the network administrator to define routing domains; routers within a domain exchange IGRP routing updates with each other but not with routers in different domains. Note that in the context of IGRP the terms "autonomous system number" and "process ID" are often used interchangeably. Since the IGRP autonomous system number is not advertised to other domains, network engineers often cook up arbitrary process IDs for their IGRP domains.

Let's say that TraderMary created a subsidiary in Africa and that the new topology is as shown in Figure 3-2.

Note that IGRP is running in the U.S. and Africa with AS numbers of 10 and 20, respectively. The U.S. routers now exchange IGRP routes with each other, as before, and the routers *Nairobi* and *Casablanca* exchange IGRP updates with each other. IGRP updates are processed only between routers running the same AS number, so

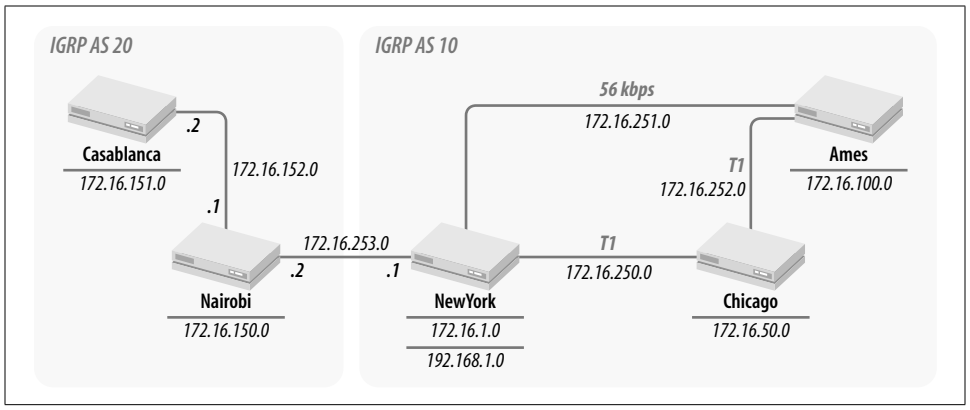


Figure 3-2. TraderMary's U.S. and African networks

NewYork and *Nairobi* do not exchange IGRP updates with each other. We will see this in more detail later, when we look at the format of an IGRP update.

The advantage of creating small domains with unique AS numbers is that a routing problem in one domain is not likely to ripple into another domain running a different AS number. So, for example, let's say that a network engineer in Africa configured network `172.16.50.0` on *Casablanca* (`172.16.50.0` already exists on *Chicago*). The U.S. network would not be disrupted because of this duplicate address. In another situation, an IGRP bug in IOS on *Chicago* could disrupt routing in the U.S., but *Nairobi* and *Casablanca* would not be affected by this problem in the AS 10.

The problem with creating too many small domains running different IGRP AS numbers is that sooner or later the domains will need to exchange routes with each other. The office in New York would need to send files to Nairobi. This could be accomplished by adding static routes on *NewYork* (to `172.16.150.0`) and *Nairobi* (to `172.16.1.0`). However, static routes can be cumbersome to install and administer and do not offer the redundancy of dynamic routing protocols. Dynamic distribution of routes between routing domains is discussed in Chapter 8.

In the meantime, all I will say is to use good judgment when breaking networks into autonomous systems. Making a routing domain too small will require extensive redistributions or the creation of static entries. Making a routing domain too big exposes the network to failures of the type just described.

The boundary between domains is often geographic or organizational.

IGRP Metric

The RIP metric was designed for small, homogenous networks. Paths were selected based on the number of hops to a destination; the lowest hop-count path was installed in the routing table. IGRP is designed for more complex networks. Cisco's implementation of IGRP allows the network engineer to customize the metric based

on bandwidth, delay, reliability, load, and MTU. In order to compare metrics between paths and select the least-cost path, IGRP converts bandwidth, delay, reliability, delay, and MTU into a scalar quantity—a *composite* metric that expresses the desirability of a path. Just as in the case of RIP, a path with a lower composite metric is preferred to a path with a higher composite metric.

The computation of the IGRP composite metric is user-configurable; i.e., the network administrator can specify parameters in the *formula* used to convert bandwidth, delay, reliability, load, and MTU into a scalar quantity.

The following sections define bandwidth, delay, reliability, load, and MTU. We will then see how these variables can be used to compute the composite metric for a path.

Interface bandwidth, delay, reliability, load, and MTU

The IGRP metric for a path is derived from the bandwidth, delay, reliability, load, and MTU values of every media in the path to the destination network.

The bandwidth, delay, reliability, load, and MTU values at any interface to a media can be seen as output of the *show interface* command:

```
router1#sh interface ethernet 0
Ethernet0 is up, line protocol is up
  Hardware is AmdP2, address is 0010.7bcf.e340 (bia 0010.7bcf.e340)
  Description: Lab Test
  Internet address is 1.13.96.1/16
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
...
```

The bandwidth, delay, reliability, load, and MTU for a media are defined as follows:

Bandwidth
: The bandwidth for a link represents how fast the physical media can transmit bits onto a wire. Thus, an HSSI link transmits approximately 45,000 kbits every second, Ethernet runs at 10,000 kbps, a T-1 link transmits 1,544 kbits every second, and a 56-kbps link transmits 56 kbits every second.

  Ethernet0 on *router1* is configured with a bandwidth of 10,000 kbps.

Delay
: The delay for a link represents the time to traverse the link in an unloaded network and includes the propagation time for the media. Ethernet has a delay value of 1 ms; a satellite link has a delay value in the neighborhood of 1 second.

  Ethernet0 on *router1* is configured with a delay of 1,000 ms.

Reliability
: Link reliability is dynamically measured by the router and is expressed as a numeral between 1 and 255. A reliability of 255 indicates a 100% reliable link.

  Ethernet0 on *router1* is 100% reliable.

Load
: Link utilization is dynamically measured by the router and is expressed as a numeral between 1 and 255. A load of 255 indicates 100% utilization.

  Ethernet0 on *router1* has a load of 1/255.

MTU
: The MTU, or Maximum Transmission Unit, represents the largest frame size the link can handle.

  Ethernet0 on *router1* has an MTU size of 1,500 bytes.

The MTU, bandwidth, and delay values are static parameters that Cisco routers derive from the media type. Table 3-1 shows some common values for bandwidth and delay. These default values can be modified using the commands shown in the next section.

Table 3-1. Default bandwidth and delay values

Media type	Default bandwidth	Default delay
Ethernet	10 Mbps	1,000 ms
Fast Ethernet	100 Mbps	100 ms
FDDI	100 Mbps	100 ms
T-1 (serial interface)[a]	1,544 kbps	20,000 ms
56 kbps (serial interface)	1,544 kbps	20,000 ms
HSSI	45,045 kbps	20,000 ms

[a] All serial interfaces on Cisco routers are configured with the same *default* bandwidth (1,544 kbits/s) and delay (20,000 ms) parameters.

The reliability and load values are dynamically computed by the router as five-minute exponentially weighted averages.

Modifying interface bandwidth, delay, and MTU

The default bandwidth and delay values may be overridden by the following interface commands:

```
bandwidth kilobits
delay tens-of-microseconds
```

So, the following commands will define a bandwidth of 56,000 bps and a delay of 10,000 ms on interface *Serial0*:

```
interface Serial0
bandwidth 56
delay 1000
```

These settings affect only IGRP routing parameters. The actual physical characteristics of the interface—the clock-rate on the wire and the media delay—have no relationship to the bandwidth or delay values configured as in this example or seen as

output of the *show interface* command. Thus, interface *Serial0* in the previous example may actually be clocking data at 128,000 bps, a rate that will be governed by the configuration of the modem or the CSU/DSU attached to *Serial0*. Note that, by default, Cisco sets the bandwidth and delay on all serial interfaces to be 1,544 kbps and 20,000 ms, respectively (see Table 3-1).

Note that delay on an interface is specified in *tens of microseconds*. Thus:

```
delay 1000
```

describes a delay of 10,000 ms.

The MTU on an interface can be modified using the following command:

```
mtu bytes
```

However, the MTU size has no bearing on IGRP route selection. The MTU size should not be modified to affect routing behavior. The default MTU values represent the maximum allowed for the media type; lowering the MTU size can impair performance by causing needless fragmentation of IP datagrams.

Later in this chapter we will see how modifications to the bandwidth and delay parameters on an interface can affect route selection.

IGRP routing update

IGRP updates are directly encapsulated in IP with the protocol field (in the IP header) set to 9. The format of an IGRP packet is shown in Figure 3-3.

Just like RIP, IGRP allows a station to request routes. This allows a router that has just booted up to request the routing table from its neighbors instead of waiting for the next cycle of updates, which could be as much as 90 seconds later for IGRP.

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

Each update packet contains three types of routes:

Interior routes
: Contain subnet information for the major network number associated with the address of the interface to which the update is being sent. If the IGRP update is being sent on a broadcast network, the internal routes are subnet numbers from the same major network number that is configured in the broadcast media.

System routes
: Contain major network numbers that may have been summarized when a network-number boundary was crossed.

Exterior routes
: Represent candidates for the default route. Unlike RIP, which uses `0.0.0.0` to represent the default, IGRP uses specific network numbers as candidates for the default by tagging the routes as exterior.

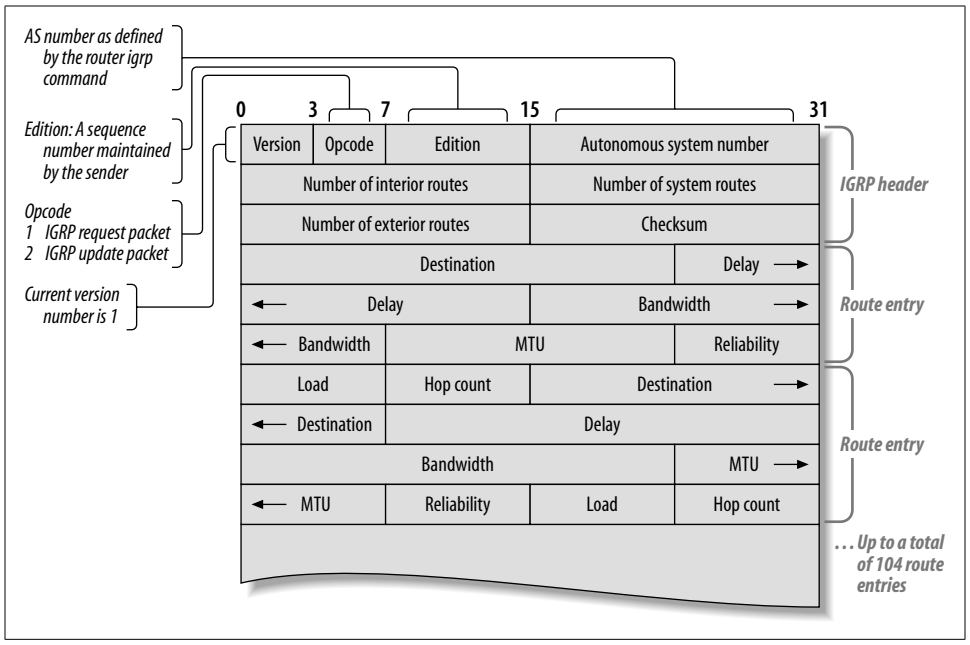


Figure 3-3. Format of an IGRP update packet

Interior, system, and exterior routes appear in order in each update packet. The count of interior, system, and exterior routes identifies the route type for each route entry.

Note that the IGRP update has only three octets for the destination network-number field, whereas IP addresses are four octets in length. IGRP extracts the four-octet IP address using the heuristic shown in Table 3-2.

Table 3-2. Deriving the four-octet IP destination address from the three-octet destination field

Route type	Heuristic to derive four-octet IP destination address
Interior route	The first octet is derived from the IP address of the interface that received the update; the last three octets are derived from the IGRP update.
System route	The route is assumed to have been summarized. The last octet of the IP destination address is 0.
Exterior route (default route)	The route is assumed to have been summarized. The last octet of the IP destination address is 0.

Just like RIP, IGRP updates do not contain subnet mask information. This classifies both RIP and IGRP as *classful* routing protocols. Subnet mask information for routes received in IGRP updates is derived using the same rules as in RIP.

When an update is received for a route, it contains the bandwidth, delay, reliability, load, and MTU values for the path to the destination network via the source of the update. I already defined bandwidth, delay, reliability, load, and MTU for an interface. Now let's define these parameters again for a path.

Path bandwidth, delay, reliability, load, and MTU

The following list defines bandwidth, delay, reliability, load, and MTU for a path:

Bandwidth
: The bandwidth for a path is the minimum bandwidth in the path to the destination network. Compare a network to a sequence of pipes for the transmission of a fluid; the slowest pipe (or the thinnest pipe) will dictate the rate of flow of the fluid. Thus, if a path to a network is through an Ethernet segment, a T-1 line, and another Ethernet segment, the path bandwidth will be 1,544 kbps (see Table 3-1).

Delay
: The delay for a path is the sum of all delay values on the path to the destination network. The IGRP unit of delay is in tens of microseconds. A path through a network via an Ethernet segment, a T-1 line, and another Ethernet segment will have a path delay of 22,000 ms or 2,200 IGRP delay units (see Table 3-1).

  The IGRP update packet has three octets to represent delay (in units of tens of microseconds). The largest value of delay that can be represented is $2^{24} \times 10$ ms, which is roughly 167.7 seconds. 167.7 seconds is thus the maximum possible delay value for an IGRP network. All ones in the delay field are also used to indicate that the network indicated is *unreachable*.

Reliability
: The reliability for a path is the reliability of the least reliable link in the path.

Load
: The load for a path is the load on the most heavily loaded link in the path.

MTU
: The MTU represents the smallest MTU along the path. MTU is currently not used in computing the metric.

Note that, in addition to these parameters, the update packet includes the hop count to the destination. The default maximum hop count for IGRP is 100. This default can be modified with the command:

```
metric maximum-hops hops
```

The maximum value for *hops* is 255. A network with a diameter over 100 is very large indeed, especially for a network running IGRP. Do not expect to modify the maximum hop count for IGRP, even if you are working for an interstellar ISP. Large networks will usually require routing features that do not exist in IGRP.

The bandwidth, delay, reliability, load, and MTU values for the path selected by a router can be seen as output of the *show ip route* command:

```
NewYork#show ip route 172.16.100.0
Routing entry for 172.16.100.0 255.255.255.0
  Known via "igrp 10", distance 100, metric 8576
  Redistributing via igrp 10
  Advertised by igrp 10 (self originated)
```

```
Last update from 172.16.251.2 on Serial1, 00:00:29 ago
Routing Descriptor Blocks:
* 172.16.251.2, from 172.16.251.2, 00:00:29 ago, via Serial1
    Route metric is 8576, traffic share count is 1
    Total delay is 21000 microseconds, minimum bandwidth is 1544 Kbit
    Reliability 255/255, minimum MTU 1500 bytes
    Loading 1/255, Hops 2
```

IGRP composite metric

The path metric of bandwidth, delay, reliability, load, and MTU needs to be expressed as a composite metric for you to be able to compare paths. The default behavior of Cisco routers considers only bandwidth and delay in computing the composite metric (the parameters reliability, load, and MTU are ignored):

Metric = *BandW* + *Delay*

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). For example, if the path from a router to a destination Ethernet segment is via a T-1 link, then:

BandW = 10,000,000/1,544 = 6,476

Delay is computed by adding the delays from all outgoing interfaces to the destination (including the delay on the interface connecting to the destination network) and dividing by 10:

Delay = (20,000 + 1,000)/10 = 2,100

And then the composite metric for the path to the Ethernet segment would be:

Metric = *BandW* + *Delay* = 1,000 + 2,100 = 3,100

Let's now go back to TraderMary's network to see why router *NewYork* selected the direct 56-kbps link to route to `172.16.100.0` and not the two-hop T-1 path via *Chicago*:

```
NewYork>sh ip route
...
I       172.16.100.0 [100/8576] via 172.16.251.2, 0:00:31, Serial0
...
```

The values of the IGRP metrics for these paths can be seen here:

```
Ames#sh interface Ethernet 0
Ethernet0 is up, line protocol is up
```

* The concept of an *outgoing* interface is best illustrated with an example. In TraderMary's network, the outgoing interfaces from *NewYork* to `172.16.100.0` will be *NewYork* interface *Serial0*, *Chicago* interface *Serial*, and *Ames* interface *Ethernet0*. When computing the metric for *NewYork* to `172.16.100.0`, we will use the IGRP parameters of bandwidth, delay, load, reliability, and MTU for these interfaces. We will not use the IGRP parameters from interfaces. However, unless they have been modified, the parameters on this second set of interfaces would be identical to the first.

```
  Hardware is Lance, address is 00e0.b056.1b8e (bia 00e0.b056.1b8e)
  Description: Lab Test
  Internet address is 172.16.100.1/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
...

NewYork# show interfaces serial 0
Serial 0 is up, line protocol is up
Hardware is MCI Serial
Internet address is 172.16.250.1, subnet mask is 255.255.255.0
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
Encapsulation HDLC, loopback not set, keepalive set (10 sec)
...
```

There are two paths to consider:

1. *NewYork* → *Ames* → `172.16.100.0`.

   Bandwidth values in the path: (serial link) 1,544 kbits/s, (Ethernet segment) 10,000 kbits/s

   Delay values in the path: (serial link) 2,000, (Ethernet segment) 100

   Smallest bandwidth in the path: 1,544

   BandW = 10,000,000/1,544 = 6,476
   Delay = 2,000 + 100 = 2,100
   Metric = *BandW* + *Delay* = 8,576

2. *NewYork* → *Chicago* → *Ames* to `172.16.100.0`.

   Bandwidth values in the path: (serial link) 1,544 kbits/s, (serial link) 1,544 kbits/s, (Ethernet segment) 10,000 kbits/s

   Delay values in the path: (serial link) 2,000, (serial link) 2,000, (Ethernet segment) 100

   Smallest bandwidth in the path: 1,544

   BandW = 10,000,000/56 = 6,476
   Delay = 2,000 + 2,000 + 100 = 4,100
   Metric = *BandW* + *Delay* = 10,576

NewYork will prefer to route via the first path because the metric is smaller. Why does *NewYork* use a bandwidth of 1,544 for the 56-kbps link to *Ames*? Go back to Table 3-1 and you will see that the default bandwidth and delay values of 1,544 kbps and 20,000 ms apply to all serial interfaces, regardless of the speed of the modem device attached to the router port.

The IGRP metric can be customized to use reliability and load with the following formula (Equation 1):

$$Metric = k1 \times BandW + k2 \times BandW/(256 - load) + k3 \times Delay$$

where the default values of the constants are k1 = k3 = 1 and k2 = k4 = k5 = 0.

If k5 is not equal to zero, an additional operation is done:

$$Metric = Metric \times [k5/(reliability + k4)]$$

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

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

where *tos* identifies the type of service and must be set to zero (because only one type of service has been defined).

Plugging the default values of k1, k2, k3, k4, and k5 into Equation 1 yields:

$$Metric = BandW + Delay$$

which we saw earlier.

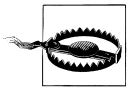
To make the metric sensitive to the network load (in addition to bandwidth and delay), set k1 = k2 = k3 = 1 and k4 = k5 = 0. This yields:

$$Metric = BandW + BandW/(256 - load) + Delay$$

The problem with using load in the metric computation is that it can make a route unstable. For example, a router may select a path through router *P* as its next hop to reach a destination. When the load on the path through *P* rises, in a few minutes (the value of load is computed as a five-minute exponentially weighted average) the metric for the path through *P* may become larger than the metric for an alternative path through router *Q*. The traffic then shifts to *Q*; this causes the load to increase on the path through *Q* and the path through *P* becomes more attractive. Thus, setting k2 = 1 can make a route unstable and cause traffic to bounce between two paths. Further, abrupt changes in metric cause flash updates; the route may also go into hold-down.

Instead of selecting the best path based on load, you may consider load balancing over several paths. Load balancing occurs automatically over equal-cost paths. If two or more paths have slightly different metrics, you may consider modifying the bandwidth and delay parameters to make the metrics equal and to utilize all the paths. See the example on modifying bandwidth and delay parameters in the next section.

To make the metric sensitive to network reliability (in addition to bandwidth and delay), set k1 = k3 = k5 =1 and k2 = k4 = 0. In the event of link errors, this will cause the metric on the path to increase, and IGRP will select an alternative path when the metric has worsened enough. A typical action in today's networks is to turn a line down until the transmission problem is resolved, not to base routing decisions on how badly the line is running.



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

Modifying IGRP metrics

TraderMary's network was still using the 56-kbps path between *NewYork* and *Ames*, even when IGRP was running on the routers (refer to "Getting IGRP Running"). Why is it that *NewYork* and *Ames* did not pick up the lower bandwidth for the 56-kbps link?

Table 3-1 contains the key to our question. All serial interfaces on a Cisco router are configured with the same bandwidth (1,544 kbps) and delay (20,000 ms) values. Thus, IGRP sees the 56-kbps line with the same bandwidth and delay parameters as a T-1 line.

In order to utilize the 56-kbps link only as backup, we need to modify TraderMary's network as follows:

```
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 igrp 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 igrp 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 igrp 10
network 172.16.0.0
```

The new routing tables look like this:

```
NewYork#show ip route
...
Gateway of last resort is 0.0.0.0 to network 0.0.0.0

     172.16.0.0/24 is subnetted, 6 subnets
I       172.16.252.0 [100/10476] via 172.16.250.2, 00:00:43, Serial0
C       172.16.250.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
I       172.16.50.0 [100/8576] via 172.16.250.2, 00:00:43, Serial0
C       172.16.1.0 is directly connected, Ethernet0
I       172.16.100.0 [100/10576] via 172.16.250.2, 00:00:43, Serial0
C    192.168.1.0/24 is directly connected, Ethernet1


Chicago#sh ip route
...
Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 6 subnets
C       172.16.252.0 is directly connected, Serial1
C       172.16.250.0 is directly connected, Serial0
I       172.16.251.0 [100/182571] via 172.16.250.1, 00:00:01, Serial0
                     [100/182571] via 172.16.252.2, 00:01:01, Serial1
C       172.16.50.0 is directly connected, Ethernet0
I       172.16.1.0 [100/8576] via 172.16.250.1, 00:00:01, Serial0
I       172.16.100.0 [100/8576] via 172.16.252.2, 00:01:01, Serial1


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

     172.16.0.0/24 is subnetted, 6 subnets
C       172.16.252.0 is directly connected, Serial0
I       172.16.250.0 [100/10476] via 172.16.252.1, 00:00:24, Serial0
```

```
C       172.16.251.0 is directly connected, Serial1
I       172.16.50.0 [100/8576] via 172.16.252.1, 00:00:24, Serial0
I       172.16.1.0 [100/10576] via 172.16.252.1, 00:00:24, Serial0
C       172.16.100.0 is directly connected, Ethernet0
```

Let's now go back to TraderMary's network and corroborate the metric values seen for `172.16.100.0` in router *NewYork's* routing table. The following calculations show TraderMary's network as in Figure 3-1 but with IGRP bandwidth and delay values for each interface. There are two paths to consider:

1. *NewYork* → *Ames* → `172.16.100.0`.

   Bandwidth values in the path: (serial link) 56 kbits/s, (Ethernet segment) 10,000 kbits/s

   Smallest bandwidth in the path: 56

   BandW = 10,000,000/56 = 178,571
   Delay = 2,000 + 100 = 2100
   Metric = *BandW* + *Delay* = 180,671

2. *NewYork* → *Chicago* → *Ames* → `172.16.100.0`

   Bandwidth values in the path: (serial link) 1,544 kbits/s, (serial link) 1,544 kbits/s, (Ethernet segment) 10,000 kbits/s

   Smallest bandwidth in the path: 1,544

   BandW = 10,000,000/1,544 = 6,476
   Delay = 2,000 + 2,000 + 100 = 4,100
   Metric = *BandW* + *Delay* = 10,576

Using the lower metric for the path via *Chicago*, *NewYork*'s route to `172.16.100.0` shows as:

```
NewYork>sh ip route
...
I       172.16.50.0 [100/1] via 172.16.250.2, 0:00:31, Serial0
I       172.16.100.0 [100/10576] via 172.16.250.2, 0:00:31, Serial0
I       172.16.252.0 [100/1] via 172.16.250.2, 0:00:31, Serial0
```

Let's corroborate IGRP's selection of the two-hop T-1 path in preference to the one-hop 56-kbps link by comparing the transmission delay for a 1,000-octet packet. A 1,000-octet packet will take 143 ms (1,000×8/56,000 second) over a 56-kbps link and 5 ms (1,000×8/1,544,000 second) 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.

Processing IGRP updates

The processing of IGRP updates is very similar to the processing of RIP updates, described in the previous chapter. The IGRP update comes with an autonomous system number. If this does not match the IGRP AS number configured on the router receiving the update, the entire upgrade is disregarded. Thus, routers *NewYork* and

Nairobi in TraderMary's network will receive updates from each other but will discard them.

Each network number received in the update is checked for validity. Illegal network numbers such as `0.0.0.0/8`, `127.0.0.0/8`, and `128.0.0.0/16` are sometimes referred to as "Martian Network Numbers" and will be disregarded when received in an update (RFCs 1009, 1122).

The rules for processing IGRP updates are:

1. If the destination network number is unknown to the router, install the route using the source IP address of the update (provided the route is not indicated as unreachable).
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, install the route as long as the maximum number of paths to the same destination is not exceeded. These parallel paths are then used for load balancing. Note that the default maximum number of paths to a single destination is six in IOS Releases 11.0 or later.

Parallel Paths

For the routing table to be able to install multiple paths to the same destination, the IGRP metric for all the paths must be equal. The routing table will install several parallel paths to the same destination (the default maximum is six in current releases of IOS).

Load-sharing over parallel paths depends on the switching mode. If the router is configured for *process switching*, load balancing will be on a packet-by-packet basis. If the router is configured for *fast switching*, load balancing will be on a per-destination basis. For a more detailed discussion of switching mode and load balancing, see Chapter 2.

Unequal metric (cost) load balancing

The default behavior of IGRP installs parallel routes to a destination only if all routes have identical metric values. Traffic to the destination is load-balanced over all installed routes, as described earlier.

Equal-cost load balancing works well almost all the time. However, consider TraderMary's network again. Say that TraderMary adds a node in London. Since traffic to London is critical, the network is engineered with two links from New York: one running at 128 kbps and another running at 56 kbps. Figure 3-4 shows unequal-cost load balancing.

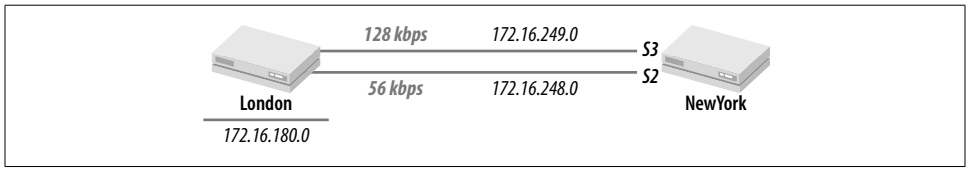


Figure 3-4. Unequal-cost load balancing

The routers are first configured as follows:

```
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 Serial2
bandwidth 128
ip address 172.16.249.1 255.255.255.0
!
interface Serial3
bandwidth 56
ip address 172.16.248.1 255.255.255.0
...
router igrp 10
network 172.16.0.0


hostname London
...
interface Ethernet0
ip address 172.16.180.1 255.255.255.0
!
interface Serial0
bandwidth 128
ip address 172.16.249.2 255.255.255.0
!
interface Serial1
bandwidth 56
ip address 172.16.284.2 255.255.255.0
...
router igrp 10
network 172.16.0.0
```

However, if you check *NewYork*'s routing table you will see that all traffic to London is being routed via the 128-kbps link:

```
NewYork>sh ip route
...
172.16.0.0/24 is subnetted, ...
I       172.16.180.0 [100/80225] via 172.16.249.2, 00:01:07, Serial2
...
```

This is because the *NewYork* → *London* metric is 80,225 via the 128-kbps path and 180,671 via the 56-kbps path.

The problem with this routing scenario is that the 56-kbps link is entirely unused, even when the 128-kbps link is congested. Overseas links are expensive: the network design ought to try to utilize all links. One way around this problem is to modify the IGRP parameters to make both links look equally attractive. This can be accomplished by modifying the 56-kbps path as follows:

```
hostname NewYork
...
interface Serial3
bandwidth 128
ip address 172.16.248.1 255.255.255.0
...
```

With this approach, both links would appear equally attractive. The routing table for *NewYork* will look like this:

```
NewYork>sh ip route
...
172.16.0.0/24 is subnetted, ...
I       172.16.180.0 [100/80225] via 172.16.249.2, 00:01:00, Serial2
                     [100/80225] via 172.16.248.2, 00:01:00, Serial3
```

However, traffic will now be evenly distributed over the two links, which may congest the 56-kbps link while leaving the 128-kbps link underutilized.

Another solution is to modify IGRP's default behavior and have it install unequal-cost links in its table, balancing traffic over the links in proportion to the metrics on the links. The variance that is permitted between the lowest and highest metrics is specified by an integer in the *variance* command. For example:

```
router igrp 10
network 172.16.0.0
variance 2
```

specifies that IGRP will install routes with different metrics as long as the largest metric is less than twice the lowest metric. In other words, if the variance is v, then:

$$\text{highest metric} \geq \text{lowest metric} \times v$$

The maximum number of routes that IGRP will install will still be four, by default. This maximum can be raised to six when running IOS 11.0 or later.

Going back to TraderMary's network, the metric value for the 128-kbps path to London is 80,225 while the metric value for the 56-kbps path is 180,671. The ratio 180,671/80,225 is 2.25; hence, a variance of 3 will be adequate. *NewYork* may now be configured as follows:

```
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 Serial2
bandwidth 128
ip address 172.16.249.1 255.255.255.0
!
interface Serial3
bandwidth 56
ip address 172.16.248.1 255.255.255.0
...
router igrp 10
network 172.16.0.0
variance 3
```

And the routing table for *NewYork* will look like this:

```
NewYork>sh ip route
...
172.16.0.0/24 is subnetted, ...
I       172.16.180.0 [100/80225] via 172.16.249.2, 00:01:00, Serial2
                     [100/180671] via 172.16.248.2, 00:01:00, Serial3
```

Traffic from *NewYork* to *London* will be divided between *Serial2* and *Serial3* in the inverse ratio of their metrics: *Serial2* will receive 2.25 times as much traffic as *Serial3*.

The default value of variance is 1. A danger with using a variance value of greater than 1 is the possibility of introducing a routing loop. Thus, *NewYork* may start routing to *London* via *Chicago* if the variance is made sufficiently large. IGRP checks that the paths it chooses to install are always downstream (toward the destination) by choosing only next hops with lower metrics to the destination.

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. This output shows the values of the IGRP timers:

```
NewYork#sh ip protocol
Routing Protocol is "igrp 10"
```

```
    Sending updates every 90 seconds, next due in 61 seconds
    Invalid after 270 seconds, hold down 280, flushed after 630
    Outgoing update filter list for all interfaces is
    Incoming update filter list for all interfaces is
    Default networks flagged in outgoing updates
    Default networks accepted from incoming updates
    IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
    IGRP maximum hopcount 100
    IGRP maximum metric variance 1
    Redistributing: igrp 10
    Routing for Networks:
      172.16.0.0
    Routing Information Sources:
      Gateway         Distance      Last Update
1     172.16.250.2         100      00:00:40
2     172.16.251.2         100      00:00:09
    Distance: (default is 100)
```

Note that IGRP updates are sent every 90 seconds and the next update is due in 61 seconds, which means that an update was issued about 29 seconds ago.

Further, lines 1 and 2 show the gateways from which router *NewYork* has been receiving updates. This list is valuable in troubleshooting—missing routes from a routing table could be because the last update from a gateway was too long ago. Check the time of the last update to ensure that it is within the IGRP update timer:

```
NewYork#show ip route
...
Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 6 subnets
I       172.16.252.0 [100/10476] via 172.16.251.2, 00:00:26, Serial1
                     [100/10476] via 172.16.250.2, 00:00:37, Serial0
C       172.16.250.0 is directly connected, Serial0
C       172.16.251.0 is directly connected, Serial1
I       172.16.50.0 [100/8576] via 172.16.250.2, 00:00:37, Serial0
C       172.16.1.0 is directly connected, Ethernet0
I       172.16.100.0 [100/8576] via 172.16.251.2, 00:00:26, Serial1
C    192.168.1.0/24 is directly connected, Ethernet1
```

One key area to look at in the routing table is the timer values. The format that Cisco uses for timers is *hh:mm:ss* (hours:minutes:seconds). You would expect the time against each route to be between 00:00:00 (0 seconds) and 00:01:30 (90 seconds). If a route was received more than 90 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.

You should also be familiar with the number of major network numbers (two in the previous output—`172.16.0.0` and `192.168.1.0`) and the number of subnets in each (six in `172.16.0.0` and one in `192.168.1.0`). In most small to mid-sized networks, these counts will change only when networks are added or subtracted.

Speeding Up Convergence

Like RIP, IGRP implements hold-downs, split horizon, triggered updates, and poison reverse (see Chapter 2 for details on these convergence methods). Like RIP, IGRP also maintains an update timer, an invalid timer, a hold-down timer, and a flush timer for every route in the routing table:

Update timer (default value: 90 seconds)
: After sending a routing update, IGRP sets the update timer to 0. When the timer expires, IGRP issues another routing update.

Invalid timer (default value: 270 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 the source of the routing information is 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, IGRP would process any updates received by updating the route's timers.

Hold-down timer (default value: 280 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.

Flush timer (default value: 630 seconds)
: Every time a router receives an update for a route, it sets the flush timer to 0. When the flush timer expires, the route is removed from the routing table and the router is ready to receive a new route update. Note that the flush timer overrides the hold-down timer.

Setting Timers

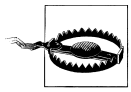
IGRP timers can be modified to allow faster convergence. The configuration:

```
router igrp 10
timers basic 30 90 90 180
```

would generate IGRP updates every 30 seconds, mark a route invalid in 90 seconds, keep the route in hold-down for 90 seconds, and flush the route in 180 seconds.

However, IGRP timers should not be modified without a detailed understanding of route convergence in Distance Vector protocols (see Chapter 2). Selecting too short a hold-down period, for example, may cause bad routing information to persist in a network. Selecting too long a hold-down period would increase the time it takes to learn a route via a different path after a failure.

Changing timers also presents the danger that sooner or later someone will configure a router with default timers. This may cause *route flapping*; i.e., routes to some network numbers may become intermittently invisible.



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

Disabling IGRP Hold-Downs

IGRP hold-downs can be disabled with the command:

```
router igrp 10
no metric holddown
```

thus speeding up convergence when a route fails. However, the problem with turning off hold-downs is that if a triggered update regarding the failure does not reach some router, that router could insert bad routing information into the network. Doesn't this seem like a dangerous thing to do?

Split horizon, triggered updates, and poison reverse are implemented in IGRP much like they are in RIP.

Route Summarization

IGRP summarizes network numbers when crossing a major network-number boundary, just like RIP does. Route summarization reduces the number of routes that need to be exchanged, processed, and stored.

However, route summarization does not work well in discontiguous networks. Consider the discontiguous network in Figure 3-5. Router *X* will receive advertisements for `10.0.0.0` from both routers *A* and *B*. If *X* sent packets with the destination `10.1.1.1` to *B*, the packet would be lost—*B* would have to drop the packet because it would not have a route for `10.1.1.1` in its table. Likewise, if *X* sent packets with the destination `10.2.1.1` to *A*, the packet would be lost—*A* would have to drop the packet because it would not have a route for `10.2.1.1`.

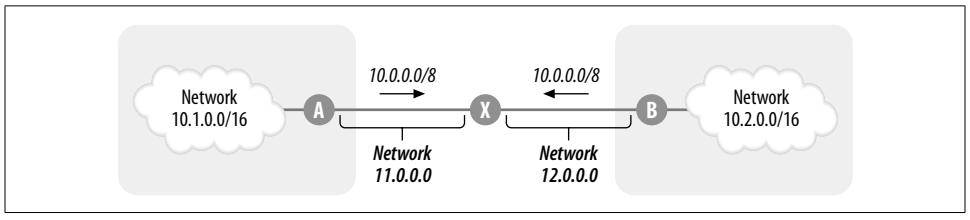


Figure 3-5. Contiguous and discontiguous networks

Both IGRP and RIP networks must be designed in contiguous blocks of major network numbers.

Default Routes

IGRP tracks default routes in the exterior section of its routing updates. A router receiving `10.0.0.0` in the exterior section of a routing update would mark `10.0.0.0` as a default route and install its next hop to `10.0.0.0` as the *gateway of last resort*. Consider the network in Figure 3-6 as an example in which a core router connects to several branch routers in remote sites.

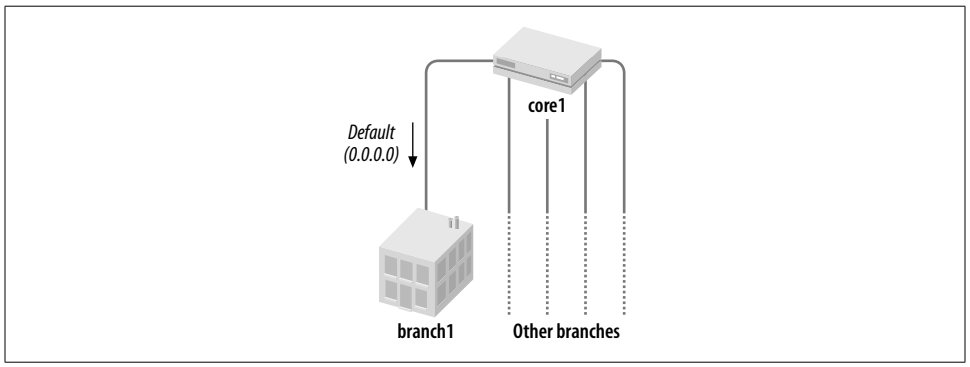


Figure 3-6. Branch offices only need a default route

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 igrp 10
 redistribute static
 network 172.16.0.0
 default-metric 10000 100 255 1 1500
!
no ip classless
ip default-network 10.0.0.0
ip route 10.0.0.0 255.0.0.0 Null0
```

The branch router is configured as follows:

```
hostname branch1
...
```

```
interface Serial0
ip address 172.16.245.2 255.255.255.0
...
router igrp 10
redistribute static
network 172.16.0.0
!
no ip classless
```

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

```
branch1#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 172.16.245.1 to network 10.0.0.0

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.245.0 is directly connected, Serial0
I*   10.0.0.0/8 [100/8576] via 172.16.245.1, 00:00:26, Serial0
```

Note that network 10.0.0.0 has been flagged as a default route (*). To ensure that the default route works, let's do a test to see if *branch1* can ping 192.168.1.1, even though 192.168.1.0 is not in *branch1*'s routing table:

```
branch1#ping 192.168.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/50/80 ms
```

Here are the steps we followed in the creation of the default route:

1. Network 10.0.0.0 was flagged as a default route by *core1* (line 5).
2. Network 10.0.0.0 was defined via a static route (line 6).
3. The default route was redistributed into IGRP, which then placed the route in the exterior section of its update message to *branch1* (line 3).
4. A default metric was attached to the redistribution (line 4).

There are a few things to note when creating default routes in IGRP. First, IGRP does not use 0.0.0.0 as a default route. Thus, if 0.0.0.0 were defined in place of 10.0.0.0, IGRP would not convey it. Second, how should one choose which network number to flag as a default route? In the previous example, the network 10.0.0.0 does not need to be a real network number configured on an interface; it could just be a fictitious number (that does not exist as a real number in the network) to which all default traffic will be sent. Using a fictitious number instead of a real network number as the default route can have certain advantages. For example, a fictitious network number will not go down if an interface goes down. Further, changing the ideal

candidate for the default route can be much easier with fictitious network numbers than with real network numbers.

Multiple Default Routes

To increase the reliability of the connection to branches, each branch may be connected to two core routers:

```
hostname core2
!
interface Ethernet0
 ip address 192.168.1.1 255.255.255.0
...
interface Serial0
ip address 172.16.246.1 255.255.255.0
...
router igrp 10
 redistribute static
 network 172.16.0.0
 default-metric 10000 100 255 1 1500
!
no ip classless
ip default-network 10.0.0.0
ip route 10.0.0.0 255.0.0.0 Null0
```

branch1 will now receive two default routes:

```
branch1>sh ip route
...
Gateway of last resort is 172.16.250.1 to network 10.0.0.0

     172.16.0.0/24 is subnetted, 2 subnets
C       172.16.245.0 is directly connected, Serial1
C       172.16.246.0 is directly connected, Serial0
I*   10.0.0.0/8 [100/8576] via 172.16.245.1, 00:00:55, Serial0
                [100/8576] via 172.16.246.1, 00:00:55, Serial1
```

Note that it is also possible to set up one router (say, *core1*) as primary and the second router (*core2*) as backup. To do this, set up the default from *core2* with a *worse* metric, as shown in line 7:

```
hostname core2
!
interface Ethernet0
 ip address 192.168.1.1 255.255.255.0
...
interface Serial0
ip address 172.16.246.1 255.255.255.0
...
router igrp 10
 redistribute static
 network 172.16.0.0
7 default-metric 1544 2000 255 1 1500
!
```

```
no ip classless
ip default-network 10.0.0.0
ip route 10.0.0.0 255.0.0.0 Null0
```

Classful Route Lookups

Router *branch1* is configured to perform classful route lookups (see line 7 in the previous code block). A classful route lookup works as follows:

1. Upon receiving a packet, the router first determines the major network number for the destination. If the destination IP address is `172.16.1.1`, the major network number is `172.16.0.0`. If the destination IP address is `192.168.1.1`, the major network number is `192.168.1.0`.
2. Next, the router checks to see if this major network number exists in the routing table. If the major network number exists in the routing table (`172.16.0.0` does), the router checks for the destination's subnet. In our example, *branch1* would look for the subnet `172.16.1.0`. If this subnet exists in the table, the packet will be forwarded to the next hop specified in the table. If the subnet does not exist in the table, the packet will be dropped.
3. If the major network number does not exist in the routing table, the router looks for a default route. If a default route exists, the packet will be forwarded as specified by the default route. If there is no default route in the routing table, the packet will be dropped.

Router *branch1* is able to ping `192.168.1.1` as a consequence of rule 3:

```
branch1#ping 192.168.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/50/80 ms
```

However, let's define a new subnet of `172.16.0.0` on *core1* (and then block the advertisement of this subnet with an access list on lines 8 and 9) and see if *branch1* can reach it using a default route:

```
hostname core1
!
interface Ethernet0
 ip address 192.168.1.1 255.255.255.0
!
interface Ethernet1
ip address 172.16.10.1 255.255.255.0
...
interface Serial0
ip address 172.16.245.1 255.255.255.0
...
router igrp 10
```

```
 redistribute static
 network 172.16.0.0
 default-metric 10000 100 255 1 1500
 distribute-list 1 out serial0
!
no ip classless
ip default-network 10.0.0.0
ip route 10.0.0.0 255.0.0.0 Null0
!
access-list 1 deny 172.16.10.0 0.0.0.255
access-list 1 permit 0.0.0.0 255.255.255.255


branch1#sh ip route
...
Gateway of last resort is 172.16.245.1 to network 10.0.0.0

     172.16.0.0/24 is subnetted, 1 subnets
C       172.16.245.0 is directly connected, Serial0
I*   10.0.0.0/8 [100/8576] via 172.16.245.1, 00:00:26, Serial0


branch1#ping 192.168.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/50/80 ms
```

This demonstrates the use of rule 2, which causes the packet for `172.16.10.1` to be dropped. Note that in this example `172.16.10.1` did not match the default route, whereas `192.168.1.1` did match the default.

Classless route lookup, the other option, is discussed in Chapter 5.

Summing Up

IGRP has the robustness of RIP but adds a major new feature—route metrics based on bandwidth and delay. This feature—along with the ease with which it can be configured and deployed—has made IGRP tremendously popular for small* to mid-sized networks. However, IGRP does not address several problems that also affect RIP:

- The exchange of full routing updates does not scale for large networks—the overhead of generating and processing all routes in the AS can be high.

* The definition of small, medium, and large IP networks can be discussed ad nauseam because of the number of variables involved (number of routers and routes, network bandwidth/utilization, network delay/latency, etc.), but rough measures are as follows: small—a few dozen routers with up to a few hundred routes; medium—a few hundred routers with a few thousand routes; large—anything bigger than medium.

- IGRP convergence times can be too long.
- Subnet mask information is not exchanged in IGRP updates, so Variable Length Subnet Masks (VLSM) and discontiguous address spaces are not supported.

These issues may be too significant to overlook in large IP networks in which address-space conservation may necessitate VLSM, full route updates would be so large that they would consume significant network resources (serial links to branches tend to saturate quickly, and smaller routers may consume a lot of CPU power just to process all the routes at every update interval), and the convergence times may be too long because of the network diameter. Even small to mid-sized networks may choose not to implement IGRP if convergence time is an issue.