Chapter1
The IP Routing Protocols

1

This chapter describes routing protocol options for the Internet Protocol (IP) suite. The chapter "Routing IP" contains all the information you need for configuring IP. This chapter focuses on IP routing protocols. Other protocol stacks—DECnet, Novell, Apollo, and so on—are described in their own chapters. Topics in this chapter include:

- An introduction to the IP routing protocols
- Starting the routing process for a particular protocol
- Configuring static and dynamic routing
- Configuring the supported IP protocols—IGRP, RIP, HELLO, EGP, and BGP.
- Configuring multiprotocol operations, including redistribution of information from one protocol to another
- Filtering incoming and outgoing updates on the interface

Cisco-Supported Routing Protocols

Routing is the process of determining where to send data packets destined for addresses outside the local network. Routers gather and maintain routing information to enable the transmission and receipt of such data packets. Conceptually, routing information takes the form of entries in a routing table, with one entry for each identified route. The router can create and maintain the routing table dynamically to accommodate network changes whenever they occur.

Note: It is traditional when discussing IP routing protocols to refer to routers as *gateways*. For this reason, many IP routing protocols contain the word *gateway* as part of their name. Keep in mind that a gateway is a generic layer 3 and above device that connects one software stack to another. So an X.25 gateway, an electronic mail (layer 7) gateway and a router are all—in a computer science sense—gateways.

Interior and Exterior Protocols

The routing protocols are broadly divided into two classes, interior gateway protocols (IGPs), and exterior gateway protocols (EGPs). The interior routing protocols supported by Cisco include the Routing Information Protocol (RIP), HELLO, and the Interior Gateway Routing Protocol (IGRP). Interior protocols are used for routing networks that are under a common network administration. The exterior routing protocols include the Exterior Gateway Protocol (EGP) and the Border Gateway Protocol (BGP). Exterior protocols are used to exchange routing information between networks that do not share a common administration.

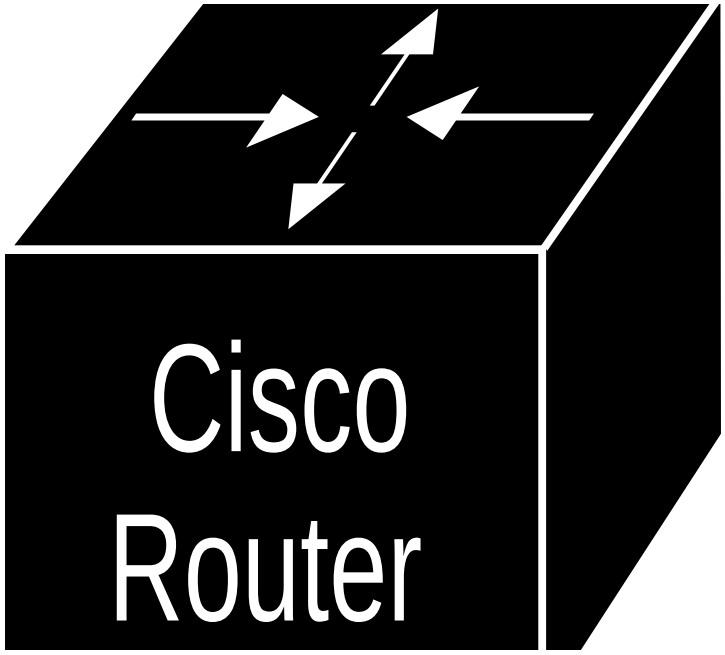
- IGRP, developed by Cisco Systems, focuses on large networks with complex topology and segments having different bandwidth and delay characteristics.
- RIP is the routing protocol used by the routed process on Berkeley-derived UNIX systems. Many networks use RIP; it works well for small, isolated, and topologically simple networks.
- HELLO is an older interior routing protocol used in the early National Science Foundation (NSF) backbone network.
- EGP is the original exterior protocol and is still used primarily in the DDN (Defense Data Network) and NSFnet (National Science Foundation Network).
- BGP is a more recent exterior routing protocol that solves some of EGP's failings.

The Cisco routers offer many protocol-independent routing features. For example, subnetting lets you divide a network into logical subparts. Load-balancing lets you split network traffic over parallel paths, which provides greater overall throughput and reliability. Because the router can avoid routing loops, you can implement general network topologies. Notification of disabled interfaces eliminates network *black holes*. Static routing table entries can provide routing information when dynamically obtained entries are not available. Most protocol-independent routing capabilities are discussed in the chapter "Routing IP." This chapter focuses on the routing protocols themselves.

Autonomous Systems

An autonomous system (AS) is a collection of networks under a common administration sharing a common routing strategy (see Figure 1-1). An autonomous system may comprise one or many networks, and each network may or may not have an internal structure (subnetting). The autonomous system number, which is assigned by the DDN Network Information Center, is a 16-bit decimal number that uniquely identifies the autonomous system. An assigned AS number is required when running EGP or BGP. All routers that belong to an autonomous system must be configured with the same autonomous system number.

Figure 1-1 Autonomous System 12 Contains Four Routers



Multiple Routing Protocols

The multiple routing protocol support in the Cisco routers was designed for connecting networks that might be using different routing protocols. It is possible, for example, to run RIP on one subnetted network, IGRP on another subnetted network, and to exchange the routing information in a controlled fashion. The routing protocols available today were not designed to interoperate with one another, so each protocol collects different types of information and reacts to topology changes in its own way. For example, RIP uses a hop count metric and IGRP uses a five-valued vector of metric information. In the case where routing information is being exchanged between different networks that use different routing protocols, there are many configuration options to enable and to filter the exchange of routing information. See the section "Redistributing Routing Information" later in this chapter.

Configuration Overview

Each routing protocol must be configured separately. Because of this need, we have separated most of the configuration information into protocol-specific subsections. The interior routing protocols will be listed first, followed by the exterior protocols. With any routing protocol, you must follow these basic steps:

Step 1: Create the routing process with one of the **router** commands.

Step 2: Configure the protocol specifics.

The next sections provide a review of the two protocol classes and how they are configured, followed by sections that explain how to configure each of the routing protocols. EXEC-level commands for monitoring the IP routing operations are also provided, and these are

explained at the end of this chapter, along with alphabetical summaries of the configuration commands.

Configuring the Interior Routing Protocols

The interior routing protocols IGRP, RIP, and HELLO must have a list of networks specified by the **network** router subcommand before routing activities can begin. The routing process will listen to updates from other routers on these networks and will broadcast its own routing information on those same networks. The IGRP routing protocol has the additional requirement of an autonomous system number, usually assigned by the DDN NIC.

Configuring the Exterior Routing Protocols

The exterior routing protocols require three sets of information before routing can begin:

- A list of neighbor (or peer) routers with which to exchange routing information. This list is created with the **neighbor** router subcommand.
- A list of networks to advertise as directly reachable, created with the **network** router subcommand.
- The AS number of the local router.

The following sections in this chapter describe the protocols, beginning with the interior routing protocols: IGRP, RIP, and HELLO.

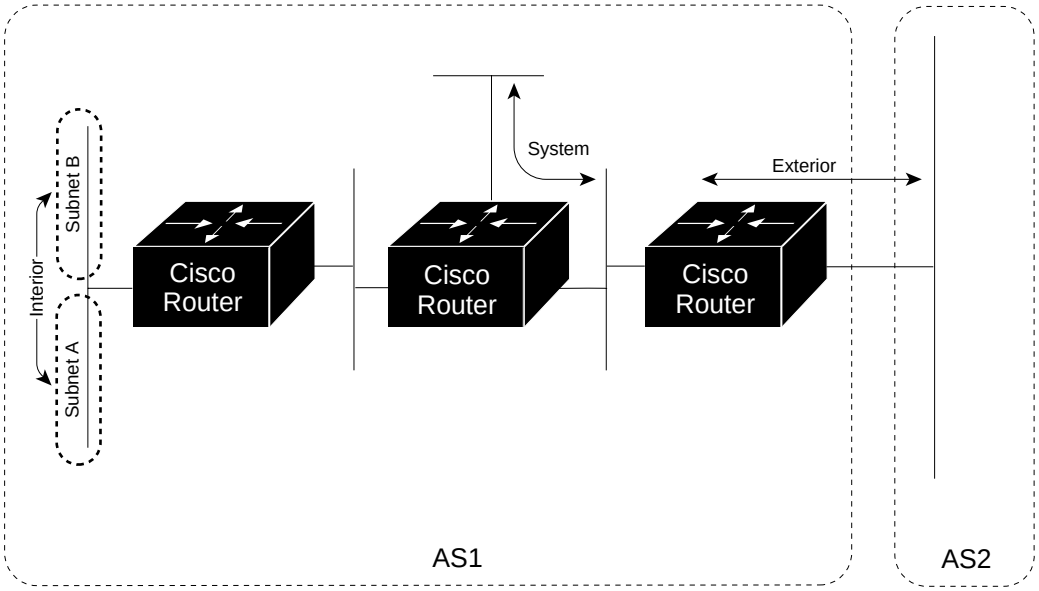
Configuring the IGRP Protocol

Cisco Systems designed the Interior Gateway Routing Protocol (IGRP) for routing in an autonomous system having arbitrarily complex topology and consisting of media with diverse bandwidth and delay characteristics. The IGRP protocol advertises all connected and IGRP-derived networks for a particular autonomous system. A single router can service up to four autonomous systems and keep the routing information for each system separate.

Interior, System, and Exterior Routes

IGRP advertises three types of routes: interior, system, and exterior as shown in Figure 1-2. Interior routes are routes between subnets in the network attached to a router interface. If the network attached to a router is not subnetted, IGRP does not advertise interior routes.

Figure 1-2 Interior, System, and Exterior Routes



System routes are routes to networks within the autonomous system. The router derives system routes from directly connected network interfaces and system route information provided by other IGRP-speaking routers. System routes do not include subnetting information. Exterior routes are routes to networks outside the autonomous system.

Creating the IGRP Routing Process

To create the routing process, use the **router** global configuration command. The full syntax of this command follows:

router igrp *autonomous-system*
no router igrp *autonomous-system*

The argument *autonomous-system* identifies the routes to other IGRP routers, and is used to tag the routing information passed along.

Use the **no router igrp** command to shut down the routing process on the AS specified by the *autonomous-system* argument.

Next, specify the list of networks with the **network** router configuration subcommand. The full syntax of this command is listed below.

network *network-number*
no network *network-number*

The argument *network-number* is a network number in dotted IP notation. Note that this number must *not* contain subnet information. You may specify multiple **network** subcommands.

When an IGRP routing process is configured, an AS number must be specified. This number may or may not be assigned by the DDN NIC. The AS number is used to tag updates belonging to one of up to four IGRP routing processes.

Use the **no network** command with the network number to remove a network from the list.

Example:

In this example, a router is configured for IGRP and assigned to AS 109. In the next two lines, two network commands assign the two networks to a list of networks to receive IGRP updates, as shown.

```
router igrp 109
network 131.108.0.0
network 192.31.7.0
```

Choosing the Gateway of Last Resort

The router chooses a *gateway of last resort* from the list of exterior routers that IGRP provides. The router uses the gateway (router) of last resort, if it does not have a better route for a packet. If the autonomous system has more than one connection to an external network, different routers may choose different exterior routers as the gateway of last resort.

IGRP Metric Information

IGRP uses several types of metric information. For each path through an autonomous system, IGRP records the segment with the lowest bandwidth, the accumulated delay, the smallest MTU (Maximum Transmission Unit), and the reliability and load.

The IGRP metric is a 32-bit quantity that is a sum of the segment delays and the lowest segment bandwidth (scaled and inverted) for a given route. For a network of homogeneous media, this metric reduces to a hop count. For a network of mixed media (FDDI, Ethernets, and serial lines running from 9,600 baud to T1 rates), the route with the lowest metric reflects the most desirable path to a destination.

IGRP Updates

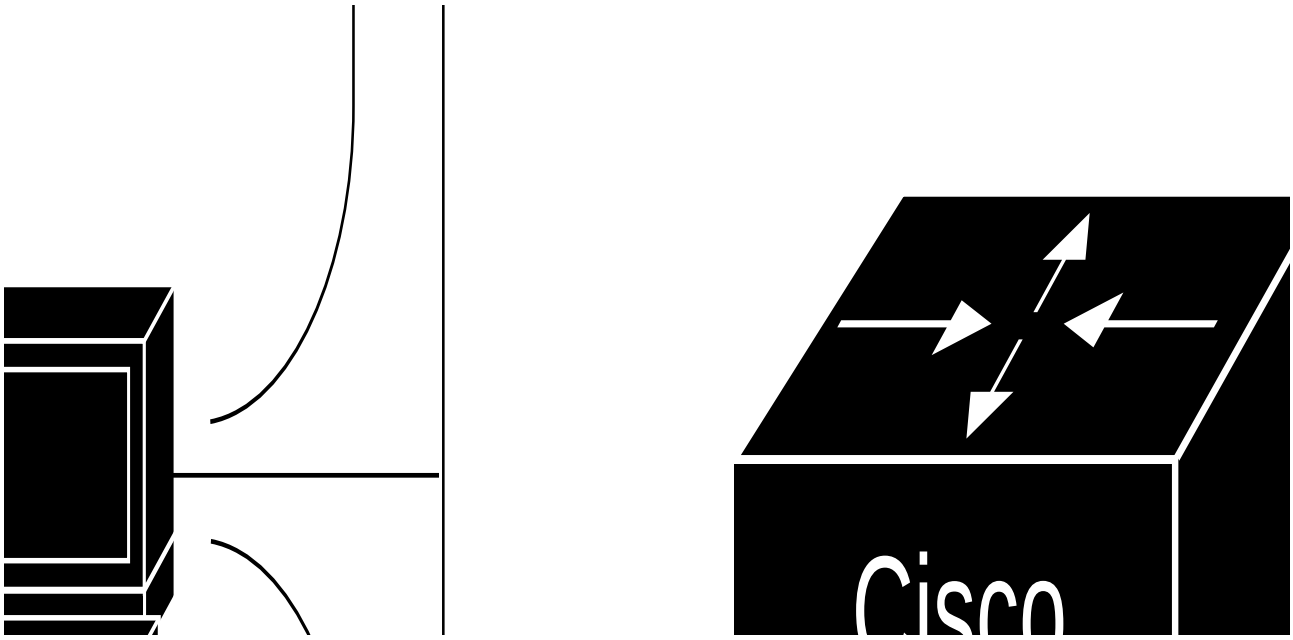
A router running IGRP sends an IGRP update broadcast every 90 seconds. It declares a route inaccessible if it does not receive an update from the first router in the route within three update periods (270 seconds). After five update periods (450 seconds), the router removes the route from the routing table. IGRP uses flash update and poison reverse to speed up the convergence of the routing algorithm.

Configuring the RIP Protocol

The Routing Information Protocol (RIP) uses broadcast User Datagram Protocol (UDP) data packets to exchange routing information. Each router sends routing information updates every 30 seconds; this process is termed *advertising*. If a router does not receive an update from another router for 180 seconds or more, it marks the routes served by the nonupdating router as being unusable. If there is still no update after 240 seconds, the router removes all routing table entries for the nonupdating router.

The measure, or metric, that RIP uses to rate the value of different routes is the *hop count*. The hop count is the number of routers that may be traversed in a route. A directly connected network has a metric of zero (see Figure 1-3); an unreachable network has a metric of 16. This small range of metrics makes RIP unsuitable as a routing protocol for wide area networks. If the router has a default network path, RIP advertises a route that links the router to the pseudo-network *0.0.0.0*. The network *0.0.0.0* does not exist; RIP treats *0.0.0.0* as a network to implement the default routing feature.

Figure 1-3 Hop Count in RIP



Creating the RIP Routing Process

To create a routing process for RIP, use the **router rip** global configuration command:

router rip
no router rip

Use the **no router rip** command to shut down the routing process.

Specifying the List of Networks

Next, specify the list of networks with the **network** router configuration subcommand. The full syntax of this command follows.

network *network-number*
no network *network-number*

The argument *network-number* is a network number in dotted IP notation. Note that this number must *not* contain subnet information. You may specify multiple **network** subcommands. RIP routing updates will be sent and received only through interfaces on this network.

Example:

The following example configuration defines RIP as the routing protocol to be used on all interfaces connected to networks *128.88.0.0* and *192.31.7.0*.

```
router rip
network 128.99.0.0
network 192.31.7.0
```

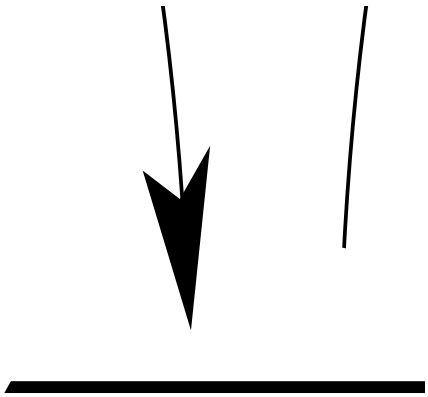
To remove a network from the list, use the **no network** router subcommand followed by the network address.

Configuring the HELLO Protocol

The HELLO protocol, described in RFC 891, was developed for the Fuzzball gateways of the Distributed Computer Network project and was used extensively in the early NSFnet backbone network. HELLO is an interior routing protocol.

The Cisco Systems implementation of HELLO does not implement the extensive timekeeping and delay measurement features. Specifically, the router sets the invalid bit in the HELLO date field and clears the time and timestamp fields.

Figure 1-4 The HELLO Protocol



The metric used in HELLO is a delay value measured in milliseconds (see Figure 1-2). This metric can range from 0 to 30,000 milliseconds, making HELLO a good candidate for routing larger networks. A network with a 30,000-millisecond delay is considered unreachable. The Cisco Systems implementation uses a delay of 100 milliseconds for all routes, regardless of their actual delay characteristics.

Creating the HELLO Routing Process

The first step is to create the routing process with the **router** global configuration command:

router hello
no router hello

Use the **no router hello** command to shut down the routing process.

Specifying the List of Networks

The next step is to specify the list of networks. This list is specified with the **network** router configuration subcommand:

network *network-number*
no network *network-number*

The argument *network-number* is a network number in dotted IP notation. Note that this number must not contain subnet information. You can specify multiple **network** subcommands.

Example:

In the following example, the network *160.1.1.0* is being set up for HELLO.

```
router hello
network 160.1.1.0
```

Configuring the BGP Protocol

BGP, as defined in RFC 1163 and RFC 1164, allows you to set up a distributed routing core that automatically guarantees the loop-free exchange of routing information on an autonomous system (AS) basis.

Creating the BGP Routing Process

To configure BGP, use the **router bgp** global configuration command:

router bgp *autonomous-system*
no router bgp *autonomous-system*

The *autonomous-system* number is used to identify the router to other BGP routers, and to tag the routing information passed along.

Example:

In the following example, a router is assigned to AS 120.

```
router bgp 120
```

Specifying the List of Networks

Use the **network** router subcommand to specify those networks that are to be advertised as originating within the current AS. These networks can be learned from connected, dynamic routing, and static route sources. The command syntax follows:

network *network-number*

The argument *network-number* is the dotted IP address of the network that will be included in the router's BGP updates.

Example:

In the following command, the network *131.108.0.0* is set up to be included in the routers BGP updates.

```
network 131.108.0.0
```

Specifying the List of Neighbors

BGP supports two different kinds of neighbors: internal and external. Internal neighbors are in the same autonomous system. External neighbors are in other autonomous systems.

BGP routing includes several related router subcommands that specify "neighbor" routers and manage your router's relationship with its neighbors. Use the **neighbor** router subcommands to:

- Identify BGP peers and their AS numbers.
- Assign access lists.
- Set up various routing policies.

Basic Neighbor Specification

In the simplest case, you simply want to specify that another router is a neighbor. Use the

simple neighbor command, as shown below:

neighbor *address* **remote-as** *autonomous-system*
no neighbor *address* **remote-as** *autonomous-system*

Using the **no** keyword removes the router as a neighbor.The arguments *address* and *autonomous-system* are the neighbor's address and AS number.

Example 1:

This example specifies that the router at the address *131.108.0.0* is a neighbor.

```
neighbor 131.108.1.2 remote-as 109
```

Example 2:

In the following example, a BGP router is assigned to AS 109, and two networks are listed as originating in the AS. Then the addresses of two remote routers (and their ASes) are listed. The router being configured will share information about networks *131.108.0.0* and *192.31.7.0* with the two neighbor routers, as shown:

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  remote-as 167
neighbor 131.108.240.67  remote-as  99
```

Routing Weights

The **neighbor weight** router subcommand specifies a weight to assign to a specific neighbor connection. Its full syntax is as follows:

neighbor *address* **weight** *weight*
no neighbor *address* **weight** *weight*

The argument *address* is the address of the neighbor connection. The argument *weight* is the weight value to assign. The route with the highest weight will be chosen as the preferred route when multiple routes are available to a particular network.

Use the **no neighbor** command with the appropriate arguments and keywords to remove this function.

Example:

In the example below, the neighbor at address *151.23.12.1* is assigned a weight of 50.

```
neighbor 151.23.12.1 weight 50
```

Filtering BGP Advertisements

You can filter BGP advertisements in two ways: by using access lists and by using AS-path filters. Access lists in IP are discussed in the chapter "Routing IP."

You can apply access lists to BGP updates with the **neighbor distribute-list** router subcommand. Its full syntax follows.

neighbor *address* **distribute-list** *list* {**in**|**out**}
no neighbor *address* **distribute-list** *list*

The argument *address* is the address of the neighbor connection. The argument *list* is a predefined access list number. The keywords **in** and **out** specify whether you are applying the access list to incoming or outgoing advertisements to that neighbor.

Use the **no neighbor** command with the appropriate arguments and keywords to remove this function.

Example:

In the example below, list 41 is applied to outgoing advertisements to neighbor 120.23.4.1.

```
neighbor 120.23.4.1 distribute-list 41 out
```

You can also filter neighbor updates coming from specific neighbors by AS number using these variations of the **neighbor** command.

neighbor *address* **filter-as** *number* **deny**
no neighbor *address* **filter-as** *number* **deny**

neighbor *address* **filter-as** *number* **permit** *weight*
no neighbor *address* **filter-as** *number* **permit** *weight*

The **permit** keyword assigns a weight to routes that include the specified AS in the path. The **deny** keyword causes the information learned about that network using the specified AS to be ignored.

Example:

This example assigns the weight 60 to all routes learned from the neighbor at address *120.23.4.1* which traverse AS 20.

```
neighbor 120.23.4.1 filter-as 20 permit 60
```

Adjusting the BGP Timers

To adjust the default BGP timers, use the **timers bg**p router subcommand. The full syntax of this command follows.

timers bgp *keepalive holdtime*
no timers bgp

The argument *keepalive* is the frequency in seconds with which the router sends *keepalive* messages to its peer (default 60 seconds), and *holdtime* is the interval in seconds after not receiving a *keepalive* message that the router declares a peer dead (default 180 seconds). The **no timers bgp** command restores the default.

Example:

In the example below, the keepalive timer is changed to 70 seconds and the holdtime is changed to 210 seconds.

```
timers bgp 70 210
```

BGP and IGP Routing Information

This section discusses the issues of BGP interacting with the various interior gateway protocols (referred to generically as IGPs), such as IGRP, RIP, and HELLO. BGP maintains its own routing table, separate from the main IP routing table used to make datagram switching decisions. The BGP routing table is organized by network, and contains information referred to as *attributes*, such as the list of ASs that a datagram must transit to reach a particular network. Information from the BGP routing table is periodically entered into the main IP routing table and is aged appropriately (see Figure 1-5). In most cases, the BGP information should override IGP information.

Figure 1-5 BGP and IGP Routing

Networks that originate in the local AS are indicated with the **network** router subcommand for the BGP process. Such networks, referred to as local networks, will have a BGP origin attribute of IGP. They appear in the main IP routing table and may have any source, for example, directly connected, static route, learned from an IGP, and so forth. The BGP routing process periodically scans the main IP routing table to detect the presence or absence of local networks, updating the BGP routing table as appropriate.

Since all networks originating within an AS are flagged by the **network** command, information from an IGP about a nonlocal network that conflicts with external BGP information will be suppressed on the grounds that BGP is supplying better information. It is possible, however, to indicate which networks are reachable using a back-door route that the border router should use instead. Back-door networks are permitted in the main IP routing table,

but are not entered into the BGP routing table, except when they are learned from another BGP-speaking router

Use this variation of the **network** router subcommand to specify a back-door route:

network *address* **backdoor**

The argument *address* is the network that you wish a backdoor route to.

Example:

In the example below, network *131.108.0.0* is a local network and network *192.31.7.0* is a backdoor network.

```
router bgp 109
network 131.108.0.0
network 192.31.7.0  backdoor
```

Using the **redistribute** router subcommand, you can inject BGP routing information into the IGP. This creates a situation where BGP is potentially deriving information about local networks from the IGP, and then sending such information back into the IGP. This is another reason to suppress nonlocal networks from the IP routing table—you do not want to hear echoes of your routing updates.

It is also possible to inject IP routing table information into the BGP routing table using the **redistribute** router subcommand. EGP-derived information will have a BGP origin attribute of EGP; all other nonlocal routes will have a BGP origin attribute of incomplete. All IGP and EGP information will now override BGP-derived IP routing table entries. If you are also redistributing information from BGP into an IGP, you must set up appropriate filtering using the **distribute-list** command to ensure that routing information doesn't loop. A configuration such as this is fairly risky, requiring careful attention to filtering. Filtering is described in more detail later in this chapter.

BGP Route Selection Rules

The BGP process selects a single AS path to pass along to other BGP-speaking routers. It is important for routing stability that each BGP-speaking router in an AS use the same set of rules so that all BGP-speaking routers arrive at a consistent view of the AS topology. To this end, the Cisco BGP implementation has a reasonable set of factory defaults that may be overridden by administrative configuration, as follows:

- An AS path for a network sourced by this BGP-speaking router has the highest preference. The Cisco router uses the sourced path with the lowest origin code.
- Administrative weighting is then considered. Larger weight takes precedence.
- Prefer the shorter AS path. All succeeding rules assume equal length paths.
- Prefer external links over internal links.
- Prefer the lowest origin code (IGP <EGP <INCOMPLETE).
- If INTER_AS metric attributes are present, prefer path with lowest metric.
- If IGP synchronization is enabled, then a path learned via BGP must be synchronized with IGP before it can be considered for selection.
- Final determinant is the peer with the largest value for the IP address.

BGP Path Attributes

The Cisco BGP implementation supports all path attributes defined in RFC 1163. This section describes some details of that implementation.

The **default-metric** router subcommand may be used to configure the value for the INTER_AS metric attribute. The same metric value will be sent with all BGP updates originating from the router. The default is to not include an INTER_AS metric in BGP updates.

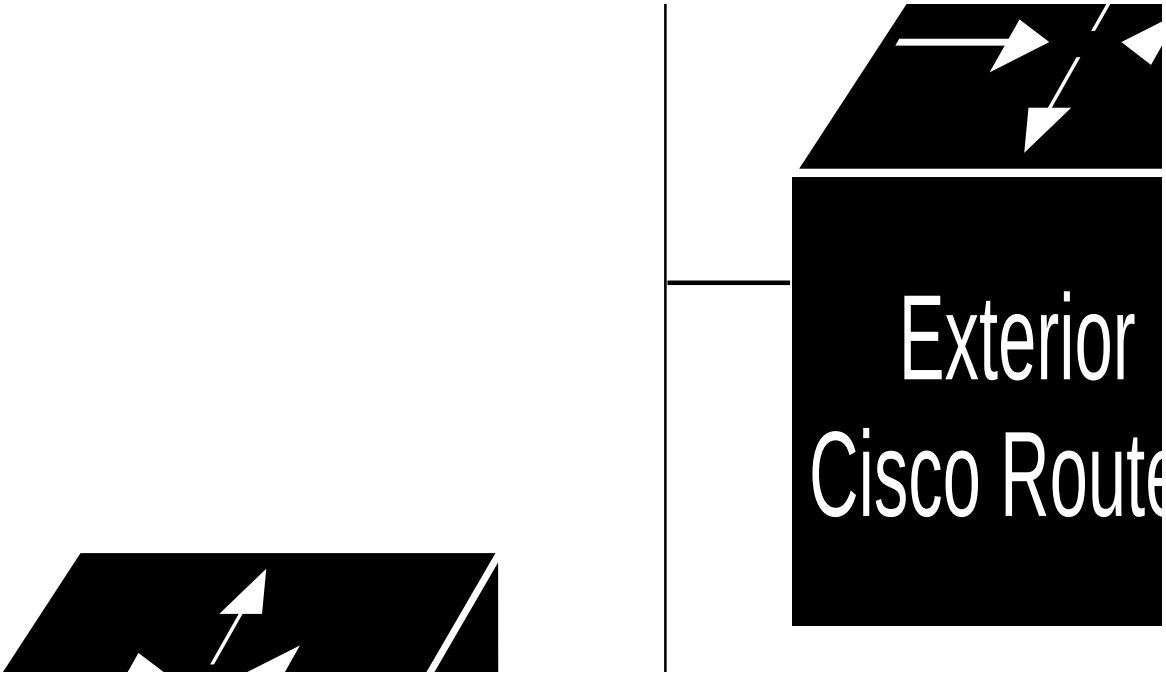
The Cisco implementation will use a third party next hop router address in NEXT_HOP attribute regardless of the AS of that third party router.

Transitive, optional path attributes are passed along to other BGP-speaking routers. The Cisco BGP implementation does not currently generate such attributes.

Configuring the EGP Protocol

The Exterior Gateway Protocol (EGP), specified in RFC 904, is used for communicating with certain routers in the Defense Data Network (DDN) that the U.S. Department of Defense designates as core routers. EGP is also extensively used when attaching to the NSFnet (National Science Foundation Network) and other large backbone networks as shown in Figure 1-6. An exterior router uses EGP to advertise its knowledge of routes to networks within its autonomous system. It sends these advertisements to the core routers, which then re-advertise their collected routing information to the exterior router. A neighbor or peer router is any router with which the router communicates using EGP.

Figure 1-6 EGP and Interior and Exterior Routers



Specifying the Autonomous System Number

Before you can set up EGP routing, you must specify an autonomous system number using the **autonomous-system** global configuration command. The syntax for this command follows:

autonomous-system *local-AS*
no autonomous-system *local-AS*

The argument *local-AS* is the local autonomous system (AS) number to which the router belongs. The local AS number will be included in EGP messages sent by the router. To remove the AS number, use the **no autonomous-system** global configuration command.

Creating the EGP Routing Process

After the local AS number has been specified, start the EGP routing process with a **router egp** global configuration command:

router egp *remote-AS*
no router egp *remote-AS*

The argument *remote-AS* is the AS number the router expects its peers to be advertising in their EGP messages. The Cisco software does not insist that the actual remote AS number match the configured remote AS numbers. (The output from **debug ip-egp** EXEC command will advise of any discrepancies, however. See the section "Debugging IP Routing" for more information.) You can create up to four different EGP routing processes. Turn off your EGP routing process with the **no router egp** subcommand.

Specifying the List of Neighbors

A router using EGP cannot dynamically determine its neighbor or peer routers. You must provide a list of neighbor routers using the **neighbor** router subcommand:

neighbor *ip-address*
no neighbor *ip-address*

The argument *ip-address* is the IP address of a peer router with which routing information will be exchanged. Multiple **neighbor** subcommands may be used to specify additional neighbors or peers. The **no neighbor** subcommand followed by an IP address removes a peer from the list.

Specifying the Network to Advertise

Use the **network** router subcommand to specify the network to be advertised to the EGP peers of an EGP routing process.

network *network-number*
no network *network-number*

The argument *network-number* is the IP address of the network. Such networks are advertised

with a distance of zero. There is no restriction on the network number other than that the network must appear in the routing table. The network may be connected, may be statically configured, or may be redistributed into EGP from other routing protocols.

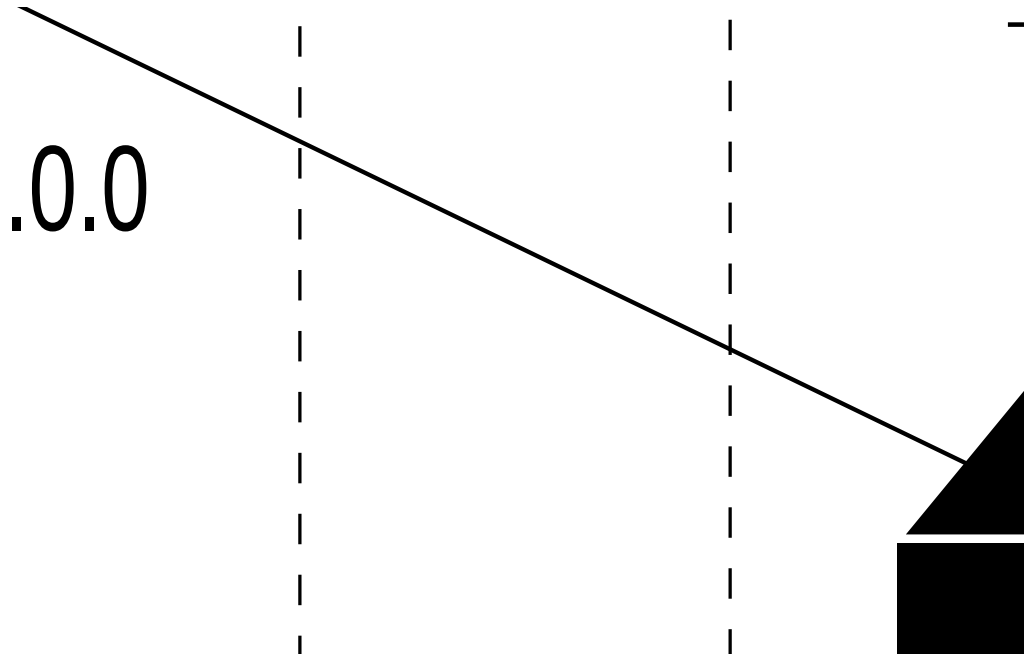
Multiple **network** subcommands may be used to specify additional networks. The **no network** subcommand followed by the network number removes a network from the list.

Example:

The following is an example configuration for an EGP router process. The router is in autonomous system 109 and is peering with routers in AS 164, as shown in Figure 1-7. It will advertise the networks *131.108.0.0* and *192.31.7.0* to the router in AS 164, *10.2.0.2*. The information sent and received from peer routers may be filtered in various ways, including blocking information from certain routers and suppressing the advertisement of specific routes.

```
autonomous-system 109
router egp 164
network 131.108.0.0
network 192.31.7.0
neighbor 10.2.0.2
```

Figure 1-7 Router in AS 164 Peers with Router in AS 109



Adjusting Timers

The HELLO and polltime timers for EGP are adjustable. To adjust the EGP timers, use the subcommand:

timers egp *hello polltime*
no timers egp

The argument *hello* is the frequency in seconds with which the router sends HELLO messages to its peer. The default is 60 seconds.

The argument *polltime* is the interval in seconds after not receiving a *hello* message that the router declares a peer dead. The default is 180 seconds, and the **no timers egp** restores this default.

Example:

This command changes the EGP timers to two minutes and five minutes respectively.

```
timers egp 120 300
```

To change the invalid time or flush time for EGP routes, use the **timers basic** command as explained in the section "Special Routing Configuration Techniques" later in this chapter.

Configuring Third-Party EGP Support

EGP supports what is termed a *third-party mechanism*. In this circumstance EGP tells its peer that another router (the third party) on the shared network is the appropriate router for some set of destinations. If updates mentioning third party routers are desired, they may be configured using a variation of the **neighbor** subcommand (full syntax follows):

neighbor *address* **third-party** *third-party-ip-address* [**internal**|**external**]
no neighbor *address* **third-party** *third-party-ip-address* [**internal**|**external**]

The argument *third-party-ip-address* is the address of another router (the third party) on the network shared by the Cisco router and the EGP peer specified by the *address* argument. All networks reachable through that third party router will be listed in the Cisco EGP updates as reachable via that router. Any other networks will be listed as reachable via the Cisco router. The optional keyword **internal** or **external** indicates whether the third party router should be listed in the internal or external section of the EGP update. Normally, all networks are mentioned in the internal section. You may use the **neighbor** *address* **third-party** router subcommand multiple times to specify additional third party routers.

Example 1:

In the following example, routes learned from router *131.108.6.99* will be advertised to *131.108.6.5* as third-party internal routes.

```
neighbor 131.108.6.5  third-party 131.108.6.99 internal
```

Example 2:

In the following example, routes learned from *131.108.6.100* will be advertised to *131.108.6.5* as third-party external routes.

```
neighbor 131.108.6.5  third-party 131.108.6.100 external
```

Configuring a Backup EGP Router

It may be desirable to have a second router belonging to a different AS act as a backup to the EGP router for your AS. To differentiate between the primary and secondary EGP routers, the two routers will advertise network routes with differing EGP distances or metrics. A network with a low metric is generally favored over a network with a high metric.

Networks flagged with the **network** router subcommand are always announced with a metric of zero. Networks that are redistributed will be announced with a metric specified by the **default-metric** router subcommand. If no metric is specified, redistributed routes will be advertised with a metric of three. All redistributed networks will be advertised with the same metric. The redistributed networks may be learned from static or dynamic routes. See the section "Redistributing Routing Information" for details about the **redistribute** router subcommand. A complete configuration example is contained in the section "IP Routing Protocols Configuration Examples."

Example:

The following example configuration illustrates that networks learned by RIP are being advertised with a distance of five. (This is *not* a complete configuration.)

```
redistribute rip
default-metric 5
```

Filtering Routing Information

The information sent and received on the various networks may be filtered in various ways, including blocking information from certain routers, not sending updates onto a particular subnet, and suppressing the advertisement of specific routes. This section reviews the options for filtering incoming and outgoing information.

Filtering Outgoing Information

This section describes the options you may use to control outgoing information.

Suppressing Updates on an Interface.

The **passive-interface** router subcommand disables sending routing updates on an interface.

passive-interface *interface*
no passive-interface *interface*

The argument *interface* specifies a particular interface. The particular subnet will continue to be advertised to other interfaces. Updates from other routers on that interface continue to be received and processed.

The **no passive-interface** command re-enables sending routing updates on the specified interface.

Example:

In the following example, IGRP updates are sent to all interfaces on network *131.108.0.0* except interface Ethernet 1. Figure 1-8 shows this configuration.

```
router igrp 109
network 131.108.0.0
passive-interface ethernet 1
```

Figure 1-8 Filtering IGRP Updates

Filtering Outband Updates

To suppress networks from being sent in updates, use the **distribute-list** router subcommand. Full syntax for this command follows.

distribute-list *access-list-number* **out** [*interface-name* | *routing-process*]
no distribute-list *access-list-number* **out** [*interface-name* | *routing-process*]

The argument *access-list-number* is a standard IP access list number as described in the section "Configuring IP Access Lists" in the the chapter "Routing IP." The list explicitly specifies which networks are to be sent and which are to be suppressed.

Use the keyword **out** to apply the access list to outgoing routing updates.

When redistributing networks, a routing process name may be specified as an optional trailing argument to the **distribute-list** subcommand. This causes the access list to be applied to only those routes derived from the specified routing process. After the process-specific access list is applied, any access list specified by a **distribute-list** subcommand without a process name argument will then be applied.

Use the **no distribute-list** command with the appropriate access list number and keyword to disable or change this function.

Note: To filter networks received in updates, use the distribute-list command with the **in**

keyword, as explained in the section "Filtering Received Updates."

Example 1:

The following example set of configuration subcommands would cause only two networks to be advertised by a RIP routing process, network *0.0.0.0* (the RIP default) and network *131.108.0.0*.

```
access-list 1 permit 0.0.0.0
access-list 1 permit 131.108.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
router rip
network 131.108.0.0
distribute-list 1 out
```

Figure 1-9 Filtering RIP Updates

Network 13

Example 2:

To filter a routing update sent on a specific interface, you may, optionally, specify the interface. If the last line of the previous example were written as follows, the access list would be applied to updates sent on Ethernet 3.

```
distribute-list 1 out ethernet 3
```

Example 3:

In the following example, access list 3 is applied to networks derived from process IGRP 109 that are being redistributed by process EGP 164. Networks suppressed by that access list will not be advertised by EGP 164.

```
router egp 164
network 131.108.0.0
redistribute igrp 109
distribute-list 3 out igrp 109
```

Point-to-Point Updates

The **neighbor** router subcommand defines a neighboring router with which to exchange routing information, as discussed under the specific protocols:

neighbor *address*

The argument *address* is the neighboring router address.

For exterior routing protocols such as EGP and BGP, this command specifies routing peers. For normally broadcast protocols such as IGRP or RIP, this subcommand permits the point-to-point (nonbroadcast) exchange of routing information. When used in combination with the **passive-interface** subcommand, routing information may be exchanged between a subset of routers on a LAN.

Adjusting Metrics

The **offset-list** router subcommand can be used to add a positive offset to incoming and outgoing metrics for networks matching an access list. Full syntax for this command follows.

offset-list {**in**|**out**} *offset* [*accesslist*]
no offset-list {**in**|**out**} *offset* [*accesslist*]

If the argument *list* is zero, the argument supplied to *offset* is applied to all metrics. If *offset* is zero, no action is taken. For IGRP, the offset is added to the delay component only. This subcommand is implemented for the RIP and HELLO routing protocols as well.

The **no offset-list** command with the appropriate keyword removes the offset list.

Example:

In the following example, a router using IGRP applies an offset of 10 to its delay component for all outgoing metrics.

```
offset-list out 10 0
```

In the next example, the router applies the same offset only to access list 121:

```
offset-list out 10 121
```

Filtering Incoming Information

This section describes the options you may use to control incoming information.

Filtering Received Updates

Use the router subcommand **distribute-list** to filter networks received in updates.

distribute-list *access-list-number* **in** [*interface-name*]
no distribute-list *access-list-number* **in** [*interface-name*]

The argument *access-list-number* is a standard IP access list number as described in the section "Configuring IP Access Lists" in the the chapter "Routing IP." The list explicitly specifies

which networks are to be received and which are to be suppressed.

Use the keyword **in** to suppress incoming routing updates.

The optional argument *interface-name* specifies the interface (for example, Ethernet 0) on which the access list should be applied to incoming updates. If no interface is specified, the access list will be applied to all incoming updates.

Use the **no distribute-list** command with the appropriate access list number and keyword to disable or change this function.

Example:

The following set of example configuration subcommands would cause only two networks to be accepted by a RIP routing process, network *0.0.0.0* (the RIP default) and network *131.108.0.0*.

```
access-list 1 permit 0.0.0.0
access-list 1 permit 131.108.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
router rip
network 131.108.0.0
distribute-list 1 in
```

Filtering Sources of Routing Information

In a large network, some routing protocols and some routers can be more reliable than others as sources of routing information. By specifying administrative distance values, you enable the router to intelligently discriminate between sources of routing information.

An administrative distance is a rating of the trustworthiness of a routing information source, such as an individual router or a group of routers. Numerically, an administrative distance is an integer between 0 and 255. In general, the higher the value, the lower the trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored.

The router always uses the best routing source available: the routing source with the lowest administrative distance. For example, consider a router using IGRP and RIP. Suppose you

trust the IGRP-derived routing information more than the RIP-derived routing information. If you set the administrative distances accordingly, the router uses the IGRP-derived information and ignores the RIP-derived information. However, if you lose the source of the IGRP-derived information (say, to a power shutdown in another building), the router uses the RIP-derived information until the IGRP-derived information reappears.

You can also use administrative distance to rate the routing information from routers running the same routing protocol. This application is generally discouraged, however, since it can result in inconsistent routing information including forwarding loops. Example 1, below, shows how to do this safely.

To define an administrative distance, use the **distance** router subcommand.

distance *weight* [[i*p-source-address ip-address-mask*] [*access-list-number*]]

no distance *weight* [[i*p-source-address ip-address-mask*] [*access-list-number*]]

The argument *weight* is an integer from 10 to 255 that specifies the administrative distance. (Values 0 through 9 are reserved for internal use.) Used alone, the argument *weight* specifies a default administrative distance that the router uses when no other specification exists for a routing information source. Weight values are subjective; there is no quantitative method for choosing weight values.

The optional argument pair *ip-source-address* and *ip-address-mask* specifies a particular router or group of routers to which the weight value applies. The argument *ip-source-address* is an Internet address that specifies a router, network, or subnet. The argument *ip-address-mask* (in dotted-decimal format) specifies which bits, if any, to ignore in the address value; a set bit in the *mask* argument instructs the router to ignore the corresponding bit in the address value. The optional argument *access-list-number* is the number of a standard IP access list. When used, it will apply the access list number when a network is being inserted into the routing table. This allows filtering of networks according to the IP address of the router supplying the routing information. This could be used, as an example, to filter out possibly incorrect routing information from routers not under your administrative control.

To remove an administrative distance value, use the **no distance** subcommand with the appropriate arguments and keywords.

Example 1:

In the example below,the **router igrp** global configuration command sets up IGRP routing in AS number 109. The network subcommands specify routing on networks *192.31.7.0* and *128.88.0.0*. The first **distance** router subcommand sets the default administrative distance to 255, which instructs the router to ignore all routing updates from routers for which an explicit distance has not been set. The second **distance** subcommand sets the administrative distance for all routers on the Class C network *192.31.7.0* to 90. The third **distance** subcommand sets the administrative distance for the router with the address *128.88.1.3* to 120.

```
router igrp 109
network 192.31.7.0
network 128.88.0.0
distance 255
distance 90  192.31.7.0  0.0.0.255
distance 120  128.88.1.3  0.0.0.0
```

Example 2:

The order in which you enter **distance** router subcommands can affect the assigned administrative distances in unexpected ways. For example, the following subcommands assign the router with the address *192.31.7.18* an administrative distance of 100, and all other routers on subnet *192.31.7.0* an administrative distance of 200.

```
distance 100 192.31.7.18  0.0.0.0
distance 200 192.31.7.0  0.0.0.255
```

Example 3:

However, if you reverse the order of these subcommands, all routers on subnet *192.31.7.0* are assigned an administrative distance of 200, even the router at address *192.31.7.18*.

```
distance 200 192.31.7.0  0.0.0.255
distance 100 192.31.7.18  0.0.0.0
```

Assigning administrative distances is a problem unique to each network and is done in response to the greatest perceived threats to the connected network. Even when general guidelines exist, the network manager must ultimately determine a reasonable matrix of administrative distances for the network as a whole. Table 1-1 below shows the default administrative distance for various sources of routing information.

Table 1-1 Default Administrative Distances

Route Source	Default Distance
connected interface	0
static route	1
IGRP	100
RIP	120
HELLO	130
EGP	140
BGP	200
unknown	255

Directly Connected Routes

Directly connected routes are routes to the networks specified by the interface addresses of the router. An interface may have multiple IP addresses.

Treatment of Directly Connected Routes

The router automatically enters a directly connected route in the routing table if the interface is usable. A "usable" interface is one through which the router can send and receive packets. If the router determines that an interface is not usable, it removes the directly connected routing entry from the routing table. Removing the entry allows the router to use dynamic routing protocols to determine backup routes to the network (if any).

To display the usability status of interfaces, use the EXEC command **show interfaces.** If the interface hardware is usable, the interface is marked "up." If the interface can provide two-way communication, the line protocol is marked "up." For an interface to be usable, both the interface hardware and line protocol must be up.

Multiple Interface Addresses

The software supports multiple IP addresses per interface. In addition to the primary address specified by the **ip address** interface subcommand, an unlimited number of secondary

addresses may be specified by adding the optional keyword **secondary**, as shown:

ip address *address mask* [**secondary**]

Example:

In the example below, *131.108.1.27* is the primary address and *192.31.7.17* is a secondary address for Ethernet 0.

```
interface ethernet 0
ip address 131.108.1.27 255.255.255.0
ip address 192.31.7.17 255.255.255.0 secondary
```

Secondary addresses are treated like primary addresses, except that the system never generates datagrams other than routing updates with secondary source addresses. IP broadcasts and ARP requests are handled properly, as are interface routes in the IP routing table.

Secondary IP addresses can be used in a variety of situations. The following are the most common applications:

- There may not be enough host addresses for a particular network segment. For example, your subnetting allows up to 254 hosts per logical subnet, but on one physical subnet you need to have 300 host addresses. Using secondary IP addresses on the routers allows you to have two logical subnets using one physical subnet.
- Many older networks were built using Level 2 bridges. The judicious use of secondary addresses can aid in the transition to a subnetted, router-based network. Routers on an older, bridged segment can be easily made aware that there are many subnets on that segment.
- Two subnets of a single network might otherwise be separated by another network. This situation not permitted when subnets are in use. In these instances, the first network is *extended*, or layered on top of the second network using secondary addresses.

Note: If any router on a network segment uses a secondary address, all other routers on that same segment must also use a secondary address from the same network or subnet. An inconsistent use of secondary addresses on a network segment can very quickly lead to routing loops.

Overriding Static Routes with Dynamic Protocols

A static routing entry remains in effect until you remove it. This section describes how a static route can be overridden by dynamic routing information from one of the IP routing protocols. The **ip route** global configuration command for static routes is described in the chapter "Routing IP." The full syntax of the command follows:

ip route *network router* [*distance*]

The argument *network* is the Internet address of the target network or subnet, and the argument *router* is the Internet address of a router that can reach that network. The *distance*

argument specifies an administrative distance.

If you specify an administrative distance, you are flagging a static route that may be overridden by dynamic information. For example, IGRP-derived routes have a default administrative distance of 100. To have a static route that would be overridden by an IGRP dynamic route, specify an administrative distance greater than 100.

Example:

In the example below, an administrative distance of 110 was chosen.

```
ip route 10.0.0.0  131.108.3.4   110
```

This implies that packets for network *10.0.0.0* will be routed to the router at *131.108.3.4*. if dynamic information about network *10.0.0.0* is not available.

Default Routes

A router may not be able to determine the routes to all other networks. To provide complete routing capability the common practice is to use some routers as "smart routers" and give the remaining routers default routes to the smart router. These default routes may be passed along dynamically or may be configured into the individual routers.

Generating Default Routes

Most dynamic interior routing protocols include a mechanism for causing a "smart router" to generate dynamic default information that is then passed along to other routers.

On the Cisco router, use this global configuration command:

ip default-network *network-number*

The argument *network-number* is a network number.

If the router has a directly connected interface onto that network, the dynamic routing protocols running on that router will generate or source a default route. In the case of RIP and HELLO, this is the mention of the pseudo-network *0.0.0.0*. In the case of IGRP, it is the network itself, flagged as an exterior route.

A router that is generating the default for a network may also need a default of its own. This may be done by specifying a static route to the network *0.0.0.0* via the appropriate router.

Picking a Default Route

When default information is being passed along through the dynamic routing protocol, no further configuration is required. The system will periodically scan its routing table to choose the optimal default network as its default route. In the case of RIP and HELLO, there will be only one choice, network *0.0.0.0*. In the case of IGRP, there may be several networks that can be candidates for the system default. The router uses both administrative distance and metric information to determine the default route. The selected default route

appears in the gateway of last resort display of the EXEC command **show ip route**.

If dynamic default information is not being passed to the router, candidates for the default route may be specified with the **ip default-network** subcommand. In this usage, **ip default-network** takes a nonconnected network as an argument. If this network appears in the routing table from any source (dynamic or static), then it is flagged as a candidate default route and is subject to being chosen as the default route for the router. Multiple **ip default-network** commands may be given. All candidate default routes, both static (that is, flagged by **ip default-network**) and dynamic, appear in the routing table preceded by an asterisk.

Example:

In the following example, a static route to network *10.0.0.0* is defined as the static default route.

```
ip route 10.0.0.0  131.108.3.4
ip default-network 10.0.0.0
```

If the following global configuration command was issued on a router not connected to network *129.140.0.0*, then the router might choose the path to that network as a default route when the network appeared in the routing table.

```
ip default-network 129.140.0.0
```

Subnet Defaults

A default subnet may be specified for a network using the **ip default-network** *network* global configuration command. The *network* argument must have a subnet address value (where the host portion is zero and the subnet portion is not zero).

If the router is unable to route a datagram to a subnet of a network and a default subnet exists, then the datagram is delivered to that subnet (if directly connected) or to another router along the route to the default subnet. A default subnet is different than a default network; it applies only to subnets of a particular network. Suppose, for example, that network *131.108.0.0* was subnetted on the third octet and that subnet 45, or 131.108.45.0 was to be the default subnet. The command specifying the default would be default network *131.108.45.0*. The EXEC command **show ip route** *network* displays the default subnet for the specified network.

Redistributing Routing Information

In addition to running multiple routing protocols simultaneously, the router can redistribute information from one routing protocol to another. For example, you can instruct the router to re-advertise IGRP-derived routes using the RIP protocol, or to re-advertise static routes using the IGRP protocol.

The metrics of one routing protocol do not necessarily translate into the metrics of another routing protocol. For example, the RIP metric is a hop count, the HELLO metric is a delay, and the IGRP metric is a combination of five quantities. In such situations, an artificial metric is assigned to the redistributed route. Because of this unavoidable tampering with dynamic information, the careless exchange of routing information between different routing protocols can create routing loops, which can seriously degrade network operation.

Supported Metric Translations

This section describes the few supported automatic metric translations between the routing protocols. These descriptions assume that you have not defined a default redistribution metric, which replaces metric conversions (see the section "Setting Default Metrics" in this chapter).

RIP can automatically redistribute static routes and HELLO-derived routes. RIP assigns static routes a metric of 1 (directly connected) and converts HELLO metrics in accordance with Table 1-2.

Values are derived from the mapping function defined by Dave Mills and the gated developers at the Cornell University Theory Center.

EGP can automatically redistribute static routes and RIP-, HELLO-, and IGRP-derived routes. EGP assigns the metric three to all static and derived routes.

BGP does not normally send metrics in its routing updates.

HELLO can automatically redistribute static routes and RIP- and IGRP-derived routes. HELLO assigns static routes a metric of 100 (directly connected) and converts RIP metrics in accordance with Table 1-2. HELLO advertises IGRP-derived routes with a metric equal to the delay portion of the IGRP metric or to 100, whichever is larger. Ethernets have a delay of 1 millisecond and serial links have a phase delay of 20 milliseconds; HELLO assigns a metric of 100 to routes using these media.

IGRP can automatically redistribute static routes and information from other IGRP-routed autonomous systems. IGRP assigns static routes a metric that identifies them as directly connected. IGRP does not change the metrics of routes derived from IGRP updates from other autonomous systems.

Note that any protocol can redistribute other routing protocols if a default metric is in effect (see the section "Setting Default Metrics" in this chapter).

Table 1-2 RIP and HELLO Metric Transformations

From HELLO	To RIP	From RIP	To HELLO
0	0	0	0
1-100	1	1	100
101-148	2	2	200
149-219	3	3	300
220-325	4	4	325
326-481	5	5	481
482-713	6	6	713
714-1057	7	7	1057
1058-1567	8	8	1567
1568-2322	9	9	2322
2323-3440	10	10	3440
3441-5097	11	11	5097

5098-7552	12	12	7552
7553-11190	13	13	11190
11191-16579	14	14	16579
16580-24564	15	15	24564
24565-30000	16	16	30000

Passing Routing Information Among Protocols

By default, the router does not exchange information among different routing protocols. If you want to pass routing information among routing protocols, use the **redistribute** router subcommand. Full syntax for this command follows.

redistribute *process-name* [*AS-number*]
no redistribute *process-name* [*AS-number*]

The argument *process-name* specifies a routing information source using one of the following keywords:

- **static**
- **rip**
- **bgp**
- **egp**
- **hello**
- **igrp**

Use the optional argument *AS-number* when you specify the **igrp** or **egp** keyword, to specify the autonomous system number.

Use the **no redistribute** command with the appropriate arguments to remove this function.

Example:

To redistribute RIP-derived information using the HELLO protocol, enter these commands:

```
router hello
redistribute rip
```

To end redistribution of information from a routing protocol, use the **no redistribute** router subcommand, supplying the appropriate arguments.

Redistributed routing information should always be filtered by the **distribute-list out** router subcommand described in the section “Filtering Outgoing Information” in this chapter. This ensures that only those routes intended by the administrator are passed along to the receiving routing protocol.

When redistributing information between IGRP processes, use the **default-information** router subcommand. The full syntax of this command follows:

default-information {**in**|**out**}
no default-information {**in**|**out**}

This subcommand controls the handling of default information between multiple IGRP processes. The **no default-information in** subcommand causes IGRP exterior or default routes to be suppressed when received by an IGRP process. Normally exterior routes are always accepted. The **no default-information out** subcommand causes IGRP exterior routes to be suppressed in updates. Default information is normally passed between IGRP processes when doing redistribution.

The default network of *0.0.0.0* used by RIP and HELLO cannot be redistributed by IGRP.

Setting Default Metrics

The **default-metric** router subcommand, used in conjunction with the **redistribute** router subcommand, causes the current routing protocol to use the same metric value for all redistributed routes. (Redistributed routes are those routes established by other routing protocols.) A default metric helps solve the problem of redistributing routes with incompatible metrics; whenever metrics do not convert, using a default metric provides a reasonable substitute and enables the redistribution to proceed.

The **default-metric** router subcommand has two forms, depending on the routing protocol specified in the **redistribute** subcommand.

For RIP, EGP, BGP, and HELLO, which use scalar, single-valued metrics, the subcommand has this syntax:

default-metric *number*

The argument *number* is the default metric value (an unsigned integer) appropriate for the specified routing protocol.

For IGRP, the **default-metric** router subcommand has this syntax:

default-metric *bandwidth delay reliability loading mtu*

- The argument *bandwidth* is the minimum bandwidth of the route in kilobits per second.
- The argument *delay* is the route delay in tens of microseconds.
- The argument *reliability* is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100x25 reliability).
- The argument *loading* is the effective bandwidth of the route in kilobits per second.
- The argument *mtu* is the minimum Maximum Transmission Unit (MTU) of the route.

Example:

For example, consider a router in autonomous system 109 using both the HELLO and IGRP routing protocols. To advertise IGRP-derived routes using the HELLO protocol and to assign the IGRP-derived routes a HELLO metric of 10,000, the configuration subcommands are:

```
router hello
```

```
default-metric 10000
redistribute igrp 109
```

Figure 1-10 shows this type of redistribution.

Figure 1-10 Assigning Metrics for Redistribution

Use the **no default metric** command to return the routing protocol to using the built-in, automatic metric translations.

Special Routing Configuration Techniques

This section describes configuration techniques for special situations and requirements.

Configuring Static Routes

The **ip route** global configuration command is used to establish static routes. A static route is appropriate if the router cannot dynamically build a route to the destination.

ip route *network address* [*distance*]

The argument *network* is the Internet address of the target network or subnet, and the argument *address* is the Internet address of a router that can reach that network. The optional *distance* argument specifies an administrative distance.

If you specify an administrative distance, you are flagging a static route that may be overridden by dynamic information.

Example 1:

In the example below, packets for network *10.0.0.0* will be routed to the router at 131.108.3.4:

```
ip route 10.0.0.0  131.108.3.4
```

Example 2:

Occasionally, you will need to set up static routes to particular destinations. In this example, the router is configured for static routes to both network and subnet routes.

```
ip route 131.22.3.21 192.31.7.19
ip route 145.42.6.63 192.31.7.19
ip route 145.45.3.24 192.31.7.19 150
```

Enabling and Disabling Split Horizon for IP Networks

Normally, routers that are connected to broadcast-type IP networks and that use distance vector routing protocols employ the *split horizon* mechanism to prevent routing loops. Split horizon blocks information about routes from being advertised by a router out any interface from which that information originated. This behavior usually optimizes communications among multiple routers, particularly when links are broken. However, with nonbroadcast networks such as frame relay and SMDS, situations can arise for which this behavior is less than ideal.

Use the **no ip split-horizon** interface subcommand to disable the split horizon mechanism.

ip split-horizon
no ip split-horizon

For all interfaces except those for which either frame relay or SMDS encapsulation is enabled, the default condition for this command is **ip split-horizon**; in other words, the split horizon feature is active. If the interface configuration includes either the **encapsulation frame-relay** or **encapsulation smds** commands, the default is for split horizon to be disabled. Split horizon is not disabled by default for interfaces using any of the X.25 encapsulations.

Note: For networks that include links over X.25 PSNs, the **neighbor** interface subcommand can be used to defeat the split horizon feature. You can as an alternative *explicitly* specify the **no ip split-horizon** command in your configuration. However, if you do so, you *must* similarly disable split horizon for all routers in any relevant multicast groups on that network.

If split horizon has been disabled on an interface and you wish to enable it, use the **ip split-horizon** interface subcommand to restore the split horizon mechanism.

Note: In general, Cisco recommends against changing the state of the default for this interface subcommand unless you are certain that your application requires doing so to properly advertise routes. Remember that if split horizon is disabled on a serial interface (and that interface is attached to a packet-switched network), you *must* disable split horizon for all routers in any relevant multicast groups on that network.

Example:

The following illustrates a simple example of disabling split horizon on a serial link. In this example, the serial link is connected to an X.25 network.

```
interface serial 0
encapsulation x25
no ip split-horizon
```

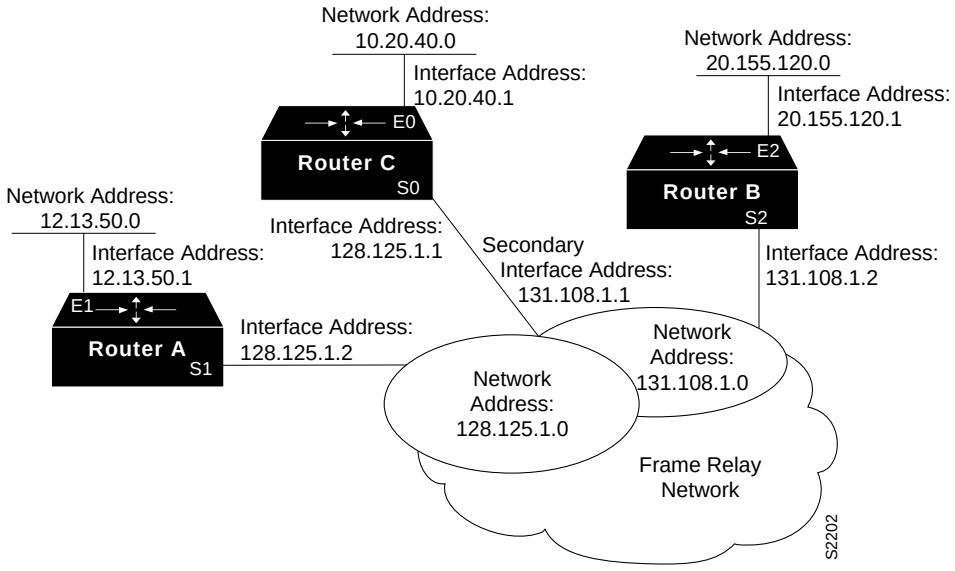
Example of Implicit Split Horizon Conditions

A typical situation in which the **no ip split-horizon** command would be useful is illustrated in Figure 14-10a. This figure depicts two IP subnets that are both accessible via a serial interface on Router C (connected to frame relay network). In this example, the serial interface on Router C accommodates one of the subnets via the assignment of a secondary IP address.

The Ethernet interfaces for Router A, Router B, and Router C (connected to IP networks 12.13.50.0, 10.20.40.0, and 20.155.120.0) all have split horizon *enabled* by default, while the serial interfaces connected to networks 128.125.1.0 and 131.108.1.0 all have split horizon *disabled* by default. The partial interface configuration specifications for each router that follow Figure 1-11 illustrate that the **ip split-horizon** interface subcommand is *not* explicitly configured under normal conditions for any of the interfaces.

In this example, split horizon must be disabled in order for network 128.125.1.0 to be advertised into network 131.108.1.0, and vice versa. These subnets overlap at Router C, interface S0. If split horizon were enabled on serial interface S0, it would not advertise a route back into the frame relay network for either of these subnets.

Figure 1-11 Disabled Split Horizon Example for Frame Relay Network



Example interface configuration for Router A:

```
interface ethernet 1
ip address 12.13.50.1
!
interface serial 1
ip address 128.125.1.2
encapsulation frame-relay
```

Example interface configuration for Router B:

```
interface ethernet 2
ip address 20.155.120.1
!
interface serial 2
ip address 131.108.1.2
encapsulation frame-relay
```

Example interface configuration for Router C:

```
interface ethernet 0
ip address 10.20.40.1
!
interface serial 0
ip address 128.125.1.1
ip address 131.108.1.1 secondary
encapsulation frame-relay
```

IGRP Metric Adjustments

The following **metric** router subcommands alter the default behavior of IGRP routing and metric computation, and allow the tuning of the IGRP metric calculation for a particular Type of Service (TOS).

metric weights *TOS K1 K2 K3 K4 K5*
no metric weights

The *TOS* parameter currently must always be zero. Parameters *K1* through *K5* are constants in the equation that converts an IGRP metric vector into a scalar quantity.

The composite IGRP metric is computed according to the following formula:

metric = [K1*bandwidth + (K2*bandwidth)/(256 - load) + K3*delay] *
[K5/(reliability + K4)]

If K5 == 0, then there is no reliability term.

The default version of IGRP has K1 == K3 == 1, K2 == K4 == K5 == 0 .

Delay is in units of 10 microseconds. This gives a range of 10 microseconds to 168 seconds. A delay of all ones indicates that the network is unreachable.

Bandwidth is inverse minimum bandwidth of the path in bits per second scaled by a factor of 10e10. The range is from a 1200 bps line to 10 Gbps.

Because of the somewhat unusual units used for bandwidth and delay, some examples seem in order. These are the default values used for several common media.

	Delay	Bandwidth
Satellite	200,000 (2 sec)	20 (500 Mbit)
Ethernet	100 (1 ms)	1,000
1.544 Mbit	2000 (20 ms)	6,476
64 Kbit	2000	156,250
56 Kbit	2000	178,571
10 Kbit	2000	1,000,000
1 Kbit	2000	10,000,000

Reliability is given as a fraction of 255. That is, 255 is 100% reliability, or a perfectly stable link.

Load is given as a fraction of 255. A load of 255 indicates a completely saturated link.

Use the **no metric weights** command to return these five constants to their default values.

The IGRP-only router subcommand **metric holddown** can be used to disable holddown. The syntax is as follows:

metric holddown
no metric holddown

Note: This command assumes that the entire AS is running Version 8.2(5) or more recent software since the hop count is used to avoid information looping. Using it with earlier software will cause problems.

The IGRP-only router subcommand **metric maximum-hops** sets a maximum hop count:

metric maximum-hops *hops*

no metric maximum-hops *hops*

This command causes the IP routing software to advertise as unreachable routes with a hop count greater than the value assigned to the *hops* argument. This is a safety mechanism that breaks any potential count-to-infinity problems. The default value is 100 hops; the maximum value is 255.

Example:

In the following example, a router in AS 71, attached to network *15.0.0.0*, wants a maximum hop count of 200, doubling the default. The network administrators decided to do this because they have a complex WAN that may generate a large hop count under normal (nonlooping) operations. (Other commands are needed between the network command and the metric command in a real-world situation; this is not a complete configuration example).

```
router igrp 71
network 15.0.0.0
metric maximum-hops 200
```

Keepalive Timers

It is possible to tune the IP routing support in the Cisco software to enable faster convergence of the various IP routing algorithms and hence quicker fall-back to redundant routers. The total effect is to minimize disruptions to end users of the network in situations where quick recovery is essential.

The network administrator can configure the keepalive interval, the frequency at which the Cisco router sends messages to itself (Ethernet and Token Ring) or to the other end (serial) to ensure a network interface is alive. The interval in previous software versions was ten seconds; it is now adjustable in one second increments down to one second. An interface is declared down after three update intervals have passed without receiving a keepalive packet.

The syntax for the **keepalive** interface subcommand is:

keepalive [*seconds*]
no keepalive

If the optional argument *seconds* is not specified, a default of ten seconds is assumed.

Example:

In the following example, the keepalive interval is set to three seconds.

```
interface ethernet 0
keepalive 3
```

Setting the keepalive timer to a low value is very useful for rapidly detecting Ethernet interface failures (transceiver cable disconnecting, cable unterminated, and so on).

A typical serial line failure involves losing Carrier Detect (CD). Since this sort of failure is typically noticed within a few milliseconds, adjusting the keepalive timer for quicker routing recovery is generally not useful.

Note: When adjusting the keepalive timer for a very low bandwidth serial interface, it is possible to have large datagrams delay the smaller keepalive packets long enough to cause the line protocol to spuriously go down. Be careful when using this feature.

Adjustable Routing Timers

The basic timing parameters for IGRP, EGP, RIP, and HELLO are adjustable. Since these routing protocols are executing a distributed, asynchronous routing algorithm, it is important that these timers be the same for all routers in the network.

To adjust timers, use the **timers basic** router subcommand. Full syntax for this command follows:

timers basic *update invalid holddown flush* [*sleeptime*]
no timers basic

- The argument *update* is the rate at which updates are sent. This is the fundamental timing parameter of the routing protocol.
- The argument *invalid* is an interval of time after which a route is declared invalid; it should be three times the value of *update.*
- The argument *holddown* is the interval during which routing information regarding better paths is suppressed. It should be at least three times the value of *update.*
- The argument *flush* is the amount of time that must pass before the route is removed from routing table; it must be at least the sum of *invalid* and *holddown*.
- The argument *sleeptime* is used with the IGRP routing protocol only and sets the number of milliseconds after the last update was sent to postpone the routing updates.

Note that the first four values for the **timers basic** command arguments are given in seconds; only the optional *sleeptime* argument is given in milliseconds.

Use the **no timers basic** command to reset the defaults.

Note: The current and default timer values can be seen by inspecting the output of the EXEC command **show ip protocols**. The relationships of the various timers should be preserved as described above.

Example 1: IGRP

In the following example, updates are broadcast every 5 seconds. If a router is not heard from in 15 seconds, the route is declared unusable. Further information is in holddown for an additional 15 seconds. At the end of the holddown period, the route is flushed from the routing table.

```
router igrp 88
timers basic 5 15 15 30
```

This example illustrates the *sleeptime* option of the command. It postpones routing updates for 1000 milliseconds.

```
router igrp 109
timers basic 5 15 15 30 1000
```

Example 2: EGP

When **timers basic** is used with EGP, the update time and holddown time are ignored. For example, the commands below will set the invalid time for EGP to 100 seconds and the flush time to 200 seconds.

```
router egp 47
timers basic 0 100 0 200
```

Gateway Discovery Protocol (GDP)

The Gateway Discovery Protocol (GDP) allows hosts to dynamically detect the arrival of new routers as well as determine when a router goes down. Host software is needed to take advantage of this protocol. Unsupported, example GDP clients can be obtained from Cisco Systems.

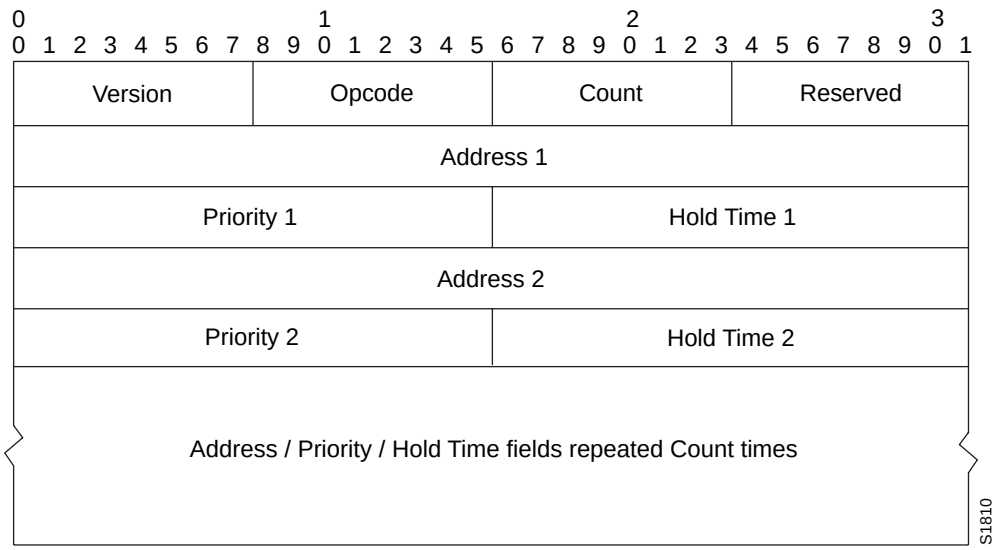
GDP is not a standard; it is a protocol designed by Cisco Systems to meet the customers' needs. Work is in progress to establish GDP or similar protocol as a standard means of discovering routers. The current GDP implementation is described next in more detail.

For ease of implementation on a variety of host software, GDP is based on the User Datagram Protocol (UDP). The UDP source and destination ports of GDP datagrams are both set to 1997 (decimal).

There are two types of GDP messages: *Report* and *Query.* On broadcast media *Report* message packets are periodically sent to the IP broadcast address announcing that the router is present and functioning. By listening for these *Report* packets, a host can detect a vanishing or appearing router. If a host issues a *Query* packet to the broadcast address, the routers each respond with a *Report* sent to the host's IP address. On nonbroadcast media, routers send *Report* message packets only in response to *Query* message packets. The protocol provides a mechanism for limiting the rate at which *Query* messages are sent on nonbroadcast media.

Figure 1-12 shows the format of the GDP *Report* message packet format. A GDP *Query* message packet has a similar format, except that the Count field is always zero and no address information is present.

Figure 1-12 GDP Report Message Packet Format



The fields in the *Report* and *Query* messages are described next.

- **Version**—8-bit field containing the protocol version number. The current GDP version number is 1. If an unrecognized version number is found, the GDP message must be ignored.
- **Opcode**—8-bit field that describes the GDP message type. Unrecognized opcodes must be ignored. Opcode 1 is a *Report* message and opcode 2 is a *Query* message.
- **Count**—8-bit field that contains the number of address, priority, and hold time tuples in this message. A *Query* message has a Count field value of zero. A *Report* message has a Count field value of 1 or greater.
- **Reserved**—8-bit reserved field; it must be set to zero.
- **Address**—32-bit fields containing the IP address of a router on the local network segment. There are no other restrictions on this address. If a host encounters an address that it believes is not on its local network segment, then the host should quietly ignore that address.
- **Priority**—16-bit fields that indicate the relative quality of the associated address. The numerically larger the value in the Priority Field, the *better* the address should be considered.
- **Hold Time**—16-bit fields. On broadcast media, the number of seconds the associated address should be used as a router without hearing further *Report* messages regarding that address. On nonbroadcast media, such as X.25, this is the number of seconds the requester should wait before sending another *Query* message.

Numerous actions can be taken by the host software listening to GDP packets.One possibility is to flush the host's ARP cache whenever a router appears or disappears.A more complex possibility is to update a host routing table based on the coming and going of routers. The particular course of action taken depends on the host software and the customer's requirements.

Using GDP Commands

The **ip gdp** interface subcommand enables GDP processing on an interface. Full syntax for this command follows:

ip gdp
no ip gdp

If you use this form of the **ip gdp** subcommand, you use only the default parameters. The default parameters are:

- A reporting interval of five seconds for broadcast media such as Ethernets, and zero seconds (never) for nonbroadcast media such as X.25
- A priority of 100
- A hold time of 15 seconds

If you want to alter some of these parameters, you can use one of the following subcommands:

ip gdp priority *number*
ip gdp reporttime *seconds*

ip gdp holdtime *seconds*

The priority keyword takes a *number* parameter and alters the priority from its default of 100. A larger number signifies a higher priority.

The **reporttime** keyword takes a time parameter in seconds.

The **holdtime** keyword also takes a time parameter in seconds.

When enabled on an interface, GDP updates report the primary and secondary IP addresses of that interface.

Example:

In the example below, GDP is enabled on interface Ethernet 1 with a report time of ten seconds, and priority and hold time set to their defaults (because none are specified).

```
ip gdp reporttime 10
```

IP Routing Protocols Configuration Examples

This section contains complete configuration examples of the IP routing protocols.

Static Routing Redistribution

In the example below, three static routes are specified, two of which wish to have the IGRP process advertise. Do this by specifying the **redistribute static** subcommand, then specify-

ing an access list that allows only those two networks to be passed to the IGRP process. Any redistributed static routes should be sourced by a single router to minimize the likelihood of creating a routing loop.

Example:

```
ip route 192.1.2.0  192.31.7.65
ip route 193.62.5.0  192.31.7.65
ip route 131.108.0.0  192.31.7.65
access-list 3 permit 192.1.2.0
access-list 3 permit 193.62.5.0
router igrp 109
network 192.31.7.0
redistribute static
distribute-list 3 out static
```

RIP and HELLO Redistribution

Consider a wide area network at a university that uses RIP as an interior routing protocol. Assume the university wants to connect its wide area network to a regional network, *128.1.1.0*, which uses HELLO as the routing protocol. The goal in this case is to advertise the networks in the university network to the routers on the regional network. The commands for the interconnecting router are:

Example:

```
router hello
network 128.1.1.0
redistribute rip
default-metric 10000
distribute-list 10 out rip
```

In this example, the **router** command starts a HELLO routing process. The **network** subcommand specifies that network *128.1.1.0* (the regional network) is to receive HELLO routing information. The **redistribute** subcommand specifies that RIP-derived routing information be advertised in the HELLO routing updates. The **default-metric** subcommand assigns a HELLO delay of 10,000 to all RIP-derived routes.

The **distribute-list** router subcommand instructs the router to use access list 10 (not defined in this example) to limit the entries in each outgoing HELLO update. The access list prevents unauthorized advertising of university routes to the regional network.

This example could have specified automatic conversion between the RIP and HELLO metrics. However, in the interest of routing table stability, it is not desirable to do so. Instead, this example limits the routing information exchanged to availability information only.

IGRP Redistribution

Each IGRP routing process can provide routing information to only one autonomous system; the router must run a separate IGRP process and maintain a separate routing database for each autonomous system it services. However, you can transfer routing information

between these routing databases.

Suppose the router has one IGRP routing process for network *15.0.0.0* in autonomous system 71 and another for network *192.31.7.0* in autonomous system 109, as the following commands specify:

Example 1:

```
router igrp 71
network 15.0.0.0
router igrp 109
network 192.31.7.0
```

To transfer a route to *192.31.7.0*, to the database of the first routing process (without passing any other information about autonomous system 109), use the following commands:

Example 2:

```
router igrp 71
redistribute igrp 109
distribute-list 3 out igrp 109
access-list 3 permit 192.31.7.0
```

Third-Party EGP Support

In this example configuration, the Cisco router is in AS 110 communicating with an EGP neighbor in AS 109 with address *131.108.6.5*. Network *131.108.0.0* is advertised as originating within AS 110. The configuration specifies that two routers, *131.108.6.99* and *131.108.6.100*, should be advertised as third-party sources of routing information for those networks that are accessible through those routers. The global configuration commands also specify that those networks should be flagged as internal to AS 110.

Example:

```
autonomous-system 110
router egp 109
network 131.108.0.0
neighbor 131.108.6.5
neighbor 131.108.6.5  third-party 131.108.6.99 internal
neighbor 131.108.6.5  third-party 131.108.6.100 internal
```

Backup EGP Router

The following example configuration illustrates a router that is in AS 110 communicating with an EGP neighbor in AS 109 with address 131.108.6.5. Network *131.108.0.0* is advertised with a distance of zero, and networks learned by RIP are being advertised with a distance of five. Access list 3 filters which RIP-derived networks are allowed in outgoing EGP updates.

Example:

```
autonomous-system 110
router egp 109
network 131.108.0.0
neighbor 131.108.6.5
redistribute rip
default-metric 5
distribute-list 3 out rip
```

Maintaining IP Routing Operations

The IP routing protocols have one command to help you maintain the IP routing operations.

Use the privileged EXEC command **clear ip route** to remove a dynamic route from the routing table. Enter this command at the EXEC prompt:

clear ip route *{network|*}*

The argument *network* is the network address portion of the routing table entry to be removed. If you specify * all routes are removed. Note that routing entries you remove with the **clear ip route** command may reappear because of dynamic routing, or because they are floating static routes.

To remove a static route from the routing table, use the **no ip route** global configuration command with the appropriate arguments for the type of static route.

Monitoring IP Routing Operations

This section describes the EXEC commands you may use to monitor IP routing operations configured on your router.

Displaying the BGP Routing Table

Use the **show ip bgp** EXEC command to display a particular network in the BGP routing table. Enter this command at the EXEC prompt:

show ip bgp [*network*]

The optional argument *network* is a network number, and is entered to display a particular network in the BGP routing table.

Following is sample output:

```
BGP Table Version is 19
Status codes: * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, * - incomplete

   Network       Next Hop        BGP Peer        Metric Weight Path
```

```
*> 128.145.0.0   0.0.0.0         0.0.0.0               0      *
*> 192.68.151.0  0.0.0.0         0.0.0.0               0      *
*> 151.142.0.0   0.0.0.0         0.0.0.0               0      *
*> 150.136.0.0   0.0.0.0         0.0.0.0               0      *
*> 192.91.180.0  0.0.0.0         0.0.0.0               0      *
*> 132.249.0.0   0.0.0.0         0.0.0.0               0      *
*> 128.18.0.0    0.0.0.0         0.0.0.0               0      *
*> 192.53.65.0   0.0.0.0         0.0.0.0               0      *
*> 192.53.66.0   0.0.0.0         0.0.0.0               0      *
*> 192.67.67.0   0.0.0.0         0.0.0.0               0      *
*> 192.53.48.0   0.0.0.0         0.0.0.0               0      *
*> 192.31.7.0    0.0.0.0         0.0.0.0               0      *
*> 130.93.0.0    0.0.0.0         0.0.0.0               0      *
*> 192.12.33.0   0.0.0.0         0.0.0.0               0      *
*  131.108.0.0   131.108.6.68    131.108.6.68          0      68 i
*>               0.0.0.0         0.0.0.0               0      i
```

In the display:

- The `Table Version` is the internal version number for the table. This is incremented any time the table changes.
- The first three characters indicate the status. The `*` (asterisk) indicates that the table entry is valid. The > character indicates that that table entry is the best entry to use for that network. The lowercase `i` indicates that the table entry was learned via an internal BGP session.
- The `Next Hop` entry is the IP address of the next system to use when forwarding a packet to the destination network. An entry of *0.0.0.0* indicates that the local router has some nonBGP route to this network.
- The `BGP peer` entry is the IP address of the BGP peer that we learned the route from. An entry of *0.0.0.0* indicates that the local router injected the route into BGP.
- The `Metric` field, if any, is the value of the inter-autonomous system metric. This is frequently not used.
- The `Weight` field is set through the use of AS filters.
- The path is the autonomous system path to the destination network. At the end of the path is the origin code for the path. The lowercase `i` indicates that the entry was originated with the local IGP and advertised with a **network** subcommand. A lowercase `e` indicates that the route originated with EGP. An asterisk (`*`) indicates that the origin of the path is not clear. Usually this a path that is redistributed into BGP from an IGP.

Displaying BGP Neighbors

Use the **show ip bgp neighbors** EXEC command to display detailed information on the TCP and BGP connections to individual neighbors. Enter this command at the EXEC prompt:

show ip bgp neighbors

Following is sample output:

```
BGP neighbor is 131.108.6.68, remote AS 68, external link
  BGP state = 3, table version = 19, up for 0:12:38
```

```
  Last read 0:00:37, hold time is 180, keepalive interval is 60 seconds
  Received 48 messages, 0 NOTIFICATIONS
  Sent 51 messages, 0 NOTIFICATIONS
  Connections established 1; dropped 0
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 131.108.6.69, 12288   Foreign host: 131.108.6.68, 179

Enqueued packets for retransmit: 0, input: 0, saved: 0

Event Timers (current time is 835828):
Timer:       Retrans   TimeWait    AckHold    SendWnd  KeepAlive
Starts:           20          0         18          0          0
Wakeups:           1          0          2          0          0
Next:              0          0          0          0          0

iss:     60876  snduna:     62649  sndnxt:     62649    sndwnd:   1872
irs:  95187024  rcvnxt:  95188733  rcvwnd:      1969  delrcvwnd:    271

SRTT: 364 ms, RTTO: 1691 ms, RTV: 481 ms, KRTT: 0 ms
minRTT: 4 ms, maxRTT: 340 ms, ACK hold: 300 ms
Flags: higher precedence

Datagrams (max data segment is 1450 bytes):
Rcvd: 36 (out of order: 0), with data: 18, total data bytes: 1708
Sent: 40 (retransmit: 1), with data: 36, total data bytes: 1817
```

In the display:

- The first line lists the IP address of the BGP neighbor and it's AS number. If the neighbor is in the same AS as the local router, then the link between them is internal. Otherwise, it is considered external.
- The `BGP state` indicates the internal state of this BGP connection. The `table version` indicates that the neighbor has been updated with this version of the primary BGP routing table. The `up time` indicates the amount of time that the underlying TCP connection has been in existence.
- The `last read` time is the time that BGP last read a message from this neighbor. The `hold time` is the maximum amount of time that can elapse between messages from the peer. The `keepalive interval` is the time period between sending keepalive packets, which help insure that the TCP connection is up.
- The number of `Received messages` indicates the number of total BGP messages received from this peer, including keepalives. The number of notifications is the number of error messages that we have received from the peer.
- The number of `Sent messages` indicates the total number of BGP messages that have been sent to this peer, including keepalives. The number of notifications is the number of error messages that we have sent to this peer.
- The number of `Connections established` is a count of the number of times that we have established a TCP connection and the two peers have agreed speak BGP with each other. The number of dropped connections is the number of times that a good connection has failed or been taken down.
- The remainder of the display describes the status of the underlying TCP connection. See the description of **show ip tcp** for more information.

Displaying EGP Statistics

Use the show ip egp command to display detailed statistics on EGP connections. Enter this command at the EXEC prompt:

show ip egp

The command output includes detailed information about neighbors. Sample output follows. Table 1-3 describes the fields seen.

```
Local autonomous system is 109
  EGP Neighbor FAS/LAS State   SndSeq RcvSeq Hello Poll j/k Flags
  10.3.0.27      1/109 IDLE       625  61323    60  180   0 Perm, Act
* 10.2.0.37      1/109 UP 12:29   250  14992    60  180   3 Perm, Act
* 10.7.0.63      1/109 UP  1d19   876  10188    60  180   4 Perm, Pass
```

Table 1-3 Show IP EGP Field Descriptions

Field	Description
EGP Neighbor	Address of the EGP neighbor.
FAS	Foreign Autonomous System number
LAS	Local Autonomous System number
State	State of the connection between peers
SndSeq	Send sequence number
RcvSeq	Receive sequence number
Hello	Interval between HELLO/I-HEARD-YOU packets
Poll	Interval between POLL/UPDATE packets
j/k	Measure of reachability; 4 is perfect
Flags	• Perm—Permanent (currently no alternatives •Act—Active, controlling the connection •Pass—Passive, neighbor controls the connection

Displaying Routing Protocol Parameters and Status

Use the EXEC command **show ip protocols** to display the parameters and current state of the active routing protocol process. Enter this command at the EXEC prompt:

show ip protocols

Following is sample output:

```
Routing Protocol is "igrp 109"
  Sending updates every 90 seconds, next due in 88 seconds
  Invalid after 270 seconds, hold down for 280, flushed after 630
  Outgoing update filter list for all routes is not set
  Incoming update filter list for all routes is not set
  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
  Redistributing: igrp 109
  Routing for Networks:
    131.108.0.0
    192.31.7.0
  Routing Information Sources:
    Gateway         Distance  Last Update
    131.108.2.201        100      0:00:08
    131.108.200.2        100      5d15
    131.108.2.200        100      0:00:09
    131.108.2.203        100      0:00:11
```

The information displayed by **show ip protocols** is useful in debugging routing operations. Information in the `Routing Information Sources` field of the **show ip protocols** output can help you identify a router suspected of delivering bad routing information.

In the display:

- The Routing Protocol field specifies the routing protocol used.
- Sending updates field specifies the time between sending updates, as well as precisely when the next is due to be sent.
- The Invalid field specifies the value of the invalid parameter.
- The hold down field specifies the current value of the holddown parameter.
- The flushed field specifies the time in seconds after which the individual routing information will be thrown (flushed) out.
- The outgoing update field specifies whether the outgoing filtering list has been set.
- The incoming update field specifies whether the incoming filtering list has been set.
- The Default networks field specifies how these networks will be handled in both incoming and outgoing updates.
- The IGRP metric field specifies the value of the K0-K5 metrics as well as the maximum hopcount.
- The Redistributing field lists the protocol that is being redistributed.
- The Routing field specifies the networks that the routing process is currently injecting routes for.
- The Routing Information Sources field lists all the routing sources the router is using to build its routing table. For each source, you will see displayed:
  - The IP address
  - The administrative distance
  - The time the last update was received from this source

Displaying the Routing Table

Use the EXEC command **show ip route** to display the current state of the routing table. Enter this command at the EXEC prompt:

show ip route [*address*]

When entered with the optional *address* argument, the command displays detailed routing information for the specified network or subnet.

Following is sample output from this command when entered without an address:

```
Codes: I - IGRP derived, R - RIP derived, H - HELLO derived
       C - connected, S - static, E - EGP derived, B - BGP derived
       * - candidate default route

Gateway of last resort is 131.108.1.6 to network 129.140.0.0

I*Net 10.0.0.0 [100/28476] via 192.31.7.130, 11 sec, Serial2
I*Net 129.140.0.0 [100/9576] via 131.108.1.6, 21 sec, Ethernet0
                             via 131.108.2.6, 21 sec, Ethernet1
                             via 192.31.7.26, 21 sec, Ethernet2
I Net 128.18.0.0 [100/8576] via 192.31.7.130, 11 sec, Serial2
C Net 192.31.7.0 is subnetted (mask is 255.255.255.240), 3 subnets
I     192.31.7.144 [100/1002100] via 192.31.7.114, 30 sec, Serial6
C     192.31.7.16 is directly connected, Ethernet2
C     192.31.7.48 is directly connected, Serial0
I Net 192.12.33.0 [100/8576] via 192.31.7.130, 11 sec, Serial2
C Net 131.108.0.0 is subnetted (mask is 255.255.255.0), 1 subnet
I     131.108.205.0 [100/1004200] via 131.108.1.6, 21 sec, Ethernet0
```

In the displays, the * (asterisk) indicates a candidate default route; NET indicates a network rather than subnetwork entry. Other fields contain this information:

- The first column lists the protocol that derived the route.
- The second column lists the address of the remote network The first number in the brackets is the administrative distance of the information source; the second number is the metric for the route.
- The third column specifies the address of the router that can build a route to the specified remote network.
- The fourth column specifies the cost in seconds associated with the specified route.
- The final column specifies the interface through which the specified network can be reached.

Directly connected routers may include subnet information where appropriate.

Following is sample output from this command when entered with the address *10.0.0.0*:

```
Routing entry for 10.0.0.0 (mask 255.0.0.0)
  Known via ”igrp 109”, distance 100, metric 28476, candidate default
path
  Redistributing via igrp 109
  Last update from 192.31.7.130 on Serial2, 29 seconds ago
  Routing Descriptor Blocks:
  * 192.31.7.130, from 192.31.7.130, 29 seconds ago, 15 uses, via Serial2
      Route metric is 28476, traffic share count is 1
     Total delay is 220000 microseconds, minimum bandwidth is 1544 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 128/255, Hops 0
```

When you specify that you want information about a specific network displayed, more detailed statistics are shown, as the following sample output indicates:

- The protocol that provided the information
- The administrative distance

- The metric as provided by the protocol (IGRP, in this case)
- The redistribution protocol
- The address of the source of this routing information, along with the following:
  - Time of the last incoming update for the route and
  - The interface that the information arrived on
- Much of this information is repeated in the Routing Descriptor Block, along with:
  - Number of times this route has been used since it was added to the table
  - Total round-trip delay in seconds
  - Minimum bandwidth on the route (the smallest pipe you will encounter along the way to the remote network, in other words)
  - `Reliability,` which is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100% reliability)
  - Minimum MTU
  - `Loading` (which is the effective bandwidth of the route in kilobits per second)
  - Hop count

Debugging IP Routing

The EXEC command **debug** and the global configuration command **logging** enable you to record useful routing information. Using the privileged EXEC command **debug,** you can instruct the router to log any combination of RIP, EGP, HELLO, BGP, and IGRP routing events as well as routing table events to the console terminal. In general, these commands are entered during troubleshooting sessions with Cisco engineers. For each **debug** command, there is a corresponding **undebug** command that disables the command output.

debug ip-bgp

The EXEC command **debug ip-bgp** displays debugging information on BGP transactions.

debug ip-egp

The **debug ip-egp** command displays EGP transactions.

debug ip-egp-events

The **debug ip-egp-events** command enables logging of major EGP transactions, such as an EGP peer being acquired or discontinued. If both **debug ip-egp** and **debug ip-egp-events** are enabled, the mention of individual networks in updates is suppressed. This reduction in the logging output permits easier debugging of EGP update problems.

debug ip-hello

The **debug ip-hello** command enables logging of HELLO routing transactions.

debug ip-igrp

The **debug ip-igrp** command enables logging of IGRP routing transactions.

debug ip-rip

The **debug ip-rip** command enables logging of RIP routing transactions.

debug ip-routing

The **debug ip-routing** command enables logging of routing table events such as network appearances and disappearances.

debug ip-tcp

The **debug ip-tcp** command enables logging of significant TCP transactions such as state changes, retransmissions, and duplicate packets.

debug ip-tcp-packet *line*

The **debug ip-tcp-packet** command enables logging of each TCP packet associated with the specified line number.

debug ip-udp

The **debug ip-udp** command enables logging of UDP-based transactions.

Global Configuration Command Summary

This section provides an alphabetical list of the global commands used with the IP Routing Protocols.

[**no**] **autonomous-system** *local-AS*

Specifies an autonomous system number. The argument *local-AS* is the local autonomous system (AS) number to which the router belongs. To remove the AS number, use the **no autonomous-system** configuration command.

[**no**] **ip default-network** *network-number*

Provides a mechanism for causing a smart router to generate dynamic default information that is then passed along to other routers. The argument *network-number* is a network number.

Router Subcommand Summary

This section provides an alphabetical list of the router subcommands used with the IP routing protocols.

[**no**] **default-information** {**in**|**out**}

Controls the handling of default information between multiple IGRP processes. The **no default-information in** subcommand causes IGRP exterior or default routes to be suppressed when received by an IGRP process. Normally exterior routes are always accepted. The **no default-information out** subcommand causes IGRP exterior routes to be suppressed in updates. Default information is normally passed between IGRP processes.

default-metric *bandwidth delay reliability loading mtu*

Sets metrics for IGRP only. The argument *bandwidth* is the minimum bandwidth of the route in kilobits per second. The argument *delay* is the route delay in tens of microseconds. The argument *reliability* is the likelihood of successful packet transmission expressed as a number between 0 and 255 (255 is 100% reliability). The argument *loading* is the effective bandwidth of the route in kilobits per second. The argument *mtu* is the minimum Maximum Transmission Unit (MTU) of the route.

default-metric *number*

Sets metrics for RIP, EGP, BGP, and HELLO, which use scalar, single-valued metrics. The argument *number* is the default metric value (an unsigned integer) appropriate for the specified routing protocol.

no default-metric

Causes the current routing protocol to return to using the built-in, automatic metric translations.

[**no**] **distance** *weight* [[i*p-source-address ip-address-mask*] [*access-list-number*]]

Defines or deletes an administrative distance. The argument *weight* is an integer from 10 to 255 that specifies the administrative distance. Used alone, the argument *weight*

specifies a default administrative distance that the router uses when no other specification exists for a routing information source. The optional argument pair *ip-source-address* and *ip-address-mask* specifies a particular router or group of routers to which the weight value applies. The argument *ip-source-address* is an Internet address that specifies a router, network, or subnet. The argument *ip-address-mask* (in dotted-decimal format) specifies which bits, if any, to ignore in the address value; a set bit in the *mask* argument instructs the router to ignore the corresponding bit in the address value. The optional argument *access-list-number* is the number of a standard IP access list.

[**no**] **distribute-list** *access-list-number* **in** [*interface-name*]

Filters networks received in updates**.** The argument *access-list-number* is a standard IP access list number. Use the keyword **in** to suppress incoming routing updates. The optional argument *interface-name* specifies the interface on which the access list should be applied to incoming updates. If no interface is specified, the access list will be applied to all incoming updates.

[**no**] **distribute-list** *access-list-number* **out** [*interface-name*|*routing-process*]

Suppresses networks from being sent in updates. The argument *access-list-number* is a standard IP access list number. Use the keyword **out** to apply the access list to outgoing routing updates. To filter a routing update sent on a specific interface, you may, optionally, specify the interface. When redistributing networks, a routing process name may be specified.

[**no**] **metric holddown**

Disables or re-enables holddown (IGRP only). This assumes that the entire AS is running Version 8.2(5) or more recent software since the hop count is used to avoid information looping. Using it with Version 7.1 or earlier software will cause problems.

[**no**] **metric maximum-hops** *hops*

Causes the IP routing software to advertise as unreachable routes with a hop count greater than the value assigned to the *hops* argument (IGRP only). The default value is 100 hops; the maximum value is 255.

[**no**] **metric weights** *TOS K1 K2 K3 K4 K5*

Allows the tuning of the IGRP metric calculation for a particular Type of Service (*TOS*). The TOS parameter currently must always be zero. Parameters *K1* through *K5* are constants in the equation that converts an IGRP metric vector into a scalar quantity.

[**no**] **neighbor** *address*

Creates a list of neighbor routers. The argument *address* is the IP address of a peer router

with which routing information will be exchanged. The **no** form of the command removes an entry. The **no** form of the command returns the default.

[**no**] **neighbor** *address* **distribute-list** *list* {**in**|**out**}

Distributes neighbor information as specified in an access list *list* for BGP. You specify the access list to be applied to incoming or outgoing updates with the **in** and **out** keywords.

[**no**] **neighbor** *address* **filter-as** *number* **deny**

Filters neighbor information for an address in a specific AS for BGP. The **deny** keyword causes the information learned about that network using the specified AS to be ignored.

[**no**] **neighbor** *address* **filter-as** *number* **permit** *weight*

Filters neighbor information for an address in a specific AS for BGP. The **permit** keyword allows incoming information to be added to the routing table with a weight specified by the *weight* argument.

[**no**] **neighbor** *address* **remote-as** *autonomous-system*

Adds a neighbor entry to the routing table for BGP. The keywords *address* and *autonomous-system* specify the IP address and AS of the neighbor router.

[**no**] **neighbor** *address* **third-party** *third-party-ip-address* [**internal**|**external**]

Adds third-party information to routing updates. The argument *third-party-ip-address* is the address of the third party on the network shared by the Cisco router and the EGP peer specified by the *address* argument. The optional keyword **internal** or **external** indicates whether the third party router should be listed in the internal or external section of the EGP update. Normally, all networks are mentioned in the internal section. You can enter this command multiple times.

[**no**] **neighbor** *address* **weight** *weight*

Specifies a weight to assign to a specific neighbor connection indicated by the argument *address*. The route with the highest weight is chosen as the preferred route when multiple routes are available to a particular network.

[**no**] **network** *address* **backdoor**

Specifies a back-door route to a BGP border router that will provide better information about the network.

[**no**] **network** *network-number*

Specifies a list of networks to be advertised as originating within an AS, or for EGP, the network to be advertised to the EGP peers of an EGP routing process. The argument *network-number* is a network number. The **no** form of the command removes an entry from the list.

[**no**] **offset-list** *list* {**in**|**out**} *offset*

Adds or removes a positive offset to incoming and outgoing metrics (as indicated by the **in** and **out** keywords) for networks matching an access list (for IGRP, RIP and HELLO only). If the argument *list* is zero, the argument supplied to *offset* is applied to all metrics. If *offset* is zero, no action is taken. For IGRP, the offset is added to the delay component only.

[**no**] **passive-interface** *interface*

Disables or enables sending routing updates on an interface. The argument *interface* specifies a particular interface.

[**no**] **redistribute** *process-name* [*AS-number*]

Passes routing information among routing protocols. The argument *process-name* specifies a routing information source using one of the keywords: **static**, **rip**, **egp**, **hello**, **igrp**. Use the optional argument *AS-number* when you specify the **igrp** or **egp** keyword, to specify the autonomous system number

[**no**] **router** *protocol* [*autonomous-system*]

Creates an IP routing process. The argument *protocol* is one of these protocol-type keywords —**rip**, **egp, hello, bgp, igrp.** The IGRP, BGP, and EGP protocols use the optional argument *autonomous-system* to supply the number of an autonomous system.

[**no**] **timers basic** *update invalid holddown flush*

Adjusts timers. The argument *update* is the rate at which updates are sent. This is the fundamental timing parameter of the routing protocol. The argument *invalid* is an interval of time after which a route is declared invalid; it should be three times the value of *update*. The argument *holddown* is the interval during which routing information regarding better paths is suppressed. It should be at least three times the value of *update*. The argument *flush* is the amount of time that must pass before the route is removed from routing table; it must be at least the sum of *invalid* and *holddown*. The **no** form of the command restores the default.

[**no**] **timers bgp** *keepalive holdtime*

Adjusts BGP timers. The argument *keepalive* is the frequency in seconds with which the

router sends *keepalive* messages to its peer (default 60 seconds), and where *holdtime* is the interval in seconds after not receiving a *keepalive* message that the router declares a peer dead (default 180 seconds). The **no** form of the command restores the default.

[**no**] **timers egp** *hello poll*

Adjusts the EGP timers. The argument *hello* adjusts the interval at which hellos are sent. The default is set to 60 seconds. The argument *poll* adjusts the interval at which polling is performed. The default is set to 180 seconds; the **no** form of the command restores the default.

IP Routing Interface Subcommands

This section provides alphabetical lists of the interface subcommands used with the IP routing protocols.

ip address *address mask* [**secondary**]

Specifies the IP address on an interface. The argument *address* supplies the address; the argument *mask* the subnet mask. The optional keyword **secondary** allows multiple IP addresses per interface.

[**no**] **ip gdp**

Enables or disables GDP routing, with all default parameters. Reporting interval is 5 seconds for Ethernet and zero seconds for non broadcast media. The priority is 100 and the hold time is 15 seconds.

[**no**] **ip gdp holdtime** *seconds*

Enables or disables GDP routing, specifying *holdtime* in seconds and keeping all other parameters (priority and reporting interval) at their default settings.

[**no**] **ip gdp priority** *number*

Enables or disables GDP routing with a priority number you specify. Report time remains at five seconds for Ethernets and the holdtime remains 15 seconds.

[**no**] **ip gdp reporttime** *seconds*

Enables or disables GDP routing, with a report time you specify. The priority remains 100 and the hold time remains 15 seconds.

[**no**] **ip route** *network address*

Establishes static routes. The argument network address is the network address for which you are establishing a static route.

[**no**] **keepalive** [*seconds*]

Adjusts the keepalive timer for a specific interface. If the optional argument *seconds* is not specified, a default of ten seconds is assumed.