Chapter 1
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, OSPF, 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, the Interior Gateway Routing Protocol (IGRP), and the Open Shortest Path First (OSPF) protocol. 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.

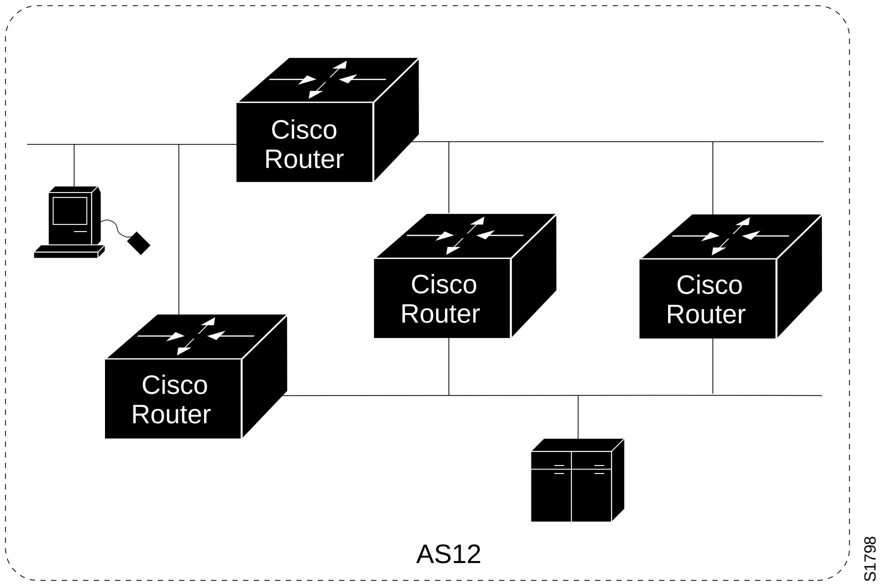
- RIP is the routing protocol used by the routed process on Berkeley-derived UNIX systems. Many networks use RIP; it works well for small, isolated, and topologically simple networks.
- Hello is an older interior routing protocol used in the early National Science Foundation (NSF) backbone network.
- IGRP, developed by Cisco Systems, focuses on large networks with complex topology and segments having different bandwidth and delay characteristics.
- OSPF, developed by the OSPF working group of the Internet Engineering Task Force (IETF), is an IGP based on *link state* technology.
- 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.

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.

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 Network Information Center (NIC), 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 use 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.

Multiple IP Routing Processes

Cisco routers can handle the simultaneous operation of up to 30 dynamic IP routing processes.

The combination of routing processes on a router can consist of the following protocols (with the limits noted):

- Any number of IGRP routing processes
- Any number of OSPF routing processes
- Any number of EGP routing processes
- One BGP routing process
- One RIP routing process

- One CHAOS routing process
- One Hello routing process

Configuration Overview

Each routing protocol must be configured separately. So, most of the configuration information is in protocol-specific subsections. The interior routing protocols are listed first, followed by the exterior protocols. With any routing protocol, follow these basic steps:

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

Step 2: Configure one or more **network** router subcommands.

Step 3: 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

All IP routing protocols 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. In addition, the routing process only advertises the (sub)nets listed in the **network** command. The IGRP routing protocol also has an autonomous system number, usually assigned by the 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.

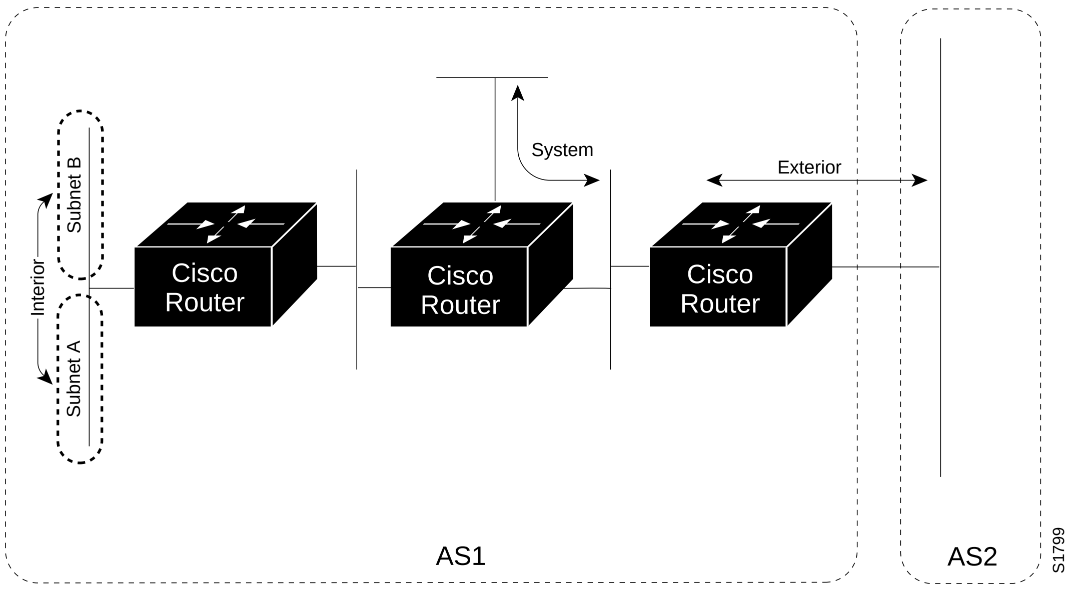
Configuring the IGRP Protocol

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 can advertise all connected and IGRP-derived networks for a particular autonomous system.

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 that are considered when identifying a gateway of last resort.

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.

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 (of directly connected networks). Note that this number must not contain subnet information. You may specify multiple **network** subcommands.

The **network** router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

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 indicate the networks directly connected to the router.

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

Unequal-Cost Load Balancing

IGRP has been enhanced to simultaneously use an asymmetric set of paths for a given destination. This feature is known as *unequal-cost load balancing*. The following general rules apply to IGRP unequal-cost load balancing:

- IGRP will accept up to four paths for a given destination network.
- The next router must be closer (have a smaller metric value) to the destination than this router.
- The metric must be within the configured *variance* of the local best metric.

Note: By using *variance*, the router can balance traffic across *all* feasible paths and can immediately converge to a new path if one of the paths should fail.

IGRP Variance Command

IGRP can balance traffic across multiple routes that have different metrics. The amount of load balancing that is performed can be controlled with the **variance** router subcommand. The command syntax is:

variance *multiplier*
no variance

The argument *multiplier* defines the range of metric values that will be accepted for load balancing. Acceptable values are nonzero, positive integers. By default, the amount of variance is set to one (equal-cost load balancing). The **no** version resets variance to one.

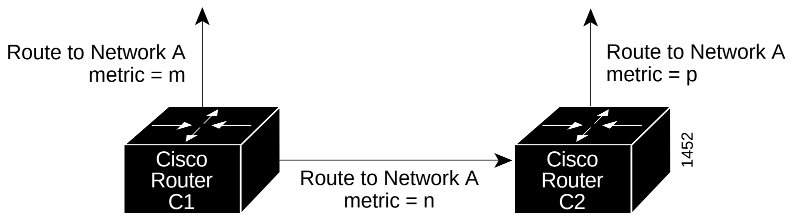
This value is used in the procedure for determining the *feasibility* of a potential route. A route is *feasible* if the next router in the path is *closer* to the destination than the current router and if the metric for the entire path is *within* the variance. Only paths that are feasible can be used for load balancing and included in the routing table. The two feasibility conditions are:

1. The local best metric must be greater than the best metric learned from the next router.
2. The *multiplier* times the local best metric for the destination must be larger than the metric through the next router

If both these conditions are met, the route is deemed feasible and can be added to the routing table.

Figure 1-3 illustrates the process of determining IGRP path feasibility.

Figure 1-3 Determining IGRP Path Feasibility



The feasibility test would work as follows:

Assume that C1 already has a route to Network A with metric m and has just received an update about Network A from C2. The best metric at C2 is p. The metric that C1 would use through C2 is n.

1. If *m* is greater than *p*, then the first condition is met.
2. If the *multiplier* times *m* is greater than or equal to *n*, then the second condition is met.

If both conditions are met, the route will be included in C1's routing table. A maximum of four paths can be in the routing table for a single destination. If there are more than four feasible paths, the four best feasible paths are used.

These conditions limit the number of cases in which load balancing can occur, but ensure that the dynamics of the network will remain stable.

Choosing the Gateway of Last Resort

The router chooses a *gateway of last resort* from the list of exterior routes that IGRP provides. The router uses the gateway (router) of last resort if it does not have a better route for a packet and the destination is not a connected network. 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, Ethernet, and serial lines running from 9,600 bps 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 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 seven update periods (630 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 OSPF Routing Protocol

This section describes the Open Shortest Path First (OSPF) routing protocol and provides the following specific discussions:

- Overview of the OSPF routing environment and conventions
- Cisco support of OSPF
- Steps in configuring OSPF for Cisco routers and details about Cisco's OSPF configuration commands

---

Note: The introductory information in this section summarizes descriptions and definitions provided in the Internet specification of OSPF Version 2 in RFC 1247. This synopsis focuses on linking Cisco's configuration command environment to the generalized OSPF capabilities; however, if you are configuring an OSPF-based internetworking scheme, refer to RFC 1247 for specific details. In general, the emphasis in this section is on Cisco's implementation of OSPF.

The OSPF **show** command descriptions, **debug** command definitions, and OSPF configuration examples are provided in subsequent sections of this chapter along with similar descriptions and examples for other IP routing protocols.

The OSPF Routing Protocol

OSPF is an Interior Gateway Protocol (IGP). As such, OSPF distributes routing information between routers belonging to a single AS. For the purposes of OSPF, an autonomous system is essentially a group of routers exchanging routing information via a common routing protocol. The OSPF protocol is based on shortest-path-first, or *link-state*, technology. This is a departure from the Bellman-Ford base, distance vector technology, used by traditional IP routing protocols such as IGRP.

The OSPF protocol was developed by the OSPF working group of the Internet Engineering Task Force (IETF). It has been designed expressly for the Internet environment, including explicit support for IP subnetting, Type of Service (TOS) based routing and the tagging of externally derived routing information. OSPF also provides for the authentication of packets, and uses IP multicast when sending/receiving packets. The OSPF (Version 2) protocol is documented in the Internet RFC 1247.

Note: Cisco currently supports only TOS 0.

The OSPF Routing Domain and Areas

OSPF allows collections of contiguous networks and hosts to be grouped together. Such a group, together with the routers having interfaces to any one of the included networks, is called an *area*. Each area runs a separate copy of the basic shortest-path-first routing algorithm. This means that each area has its own topological database.

The topology of an area is invisible from the outside of the area. Conversely, routers internal to a given area know nothing of the detailed topology external to the area. This isolation of knowledge enables the protocol to effect a marked reduction in routing traffic as compared to treating the entire Autonomous System as a single SPF domain.

With the introduction of areas, it is no longer true that all routers in the AS have an identical topological database. A router actually has a separate topological database for each area to which it is connected. Routers connected to multiple areas are called *area border routers*. Two routers belonging to the same area have, for that area, identical area topological databases.

Routing in the autonomous system takes place on two levels, depending on whether the source and destination of a packet reside in the same area (intraarea routing is used) or different areas (interarea routing is used). In intraarea routing, the packet is routed solely on information obtained within the area; no routing information obtained from outside the area can be used. This protects intraarea routing from the injection of bad routing information.

OSPF Backbones

Every OSPF routing domain AS must have a *backbone*. The backbone is a special OSPF area that must have an area id of 0.0.0.0 (or simply 0). It consists of those networks not contained in any specific area, their attached routers, and those routers that belong to multiple areas. The backbone must be contiguous. Each router's interface that is configured in Area 0 must be reachable via other routers where each interface in the path is configured as being in Area 0.

However, it is possible to define areas in such a way that the backbone is no longer contiguous—where the continuity between routers is broken. In this case, you must establish backbone continuity by configuring *virtual links*. Virtual links are useful when the backbone area is either purposefully partitioned or when restoring inadvertent breaks in backbone continuity.

OSPF Router Classifications

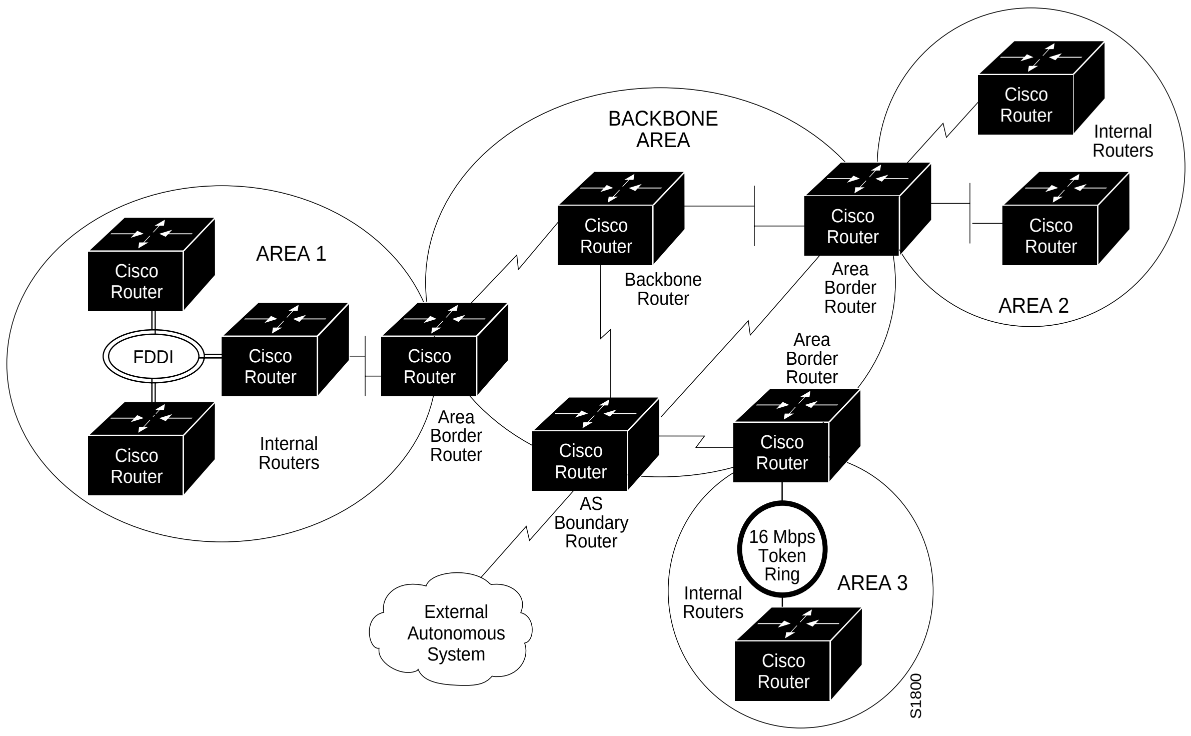
When the AS is split into OSPF areas, the routers are further divided according to function into the following four overlapping categories:

- **Internal router**—A router with all directly connected networks belonging to the same area. Routers with only backbone interfaces also belong to this category. These routers run a single copy of the basic routing algorithm.

- **Area border router**—A router that attaches to multiple areas. Area border routers run multiple copies of the basic algorithm, one copy for each attached area and an additional copy for the backbone. Area border routers condense the topological information of their attached areas for distribution to the backbone. The backbone in turn distributes the information to the other areas.
- **Backbone router**—A router that has an interface to the backbone. This includes all routers that interface to more than one area (area border routers). However, backbone routers are not required to be area border routers. Routers with all interfaces connected to the backbone are considered to be internal routers.
- **AS boundary router**—A router that exchanges routing information with routers belonging to other ASes. Such a router has AS external routes that are advertised throughout the AS. The path to each AS boundary router is known by every router in the AS. This classification is completely independent of the previous classifications; AS boundary routers may be internal or area border routers, and may or may not participate in the backbone.

Figure 1-4 illustrates the various OSPF router types and their relationships to each other and to the overall OSPF environment.

Figure 1-4 Overview of OSPF Router Types and Relationships



OSPF Routing Conventions

All OSPF routing implementations adhere to an essentially common set of rules and conventions. The following descriptions outline these conventions and, where applicable, specific characteristics of Cisco's implementation.

OSPF Physical Network Support

OSPF routing implementations support the following types of physical networks:

- **Point-to-point networks**—A network that joins a single pair of routers, such as a 56 Kbps serial line connecting a remote site to a main campus.
- **Broadcast networks**—Networks supporting more than two attached routers together that can broadcast a single physical message to all of the attached routers. Neighboring routers are discovered dynamically on these nets using OSPF's *Hello Protocol*. The Hello Protocol itself takes advantage of the broadcast capability. The protocol makes further use of multicast capabilities, if they exist. Ethernet is an example of a broadcast network type.
- **Nonbroadcast Networks**—Networks supporting more than two routers, but having no broadcast capability. Neighboring routers are also discovered on these networks using OSPF's Hello Protocol. However, due to the lack of broadcast capability, some configuration information is necessary for the correct operation of the Hello Protocol. On these networks, OSPF protocol packets that are normally multicast need to be sent to each neighboring router, in turn. An X.25 Public Data Network (PDN) is an example of a nonbroadcast network.

Support for IP Subnetting with OSPF

OSPF attaches an IP address mask to each advertised route. The mask indicates the range of addresses being described by the particular route. For example, a summary advertisement for the destination 128.185.0.0 with a mask of 255.255.0.0 actually is describing a single route to the collection of destinations 128.185.0.0 to 128.185.255.255. Including the mask with each advertised destination enables the implementation of *variable-length subnet masks*.

Note: In the current router software release, Cisco does not support variable length subnet masks for a single network. The administrator must use the same subnet mask for all subnets of the same network.

The OSPF *area* concept is modeled after an IP subnetted network. OSPF areas have been loosely defined to be a collection of networks. Instead, an OSPF area is specified to be a list of address ranges. Each address range is defined as an *address/mask* pair. Many separate networks may then be contained in a single address range, just as a subnetted network is composed of many separate subnets. Area border routers then summarize the area contents (for distribution to the backbone) by advertising a single route for each address range. The cost of the route is the minimum cost to any of the networks falling in the specified range.

Intraarea Routing

When a source and destination reside within the same area, routing is said to be *intraarea*. The router sends *Hello* packets to its neighbors, and in turn receives their Hello packets.

The router will attempt to form *adjacencies* with some of its newly acquired neighbors. Topological databases are synchronized between pairs of *adjacent routers*. On multiaccess networks, the *designated router* determines which routers should become adjacent.

Adjacencies control the distribution of routing protocol packets. Routing protocol packets are sent and received only on adjacencies. In particular, distribution of topological database updates proceeds along adjacencies.

Interarea Routing

When a packet's source and destination reside in different areas, routing is *interarea*. For interarea routing, *area border routers* use the same basic routing strategy as with intraarea routing, but also inject additional routing information into an area. This additional information is a distillation of the rest of the AS's topology.

External Routing

Routers that have information regarding other ASes can flood this information throughout an AS. This external routing information is distributed verbatim to every participating router. There is one exception: external routing information is not flooded into *stub areas* (described in the next section).

To use external routing information, the path to all routers advertising external information must be known throughout the AS (not including stub areas). For that reason, the locations of AS boundary routers are summarized by nonstub area border routers.

OSPF Support of Stub Areas

OSPF allows certain areas to be configured as *stub areas*. A stub area can be defined as any area that does not allow the advertisement of external routes. In other words, external advertisements are not flooded into or throughout stub areas; routing to AS external destinations in these areas is based on a per-area default only. This reduces the topological database size and the memory requirements associated with a stub area's internal routers.

In order to take advantage of the OSPF stub area support, *default routing* must be used in the stub area.

Neighbors and Adjacency

OSPF creates adjacencies between *neighboring routers* to facilitate exchange of routing information. *Neighboring routers* are two routers that have interfaces to a common network. On multiaccess networks, neighbors are dynamically discovered by OSPF's Hello Protocol. An *adjacency* is a relationship formed between selected neighboring routers for the purpose of exchanging routing information.

Not every pair of neighboring routers becomes adjacent. Instead, adjacencies are established with some subset of the router's neighbors. Routers connected by point-to-point networks and virtual links always become adjacent. On multiaccess networks, all routers become adjacent to both the designated router and the backup designated router (defined later in this section).

The Hello Protocol

The Hello Protocol is responsible for establishing and maintaining neighbor relationships. It also ensures that communication between neighbors is bidirectional. Hello packets are sent periodically out all router interfaces. Bidirectional communication is indicated when the router sees itself listed in the neighbor's Hello Packet.

On multiaccess networks, the Hello Protocol elects a designated router for the network. Among other things, the designated router controls what adjacencies will be formed over the network (see below).

Designated Routers

Every multiaccess network has a *designated router.* The designated router performs two main functions for the routing protocol:

- The designated router originates a *network links advertisement* on behalf of the network. This advertisement lists the set of routers, including the designated router, currently attached to the network.
- The designated router becomes adjacent to all other routers on the network. Since the link-state databases are synchronized across adjacencies (through adjacency initialization and the flooding procedure), the designated router plays a central part in the synchronization process.

Virtual Links

The backbone area (area id = 0) *cannot* be disconnected from any area, or some areas of the AS will become unreachable. To establish or maintain connectivity of the backbone, *virtual links* can be configured through nonbackbone areas. Virtual links connect separate components of the backbone (for example, two areas of the same AS). The two endpoints of a virtual link are *area border routers*. The virtual link must be configured in both routers. The configuration information in each router consists of the other virtual endpoint (the other area border router), and the nonbackbone area the two routers have in common (called the *transit area*).

Note: Virtual links cannot be configured through stub areas.

Cisco's OSPF Implementation

The sections that follow detail the specific router and interface subcommands used to enable and configure the OSPF IP routing protocol on Cisco routers. Cisco's implementation conforms to the OSPF Version 2 specifications detailed in Internet RFC 1247. The following list outlines key features supported in Cisco's OSPF implementation:

- **Stub areas**—Definition of stub areas is supported using the **stub** and **default-cost** options of the **area** router subcommand.
- **Route redistribution**—Routes learned via any IP routing protocol may be redistributed into any other IP routing protocol. At the intradomain level, this means that OSPF can import routes learned via IGRP, RIP and Hello. OSPF routes can also be exported into IGRP, RIP and Hello. At the interdomain level, OSPF can import routes learned via EGP and BGP. OSPF routes can be exported into EGP and BGP. Redistribution is enabled with the **redistribute** and **default-information** router subcommands
- **Authentication**—Authentication among neighboring routers within an area is supported using two commands. An **area** router subcommand enables authentication, while the **ip ospf authentication-key** interface subcommand specifies the password used in a specific area or on a specific interface by OSPF routers.
- **Router interface parameters**—Router interface parameters are configured using interface subcommands. Configurable parameters supported include interface output cost, retransmission interval, interface transmit delay, router priority, Hello Protocol interval, router "dead" interval, and authentication key. Each of these is configured using various options of the **ip ospf** interface subcommand.
- **Nonbroadcast networks**—OSPF is supported over nonbroadcast networks by Cisco routers using the **neighbor** router subcommand. The **neighbor** subcommand allows specification of three parameters:
  - The priority for a neighboring router
  - The nonbroadcast poll interval
  - The interface through which the neighbor is reachable

- **Virtual links**—Virtual links are created using the **virtual-link** option of the **area** router subcommand.

Steps in Configuring OSPF Routing

Configuration of an OSPF routing process for a Cisco router involves the following general steps. Step 1 is mandatory; the other steps are optional, but may be required for your specific application.

Step 1: Enable OSPF using the **router ospf** global command and **network** router subcommand.

Step 2: Specify route redistribution, as needed (addressed separately with redistribution discussion for all IP routing protocols).

Step 3: Set any OSPF area parameters, as needed.

Step 4: Set interface parameters, as needed.

Step 5: For nonbroadcast networks, specify appropriate neighbor parameters and the router's polling interval, as required.

The commands for each of these steps are detailed in the sections that follow.

Enabling the OSPF Routing Processes and Defining Areas

To start an OSPF routing process, you must enable OSPF in the router, specify the range of IP addresses to be associated with the routing process, and assign area ids to be associated with that range of IP addresses. Two commands are associated with this initial step: **router ospf** (a global configuration command) and **network** (a router subcommand).

Enabling OSPF Routing

The first step in setting up OSPF support on a router is to enable OSPF using the **router ospf** global configuration command. The syntax for this command is:

router ospf *ospf-process-id*
no router ospf *ospf-process-id*

The argument *ospf-process-id* is an internally used identification parameter. It is locally assigned and can be any positive integer. A unique value is assigned for each OSPF routing process. You can specify multiple OSPF routing processes in each router.

The **no router ospf** command must be specified with the *ospf-process-id* and terminates individual OSPF routing processes in the router.

Defining OSPF on Networks and Assigning Area IDs

Use the **network** router subcommand to define the interfaces on which OSPF run and the define the area id for those interfaces. The general syntax for this command is:

network *address wildcard-mask* **area** *area-id*
no network *address wildcard-mask* **area** *area-id*

The **network** router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

The *address* and *wildcard-mask* arguments together allow you to define one or multiple interfaces to be associated with a specific OSPF area using a single command. The argument *address* is formed as an IP address. The argument *wildcard-mask* is an IP-address-type mask that includes "don't care" bits.

The keyword/variable argument pair **area** *area-id* specifies an area to be associated with the OSPF address range as defined in the same **network** command. The argument *area-id* can be specified as either a decimal value or as an IP address. If you intend to associate areas with IP subnets, you can specify a subnet address as the *area-id*.

Note: Any individual interface can only be attached to a single area. If the address ranges specified for different areas overlap, the router will adopt the first area in the **network** subcommand list and ignore the subsequent overlapping portions. In general, Cisco recommends devising address ranges that do not overlap in order to avoid inadvertent conflicts.

The **no network** router subcommand must be specified with the complete address range and area id; it disables OSPF routing for interfaces defined with the *address wildcard-mask* pair.

Ecmaple:

The **router ospf** and **network** commands defined in this section control the assignment of area ids to specific address ranges. The following example illustrates the assignment of four area ids to four IP address ranges.

In the following example, OSPF routing process 109 is initialized, and four OSPF areas are defined: 10.9.50.0, 2, 3, and 0. Areas 10.9.50.0, 2, and 3 mask specific address ranges, while Area 0 enables OSPF for *all other* networks.

```
router ospf 109
network 131.108.20.0  0.0.0.255 area 10.9.50.0
network 131.108.0.0  0.0.255.255 area 2
network 131.109.10.0  0.0.0.255 area 3
network 0.0.0.0  255.255.255.255 area 0
!
! Interface Ethernet0 is in area 10.9.50.0:
interface Ethernet 0
ip address 131.108.20.5 255.255.255.0
!
! Interface Ethernet1 is in area 2:
interface Ethernet 1
ip address 131.108.1.5 255.255.255.0
!
! Interface Ethernet2 is in area 2:
interface Ethernet 2
```

```
ip address 131.108.2.5 255.255.255.0
!
! Interface Ethernet3 is in area 3:
interface Ethernet 3
ip address 131.109.10.5 255.255.255.0
!
! Interface Ethernet4 is in area 0:
interface Ethernet 4
ip address 131.109.1.1 255.255.255.0
!
! Interface Ethernet5 is in area 0:
interface Ethernet 5
ip address 10.1.0.1 255.255.0.0
```

Each **network** subcommand is evaluated sequentially, so the specific order of these commands in the configuration is important. The router sequentially evaluates the *address*/*wildcard-mask* pair for each interface as follows:

Step 1: The *wildcard-mask* is logically ORed with the interface IP address.

Step 2: The *wildcard-mask* is logically ORed with *address* in **network** command.

Step 3: The router compares the two resulting values.

Step 4: If they match, OSPF is enabled on the associated interface and this interface is attached to the OSPF area specified.

Consider the first **network** subcommand. area id 10.9.50.0 is configured for the interface on which subnet 131.108.20.0 is located. Assume that a match is determined for interface Ethernet 0. Interface Ethernet 0 is attached to Area 10.9.50.0 only.

The second **network** subcommand is evaluated next. For Area 2, the same process is then applied to all interfaces (except interface Ethernet 0). Assume that a match is determined for interface Ethernet 1. OSPF is then enabled for that interface and Ethernet 1 is attached to Area 2.

This process of attaching interfaces to OSPF areas continues for all network subcommands. Note that the last **network** subcommand in this example is a special case. With this command all available interfaces (not explicitly attached to another area) are attached to Area 0.

When you configure secondary addresses with OSPF, make certain that the primary and secondary addresses of an interface fall into the same area. If they are in different areas, secondary addresses will be ignored. The following example illustrates the correct way to configure secondary addresses.

```
interface ethernet 0
ip address 131.108.10.10 255.255.255.0
ip address 10.0.0.10 255.0.0.0 secondary

router ospf 109
network 131.108.0.0 0.0.255.255 area 0
network 10.0.0.0. 0.255.255.255 area 0
```

Configuring OSPF Area Parameters

The **area** router subcommand allows control of various area parameters as defined by RFC 1247. This router subcommand also allows you to control certain special capabilities. (such as stub areas and virtual links). The discussions that follow outline the various options for the **area** subcommand and summarize their applications.

Note: The specification of the **no area** *area-id* router subcommand (with no other keywords or arguments) removes the specified area from the router's configuration. The *area-id* argument is defined in more detail in the following **area** router subcommand descriptions.

Setting Simple OSPF Area Authentication

In general, authentication is configured on a per-area basis. It is enabled for an area using the **authentication** option of the **area** router subcommand. The syntax for this command is:

area *area-id* **authentication**
no area *area-id* **authentication**

The argument *area-id* is the specific area id of the area for which authentication is to be enabled. The argument *area-id* can be specified as either a decimal value or as an IP address. Specifying authentication for an area sets the authentication to Type 1 (simple password). If this command is not included in the configuration for a router, authentication is of Type 0 (no authentication).

The authentication type must be the same for all routers in an area. The authentication *key* (password) for all OSPF routers on a network must be the same if they are to communicate with each other via OSPF. Use the **ip ospf authentication-key** interface subcommand to specify this password.

The **no area** *area-id* **authentication** option removes the area's authentication specification.

Defining a Stub Area

There are two stub area router subcommands: the **stub** and **default-cost** options of the **area** router subcommand. In all routers attached to the stub area, the area should be configured as a stub area using the **stub** option of the **area** command. Use the **default-cost** only on an area border router attached to the stub area. This command provides the metric for the summary default route generated by the area border router into the stub area. The syntax for the **area** *area-id* **stub** router subcommand is as follows:

area *area-id* **stub**
no area *area-id* **stub**

The argument *area-id* is the specific area id for the stub area. The argument *area-id* can be specified as either a decimal value or as an IP address.

The **stub** option is used to enable the stub area.

The **no area** *area-id* **stub** option removes the specified area as a stub area.

The syntax for the **area** *area-id* **default-cost** *cost* router subcommand is as follows:

> **area** *area-id* **default-cost** *cost*
> **no area** *area-id* **default-cost** *cost*

The argument *area-id* is the specific area id for the stub area. The argument *area-id* can be specified as either a decimal value or as an IP address.

The **default-cost** *cost* keyword/argument pair assigns a specific cost for the default summary route used for the stub area. The accepted value is a 24-bit number.

The **no area** *area-id* **default-cost** *cost* removes the assigned default route cost.

Consolidating Advertised Addresses

Routing information is condensed at area boundaries. External to the area, a single route is advertised for each address range. To consolidate or summarize routes, you can use the **range** option of the **area** router subcommand. The result is that a single summary route is advertized to other areas by the area border router. This command is only used with area border routers. The syntax for this router subcommand is as follows:

> **area** *area-id* **range** *address mask*
> **no area** *area-id* **range** *address mask*

The argument *area-id* is the specific area id for the area about which routes are to be summarized. The argument *area-id* can be specified as either a decimal value or as an IP address.

Note: Multiple **area** router subcommands specifying the **range** option can be specified. Thus, OSPF can summarize addresses for many different sets of address ranges.

The *address* argument is a standard IP address. The *mask* argument is a standard IP mask.

Configuring OSPF Interface-Specific Parameters

The interface subcommands described in this section define how you can configure each OSPF router interface parameter. In general, you do not need to configure any of these parameters. However, some interface parameters must be consistent across all routers in an attached network; be sure that if you do configure any of these parameters, that the configurations for all routers on the network have compatible values.

Specifying OSPF Path Cost

To explicitly specify the cost of sending a packet on an interface, use the **ip ospf cost** interface subcommand. The syntax for this command is:

> **ip ospf cost** *cost*
> **no ip ospf cost** *cost*

The argument *cost* is expressed as the link state metric. It is a dimensionless integer value that is always greater than zero. This value is advertized as the link cost in the router's router links advertisement. Cisco does not support Type of Service (TOS), so you can assign only one cost per interface.

Using the following formula, the default path costs were calculated as noted in the list that follows the formula. If these values do not suit your network, you can use your own method of calculating path costs.

- 56 Kbps Serial Link—Default cost is 1785
- 64 Kbps Serial Link—Default cost is 1562
- T1 (1.544 Mbps Serial Link)—Default cost is 65
- E1 (2.048 Mbps Serial Link)—Default cost is 48
- 4 Mbps Token Ring—Default cost is 25
- Ethernet—Default cost is 10
- 16 Mbps Token Ring—Default cost is 6
- FDDI—Default cost is 1

In general, the path cost is calculated as follows:

$$\frac{10^8}{Bandwidth}$$

The **no ip ospf cost** interface subcommand resets the path cost for an interface to the default value.

Setting the Link State Retransmission Interval

The number of seconds between link state advertisement retransmissions, for adjacencies belonging to the interface, is specified with the **ip ospf retransmit-interval** interface subcommand. The syntax for this command is:

> **ip ospf retransmit-interval** *number-of-seconds*
> **no ip ospf retransmit-interval** *number-of-seconds*

The value for the *number-of-seconds* argument is an integer number, that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links. The retransmit interval is the number of seconds a router should wait before retransmitting a link state advertisement (LSA) in the absence of an acknowledgment. If a change in the link occurs, an LSA is immediately transmitted.

The default value is five seconds.

The **no ip ospf retransmit-interval** subcommand resets the link state advertisement retransmission interval to the default value.

Setting the Transmission Time for Link State Updates

To set the estimated number of seconds it takes to transmit a link state update packet on the interface, use the **ip ospf transmit-delay** interface subcommand. The syntax for this command is:

> **ip ospf transmit-delay** *number-of-seconds*
> **no ip ospf transmit-delay** *number-of-seconds*

Link state advertisements in the update packet must have their age incremented by this amount before transmission. The value assigned should take into account the transmission and propagation delays for the interface.

The argument *number-of-seconds* is an integer value that must be greater than zero. The default value is 10 seconds.

If the delay is not added before transmission over a link, the time in which the LSA propagates over the link is not considered. This setting has more significance on very low-speed links.

The **no ip ospf transmit-delay** subcommand resets the estimated transmission time for the link state update packet the default value.

Setting Router Priority

The router priority is used to help determine the designated router for a network. To set the router priority, use the **ip ospf priority** interface subcommand. The syntax for this command is:

> **ip ospf priority** *8-bit-number*
> **no ip ospf priority** *8-bit-number*

The argument *8-bit-number* is an 8-bit unsigned integer.

When two routers attached to a network both attempt to become the designated router, the one with the highest router priority takes precedence. If there is a tie, the router with the highest router id takes precedence. A router with a router priority set to zero is ineligible to become the designated router or backup designated router. Router priority is only configured for interfaces to multiaccess networks (in other words, not point-to-point networks).

The default router priority is 1.

The **no ip ospf priority** subcommand resets the router priority to the default value.

Setting the Advertised Hello Interval

Use the **ip ospf hello-interval** interface subcommand to specify the length of time, in seconds, between the Hello packets that the router sends on the interface. The syntax for this command is:

> **ip ospf hello-interval** *number-of-seconds*
> **no ip ospf hello-interval** *number-of-seconds*

The argument *number-of-seconds* is an unsigned integer value. This value is advertised in the router's Hello packets. It must be the same for all routers attached to a common network. The smaller the Hello interval, the faster topological changes will be detected, but more routing traffic will ensue.

The default differs for broadcast and nonbroadcast networks. The default for broadcast networks is 10 seconds. The default for nonbroadcast networks (for example, X.25, Frame Relay, and SMDS) is 30 seconds (determined automatically when any of the serial encapsulations are specified).

Note: The **hello-interval** specification must be the same for all nodes on a specific network.

The **no ip ospf hello-interval** subcommand resets the length of time between Hello packet transmissions on an interface to the default value.

Setting the Router Dead Interval

Use the **ip ospf dead-interval** interface subcommand to set the number of seconds that a router's Hello packets have not been seen before its neighbors declare the router down. The command syntax is:

> **ip ospf dead-interval** *number-of-seconds*
> **no ip ospf dead-interval** *number-of-seconds*

The argument *number of-seconds* is an unsigned integer value.

The default is four times the **ip ospf hello-interval** value. As with the Hello interval, this value must be the same for all routers attached to a common network.

The **no ip ospf dead-interval** subcommand resets the length of time that a router's Hello packets have not been seen before its neighbors declare the router down to the default value.

The *number-of-seconds* specified must be the same for all nodes on a network.

Specifying the OSPF Authentication Key

Use the **ip ospf authentication-key** interface subcommand to assign a specific password to be used by neighboring routers on a wire that are using OSPF's simple password authentication. The command syntax is:

> **ip ospf authentication-key** *8-bytes-of-password*
> **no ip ospf authentication-key** *8-bytes-of-password*

The argument *8-bytes-of-password* is any continuous string of characters that you can enter from the keyboard up to eight bytes in length.

Note: A router will use this key only when authentication is enabled for an area with the **area** *area-id* **authentication** router subcommand.

This key is inserted directly into the OSPF header when originating routing protocol packets. A separate password can be assigned to each network on a per-interface basis. All neighboring routers on the same network must have the same password to be able to route OSPF traffic.

The default is null. The **no ospf authentication-key** interface subcommand removes any OSPF password that was previously assigned.

Configuring OSPF for Nonbroadcast Networks

OSPF treats a nonbroadcast, multiaccess network (X.25, Frame Relay, or SMDS) much like it treats a broadcast network. Since there may be many routers attached to the network, a designated router is *selected* for the network. This designated router then originates a networks links advertisement, which lists all routers attached to the nonbroadcast network.

However, due to the lack of broadcast capabilities, it is necessary to use special configuration parameters in the designated router selection. These parameters need only be configured in those routers that are themselves eligible to become the designated router or backup designated router (in other words, routers with a positive, nonzero router priority value).

Use the **neighbor** router subcommand to configure routers interconnecting to nonbroadcast networks. The syntax for this command is:

> **neighbor** *address* **interface** *type unit-number* [**priority** *8-bit-number*]
> [**poll-interval** *number-of-seconds*]
>
> **no neighbor** *address* **interface** *type unit-number* [**priority** *8-bit-number*]
> [**poll-interval** *number-of-seconds*]

One neighbor entry must be included in the router's configuration for each known nonbroadcast network neighbor.

The argument *address* is the specific interface IP address of the neighbor.

The keyword/argument sequence **interface** *type unit-number* identifies the specific router interface for this **neighbor** command. The interface must be connected to a nonbroadcast, multiaccess type network. In addition, the neighbor specified must be eligible to be a designated router or backup designated router.

The keyword/argument pair **priority** *8-bit number* is the router priority value of the nonbroadcast neighbor associated with the IP address specified.

If a neighboring router has become inactive (Hello packets have not been seen for Router DeadInterval period), it may still be necessary to send Hello packets to the dead neighbor. These Hello packets will be sent at a reduced rate called the *Poll Interval*.

Specify the poll interval with the keyword/argument pair **poll-interval** *number-of-seconds*. The argument *number-of-seconds* is an unsigned integer value. RFC 1247 recommends that this value should be much larger than the Hello interval. The default is 2 minutes (120 seconds).

The **no neighbor** router subcommand requires specification of the neighbor's IP address; this command removes the specific neighbor from the list.

Creating Virtual Links

In OSPF, all areas must be connected to a backbone area. If the connection to the backbone is lost, it can be repaired by establishing a *virtual link*. With Cisco routers, virtual links are defined using the **virtual link** option of the **area** router subcommand. The general syntax for this subcommand is as follows:

area *area-id* **virtual-link** *router-id* [**hello-interval** *number-of-seconds*]
[**retransmit-interval** *number-of-seconds*]
[**transmit-delay** *number-of-seconds*]
[**dead-interval** *number-of-seconds*]
[**authentication-key** *8-bytes-of-password*]

no area *area-id* **virtual-link** *router-id* [**hello-interval** *number-of-seconds*]
[**retransmit-interval** *number-of-seconds*]
[**transmit-delay** *number-of-seconds*]
[**dead-interval** *number-of-seconds*]
[**authentication-key** *8-bytes-of-password*]

The argument *area-id* is the area id assigned to the transit area for the virtual link.This can be either a decimal value or a valid IP address.

The argument *router-id* is the router id associated with the virtual link neighbor. The router id appears in the **show ip ospf** display. It is internally derived by each router from the router's interface IP addresses. This value must be entered in the format of an IP address.

The default is null.

Note: Each virtual link neighbor must include the transit area id and the corresponding virtual link neighbor's router id in order for a virtual link to be properly configured.

Specify the length of time, in seconds, between the Hello packets that the router sends on the interface with the **hello-interval** option of the **area** *area-id* **virtual-link** router subcommand. The argument *number-of-seconds* is an unsigned integer value. This value is advertised in the router's Hello packets. It must be the same for all routers attached to a common network. The smaller the Hello interval, the faster topological changes will be detected, but more routing traffic will ensue. The default is ten seconds.

Specify the number of seconds between link state advertisement retransmissions for adjacencies belonging to the interface, with the **retransmit-interval** option of the **area** *area-id* **virtual-link** router subcommand. The value for the *number-of-seconds* argument is an integer that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links. The default value is five seconds.

Specify the estimated number of seconds it takes to transmit a link state update packet on the interface with the **transmit-delay** option of the **area** *area-id* **virtual-link** router subcommand. Link state advertisements in the update packet must have their age incremented by this amount before transmission. The value assigned should take into account the transmission and propagation delays for the interface. The argument *number-of-seconds* is an integer value that must be greater than zero. The default value is one second.

Set the number of seconds that a router's Hello packets have not been seen before its neighbors declare the router down with the **dead-interval** option of the **area** *area-id* **virtual-link** router subcommand. The argument *number-of-seconds* is an unsigned integer value. The default is four times the Hello interval. As with the Hello interval, this value must be the same for all routers attached to a common network.

Assign a specific password to be used by neighboring routers with the **authentication-key** option of the **area** *area-id* **virtual-link** router subcommand. The argument *8-bytes-of-password* is any continuous string of characters that you can enter from the keyboard up to eight bytes in length. This configured data allows the authentication procedure to generate and/or verify the authentication field in the OSPF header. This key is inserted directly into the OSPF header when originating routing protocol packets. A separate password can be assigned to each network on a per-interface basis. All neighboring routers on the same network must have the same password to be able to route OSPF traffic. There is no default value.

Note: A router will use this authentication key only when authentication is enabled for the backbone with the **area 0 authentication** router subcommand.

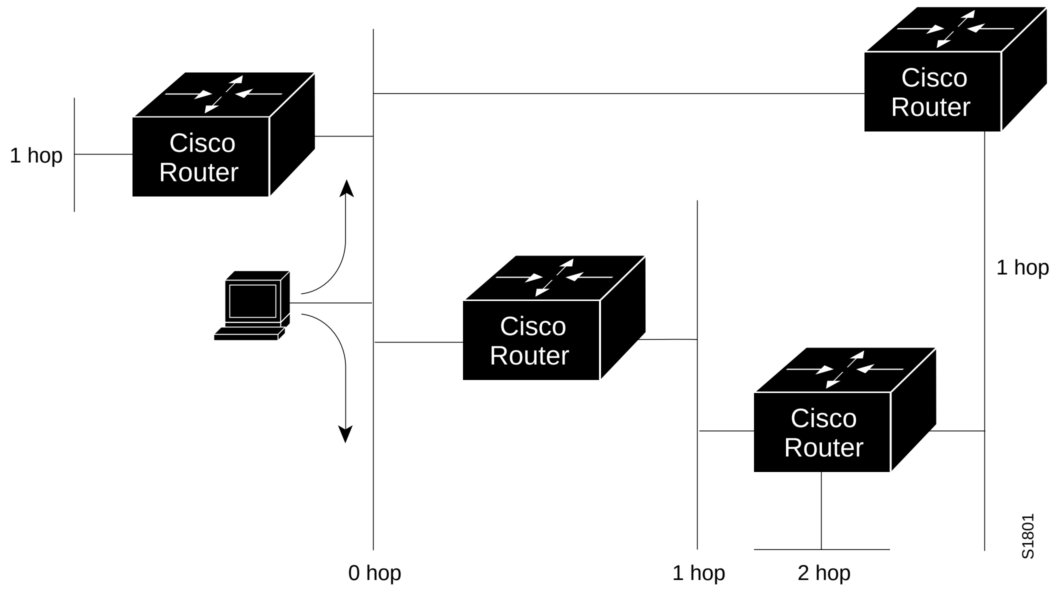
The **no area** *area-id* **virtual-link** *router-id* command removes a virtual link.

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 1-5); an unreachable network has a metric of 16. This small range of metrics makes RIP unsuitable as a routing protocol for large 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 1-5 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 (of directly connected networks). 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.

The **network** router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

Example:

The following example configuration defines RIP as the routing protocol to be used on all interfaces connected to networks 128.99.0.0 and 192.31.7.0.

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

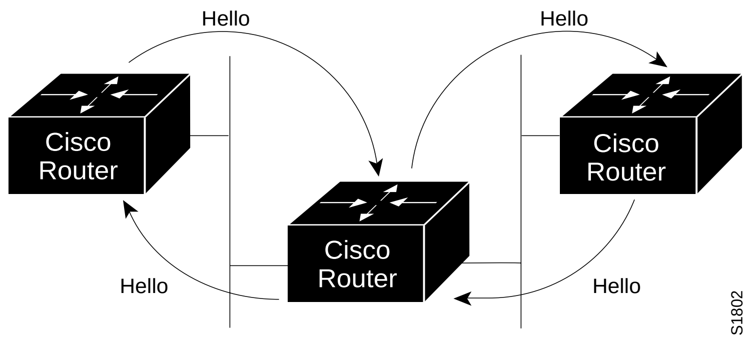
To remove a network from the list, use the **no network** router subcommand followed by the network address.

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 time-keeping and delay measurement features. Specifically, the router sets the invalid bit in the Hello date field and clears the time and timestamp fields.

Figure 1-6 The Hello Protocol



The metric used in Hello is a delay value measured in milliseconds (see Figure 1-6). 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

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 Hello Networks

After you have created the routing process, specify the list of networks. This list is specified with the **network** router configuration subcommand:

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

The **network** router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

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

Example:

In the following example, the network 160.1.0.0 is being set up for Hello.

```
router hello
network 160.1.0.0
```

Configuring the BGP Protocol

BGP, as defined in RFC 1267, 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 BGP Networks

Use the **network** router subcommand to specify 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*
no 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. The **no** version removes the specified *network-number.*

The **network** router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process. The maximum number of network subcommands allowed under a router BGP process is 200.

Note: For exterior protocols, a reference to an IP network from the **network** command that is learned by another routing protocol does not require a **redistribute** command. This is in contrast to interior gateway protocols, such as IGRP, which require the use of the **redistribute** command.

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 want to specify that another router is a neighbor. Use the **neighbor** command, as shown below:

neighbor *address* **remote-as** *number*
no neighbor *address*

The argument *address* is the IP address of the neighbor. The argument *number* is the AS to which the neighbor belongs. Specifying a neighbor with a *number* that matches the AS number specified in the **router bgp** command identifies the neighbor as internal to the same AS to which the router belongs.

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 131.108.1.2 is a neighbor in AS number 109.

```
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 three 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 neighbor routers. The first router listed is in the same Class B network-address space, but in a different AS; the second **neighbor** command illustrates specification of an internal neighbor (with the same AS number) at address 131.108.234.2; and the last **neighbor** command specifies a neighbor on a different network.

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  remote-as 167
neighbor 131.108.234.2 remote-as 109
neighbor 150.136.64.19  remote-as  99
```

Setting Route 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. The argument *weight* is the weight to assign. All routes learned from this neighbor will have this initial weight. The route with the highest weight will be chosen as the preferred route when multiple routes are available to a particular network. Negative values for the *weight* argument are not allowed.

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. 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. Only standard access lists can be used with this command.

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
```

Filtering BGP Routes

You can specify an access list filter on both incoming and outbound BGP routes. In addition, you can assign *weights* based on a set of filters. Each filter is an access list based on regular expressions. Use the **ip as-path access-list** global configuration command to define an BGP access list, and the **neighbor** router subcommand to apply a specific access list.

Defining a BGP Access List

To define a BGP-related access list, use the **ip as-path access-list** global configuration command. The command syntax is:

ip as-path access-list *list* [**permit**|**deny**] *as-regular-expression*
no ip as-path access-list *list* [**permit**|**deny**] *as-regular-expression*

The *list* argument is an integer from 1 to 99.

Regular expressions are defined in the appendix "Pattern Matching." If the regular expression matches the representation of the autonomous system path of the route as an ASCII string, then the **permit** or **deny** condition applies.

Specifying BGP Route Filters

Filters are established with the **neighbor** router subcommand, using access lists defined with the **ip as-path access-list** command. Several variations are allowed. The command syntax is:

neighbor *address* **filter-list** *list* {**in**|**out**|**weight** *weight*}
no neighbor *address* **filter-list** *list* {**in**|**out**|**weight** *weight*}

The argument *address* is the address of the neighbor.

The argument *list* is a predefined BGP access list number.

One of three alternative keywords must also be used:

- The keyword **in** specifies that you are applying the access list to incoming routes.
- The keyword **out** specifies that you are applying the access list to outgoing routes.
- The keyword/argument pair **weight** *weight* assigns a relative importance to a specific list. The given *weight* is added to the weight of the route if the AS path matches the regular expression. Any number of weight filters are allowed on a per neighbor basis, but only one in or out filter is allowed. The weight of a route affects BGP's route-selection rules. Acceptable values are 0 to 65535. The default is zero (0).

Example:

In the following example, the BGP-neighbor with IP address 128.125.1.1 is not sent advertisements about any path through or from the adjacent AS 123.

```
ip as-path access-list 1 deny ^123$
ip as-path access-list 1 deny ^123 .*
! The space in the above expression (^123.*)is required.

router bgp 109
network 131.108.0.0
neighbor 129.140.6.6 remote-as 123
neighbor 128.125.1.1 remote-as 47
neighbor 128.125.1.1 filter-list 1 out
```

Specifying BGP Version Number

Cisco's implementation of BGP now supports Versions 2 and 3 of the protocol and permits dynamic version negotiation with neighbors. Cisco routers can be configured to handle only Version 2 of the protocol using the **neighbor version** router subcommand. The command syntax is:

neighbor *ip-address* **version** *value*
no neighbor *ip-address* **version** *value*

The argument *ip-address* is the address of the BGP-speaking neighbor; the version *value* can be set to 2 to force the router to only use Version 2 with the specified neighbor. The default is to use Version 3 of BGP and dynamically negotiate down to Version 2 if requested. The **no neighbor** *ip-address* **version** *value* command returns the version to the default state for that neighbor.

Specifying BGP Administrative Distance

Cisco's implementation of BGP allows the use of three possible administrative distances, assigned with the **distance bgp** router subcommand. The command syntax is:

distance bgp *external-distance internal-distance local-distance*
no distance bgp

Use this command if another protocol is known to be able to provide a better route to a node that was actually learned via external BGP, or if some local routes should really be preferred by BGP.

Note: Changing the administrative distance of BGP internal routes is considered dangerous and is generally not recommended. One problem that can arise is the accumulation of routing table inconsistencies which can break routing.

The argument *external-distance* specifies the value for BGP external routes. External routes are routes for which the best path is learned from a neighbor external to the autonomous system. Acceptable values are positive, nonzero integers.

The argument *internal-distance* specifies the value for BGP internal routes. Internal routes are those routes that are learned from another BGP entity within the same autonomous system. Acceptable values are positive, nonzero integers.

The argument *local-distance* specifies the value for BGP local routes. Local routes are those networks listed with a **network** command—possibly as back doors (discussed in the next section)—for that router or for networks that are being redistributed from another process.

By default, the administrative distances are as follows

- *external-distance*—20
- *internal-distance*—200
- *local-distance*—200

The **no distance bgp** command resets these values to the defaults.

Adjusting the BGP Timers

To adjust the default BGP timers, use the **timers bgp** 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
```

Clearing BGP Connections

Use the EXEC command **clear ip bgp** to reset BGP connections. The command syntax is:

> **clear ip bgp ***
> **clear ip bgp** *address*

This command resets the BGP connection with the specified BGP neighbor (identified with the *address* argument). If you specify an asterisk (*), all current BGP sessions are reset.

In general, use this command whenever a policy changes. Changes that might prompt you to use this command are as follows:

- Additions or changes to the BGP-related access lists
- Changes to BGP-related weights
- Changes to BGP-related distribution lists
- Changes in the BGP timers specifications

Controlling BGP to Determine Which Neighbors Are Peers

To control how a BGP process determines which neighbors will be treated as peers, use the **neighbor any** router subcommand with the **router bgp 0** global subcommand. The syntax for this command is as follows:

> **neighbor any** [*list*]
> **no neighbor any** [*list*]

If the *list* argument is specified, the neighbor *must* be accepted by the access list number specified to be allowed to peer with the BGP process.

Continuing with the preceding sample configuration, the configuration would look like the following:

```
router bgp 109
network 131.108.0.0
network 192.31.7.0
neighbor 131.108.200.1  remote-as 167
neighbor 131.108.234.2  remote-as 109
neighbor 150.136.64.19  remote-as  99
neighbor any 1
access-list 1 permit 192.31.7.0 0.0.0.255
```

Using BGP without IGP Redistribution

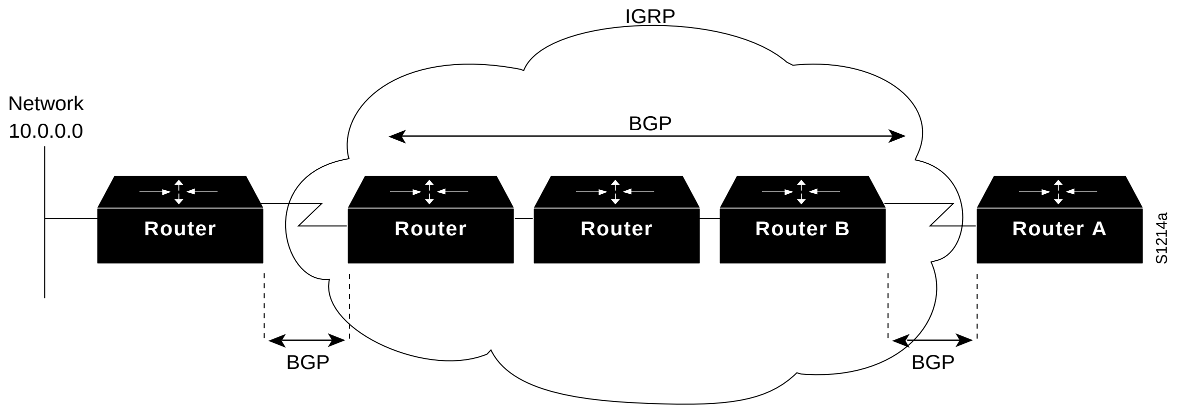
Use the **no synchronization** subcommand of the **router bgp** global command when you want to disable the synchronization between BGP and your IGP.

Usually, a BGP speaker does not advertise a route to an external neighbor unless that route is local or exists in the IGP. The **no synchronization** command will allow a router to advertise a network route without waiting for the IGP. This feature allows routers within an AS to have the route before BGP makes it available to other ASs. Use the **synchronization** command if there are routers in the AS that do not speak BGP. The default is synchronization.

> **no synchronization**
> **synchronization**

In Figure 1-7, with synchronization on, Router B will not advertise network 10.0.0.0 to Router A until an IGRP route for network 10.0.0.0 exists. If you specify the **no synchronization** command, Router B will advertise network 10.0.0.0 as soon as possible.

Figure 1-7 Synchronization Diagram



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 of the command without specifying a network number:

```
puck> show ip bgp
BGP table version is 22855, local router ID is 192.54.222.5
Status codes: * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network         Next Hop         Metric Weight Path
*> 128.128.0.0    131.192.115.3             0 ?
*> 192.132.70.0   192.54.222.9             20 702 701 ?
*> 152.155.0.0    131.192.77.1              0 ?
*> 192.74.137.0   192.54.222.9             20 702 701 i
*> 192.52.247.0   131.192.2.1               0 ?
*> 146.150.0.0    131.192.77.1              0 ?
*> 192.139.79.0   192.54.222.9             20 702 701 ?
*> 192.75.142.0   192.54.222.9             20 702 701 ?
*> 192.75.141.0   192.54.222.9             20 702 701 ?
*> 142.136.0.0    192.54.222.9             20 702 701 ?
*> 192.139.77.0   192.54.222.9             20 702 701 ?
*> 129.135.0.0    192.54.222.9             20 702 701 ?
*> 192.160.103.0 192.54.222.9             20 702 701 ?
*> 7.0.0.0        192.54.222.9             20 702 701 35 e
*> 139.140.0.0    131.192.34.2              0 ?
*> 8.0.0.0       131.192.2.1                 0 ?
*> 192.58.242.0  131.192.2.1                 0 ?
```

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 (Status codes) indicate the status. The asterisk (*) indicates that the table entry is valid. The > character indicates that the 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 non-BGP route to this network.
- The Metric field, if any, is the value of the interautonomous system metric. This is frequently not used.
- The Weight field is set through the use of AS filters.
- The Path field 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. A question mark (?) indicates that the origin of the path is not clear. Usually this a path that is redistributed into BGP from an IGP.

Following is sample output of the command when used with a network number:

```
puck> show ip bgp 7.0.0.0
BGP routing table entry for 7.0.0.0, version 20760
Paths: (1 available, best #0)
  702 701 35
    192.54.222.9 from 192.54.222.9, metric 0, weight 20
      Origin EGP, valid, external, best
```

In the display:

- The first line indicates the network that the route is for and the version number of the table the last time the route changed.
- Paths indicates the number of paths, and the index to the best path.
- The third line is the AS path associated with the route.
- 192.54.222.9 from 192.54.222.9 indicates the next hop for the route and the router that it is learned from.
- The metric is the metric assigned to the route.
- The weight is the administrative weight assigned to the route.
- Origin EGP is the origin code for the route.
- Valid indicates whether or not the route is valid (usable).
- External indicates whether or not the route was learned externally or internally.
- Best indicates if the route is the best route or not.

Displaying Routes Learned from a Neighbor

Use the **show ip bgp neighbors** *network* routes command to show the routes learned from that particular neighbor. Enter this command at the EXEC prompt:

show ip bgp neighbors *network* **routes**

The optional argument *network* is the network number for the neighbor whose routes you have learned from.

The display is the same as the display for the **show ip bgp** command.

Displaying BGP Paths

Use the **show ip bgp paths** command to display all the BGP paths in the database. Enter this command at the EXEC prompt:

show ip bgp paths

Following is sample output:

```
Address    Hash Refcount Metric Path
0x297A9C      0        2      0 i
0x30BF84      1        0      0 702 701 ?
0x2F7BC8      2      235      0 ?
0x2FA1D8      3        0      0 702 701 i
```

In the display:

Address—Internal address where the path is stored

Hash—Hash bucket where path is stored

Refcount—Number of routes using that path

Metric—INTER_AS metric for the path

Path—AS_PATH for that route, followed by the origin code for that route

Displaying BGP Summaries

Use the **show ip bgp summary** command to display the status of all BGP connections. Enter the command that follows at the EXEC prompt.

show ip bgp summary

Following is sample output:

```
BGP table version is 3937, main routing table version 3937

Neighbor        V    AS MsgRcvd MsgSent TblVer InQ OutQ Uptime/State
192.54.222.6    2   690    7655     268   3937   0    0 2:39:51
192.54.222.9    3   702     682     364   3937   0    0 2:39:54
```

In the display:

- BGP table version—Internal version number of BGP database
- routing table version—Last version of BGP database that was injected into main routing table
- Neighbor—IP address of a neighbor
- V—BGP version number spoken to that neighbor
- AS—Autonomous system number of that neighbor
- MsgRcvd—BGP messages received from that neighbor
- MsgSent—BGP messages sent to that neighbor
- TblVer—Last version of the BGP database that was sent to that neighbor
- InQ—Number of messages from that neighbor waiting to be processed
- OutQ—Number of messages waiting to be sent to that neighbor
- Update/State—Length of time that the BGP session has been in state Established, or the current state if it is not Established

Debugging IP-BGP Updates

Use the following EXEC commands to debug BGP. For each **debug** command, there is a corresponding **undebug** command that runs the messages off.

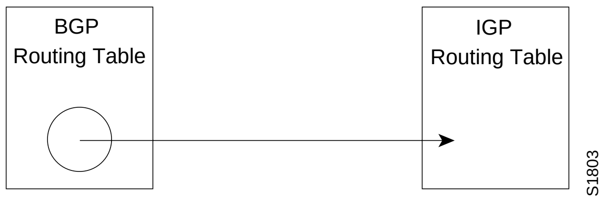
debug ip-bgp-updates

The **debug ip-bgp-updates** command generates per-update messages.

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 ASes that a datagram must transit to reach a particular network. Information from the BGP routing table is entered into the main IP routing table (see Figure 1-8). In most cases, the BGP information should override IGP information.

Figure 1-8 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.

It is possible to indicate which networks are reachable using a *back-door* route that the border router should use. A back door network is treated as a local network, except that it is not advertised.

Use this variation of the **network** router subcommand to specify a backdoor route:

network *address* **backdoor**

The **network** router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

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 to ensure that routing information does not loop. A configuration such as this is fairly risky, requiring careful attention to filtering. Filtering is described in more detail earlier in this chapter and in subsequent discussions.

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 the path with lowest metric.
- Final determinant is the peer with the largest value for the IP address.

- If IGP synchronization is enabled, then a path learned via BGP must be synchronized with IGP before it can be considered for selection.

You can also create routing loops if a BGP speaker injects a route into the IGP and then forwards packets back into the AS. For this reason, routes learned via internal BGP cannot be redistributed into other routing protocols.

BGP Path Attributes

The Cisco BGP implementation supports all path attributes defined in RFC 1163 and 1267. 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.

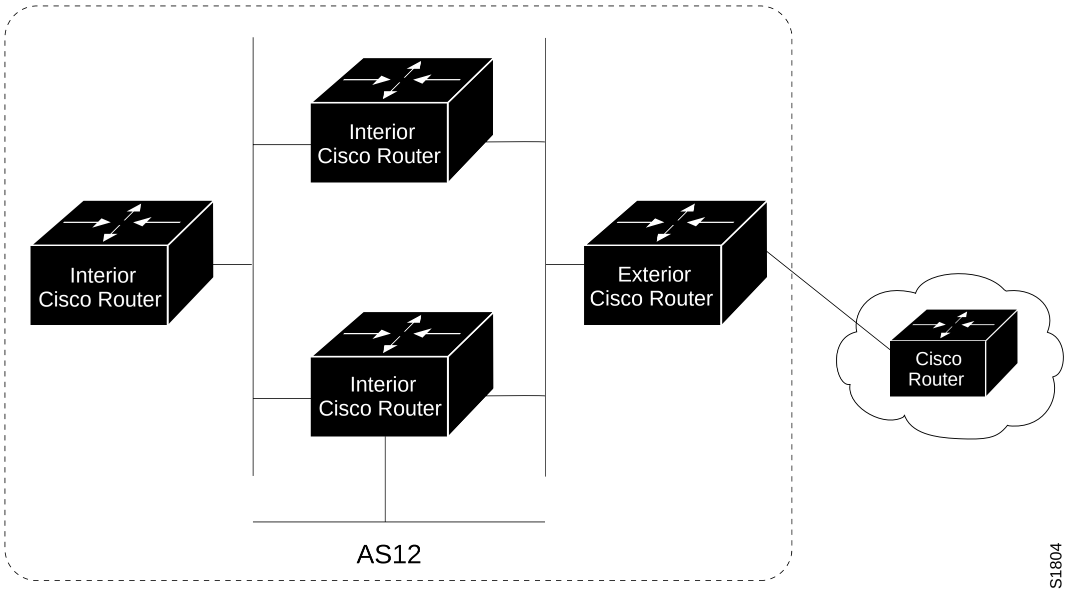
The Cisco implementation will use a third-party next hop router address in NEXT_HOP attribute regardless of the AS of that third-party router.

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 1-9. 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 1-9 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.) 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.

The **network** router subcommand is a mandatory configuration command and must be included in the configuration of each IP routing process.

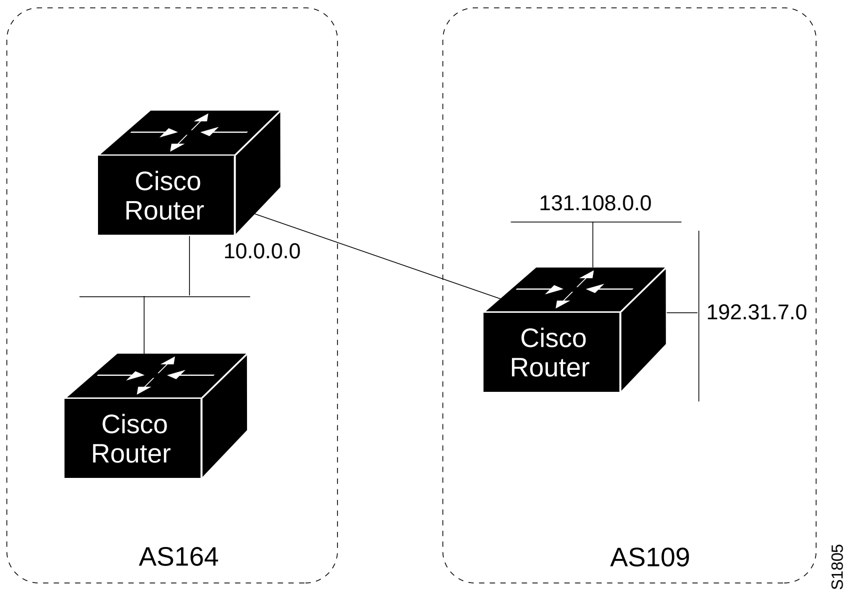
Note: For exterior protocols, a reference to an IP network from the **network** command that is learned by another routing protocol does not require a **redistribute** command. This is in contrast to interior gateway protocols, such as IGRP, which require the use of the **redistribute** command.

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 1-10. 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 1-10 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** router subcommand:

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
```

Generating an EGP Default Route

EGP can now use network 0.0.0.0 as a default route. EGP must be explicitly configured to generate a default route. The router subcommand is:

default-information originate
no default-information originate

If the next hop for the default route can be advertised as a third party, it will be included as a third party.

Defining a Core Gateway EGP Process

In some situations, certain external routing problems can be solved by having a single, central clearinghouse of routing information. The EGP protocol with *core gateway* support can be used to implement this structure.

Use the **router egp 0** global configuration command to allow a specific router to have an EGP process that will enable a router to act as a peer with any reachable autonomous system. This defines the router as a *core gateway*. Only one core gateway process may be configured in a router. The command syntax is:

router egp 0
no router egp 0

The EGP process defined with this command can act as a peer with any autonomous system (AS) and information is interchanged freely between autonomous systems.

Normally,. an EGP process expects to communicate with neighbors from a single AS. Since all neighbors are in the same AS, the EGP process assumes that these neighbors all have consistent internal information. So if the EGP process is informed about a route from one of its neighbors, it will not send it out to other neighbors.

With *core EGP*, the assumption is that all neighbors are from different ASes, and all have inconsistent information. In this case, the EGP process distributes routes from one neighbor to all others (but not back to the originator). This allows the EGP process to be a central clearinghouse for information.

Note: Split horizon is performed only on a *per-gateway* basis (in other words, if an external router informs a Cisco router about a specific network, and that router is the *best* path, the Cisco router will *not* inform the originating external router about that path). Cisco routers can also perform per-gateway split horizon on third-party updates.

To control how an EGP process determines which neighbors will be treated as peers, use the **neighbor any** router subcommand with the **router egp 0** global subcommand.The syntax for this command takes two forms:

neighbor any [*list*]
no neighbor any [*list*]

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

If the *list* argument is specified, the neighbor *must* be accepted by the access-list number specified to be allowed to peer with the EGP process.

The keyword/argument pair **third-party** *address* allows the specified address to be used as the next hop in EGP advertisements.

The optional keywords **internal** or **external** indicate whether the third-party router should be listed in the internal or external section of the EGP update.

Example Core Gateway EGP Configuration

Figure 1-11 illustrates an environment with three routers (designated C1, C2, and C3) attached to a common X.25 network that are intended to route information using EGP.

With the following configuration (on the router designated Core), C1, C2, and C3 cannot route traffic directly to each other via the X.25 network.

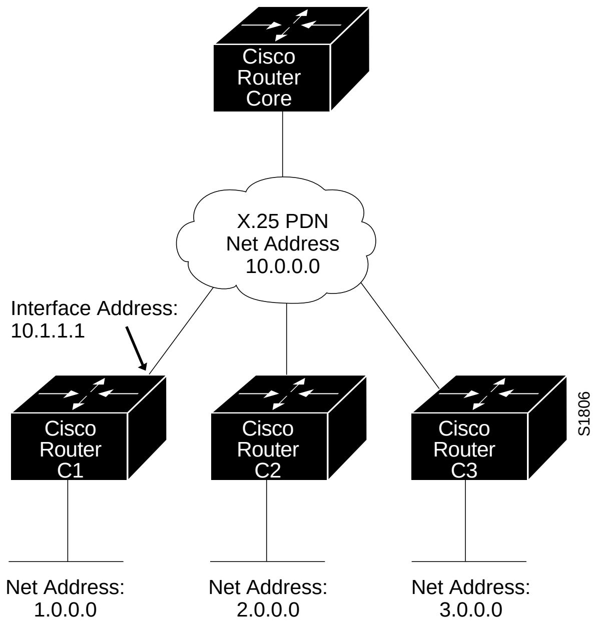
```
access-list 1 permit 10.0.0.0 0.255.255.255
! global access list assignment
router egp 0
neighbor any 1
```

This configuration specifies that an EGP process on any router on network 10.0.0.0 can act as a peer with the Core router. All traffic in this configuration will flow through the Core router.

Third-party advertisements allow traffic to bypass the Core router and go directly to the router that advertised reachability to Core.

```
access-list 2 permit 10.0.0.0 0.255.255.255
! global access list assignment
router egp 0
neighbor any 2
neighbor any third-party 10.1.1.1
```

Figure 1-11 Core EGP Third-Party Update Configuration Example



Filtering Routing Information

The information sent and received on the 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.

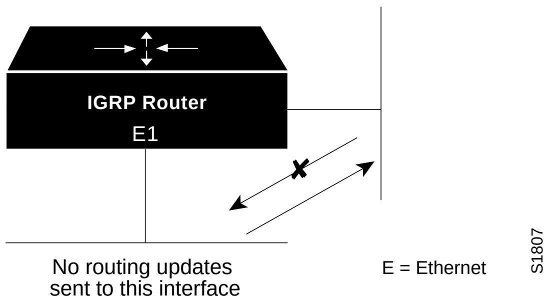
The **no passive-interface** command re-enables sending routing updates on the specified interface.

Example 1: IGRP Implementation

In the following example, IGRP updates are sent to all interfaces on network 131.108.0.0 except interface Ethernet 1. Figure 1-12 shows this configuration.

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

Figure 1-12 Filtering IGRP Updates



Example 2: With Neighbor Specification

As with Example 1, in the following example, IGRP updates are sent to all interfaces on network 131.108.0.0 except interface Ethernet 1. However, in this case a **neighbor** router subcommand is included. This command permits the sending of routing updates to specific neighbors. One copy of the routing update is generated per neighbor.

```
router igrp 109
network 131.108.0.0
passive-interface ethernet 1
neighbor 131.108.20.4
```

Example 3: OSPF Implementation

In OSPF, the **passive-interface** command disables OSPF Hello Protocol neighbor discovery.

This results in the Hello protocol on that interface being disabled, hence the router will not be able to discover any neighbors, and no OSPF neighbor will be able to see the router on that network. In effect, this interface will appear as stub network to OSPF domain. This is useful if you want to import routes associated with a connected network into OSPF domain, without any OSPF activity on that interface.

This command typically is used when the wildcard specification on the **network** router subcommand configures more interfaces than is desirable. The following configuration causes OSPF to run on all subnets of 131.108.0.0:

```
interface Ethernet 0
ip address 131.108.1.1 255.255.255.0
interface Ethernet 1
ip address 131.108.2.1 255.255.255.0
interface Ethernet 2
ip address 131.108.3.1 255.255.255.0

router ospf 109
network 131.108.0.0 0.0.255.255 area 0
```

If you do not want OSPF to run on 131.108.3.0 (as an example), then enter the following command:

```
router ospf 109
network 131.108.0.0 0.0.255.255 area 0
passive-interface Ethernet 2
```

Filtering Outbound 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 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.

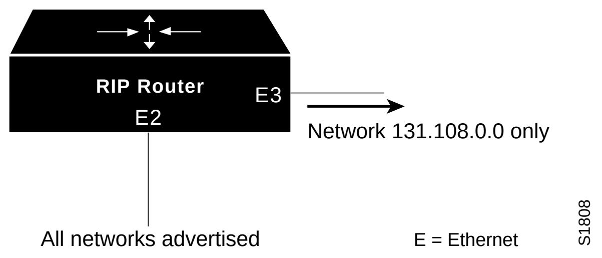
```
access-list 1 permit 0.0.0.0
access-list 1 deny 0.0.0.0 255.255.255.255
router rip
network 131.108.0.0
distribute-list 1 out
```

Example 2:

To filter a routing update sent on a specific interface, you can 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. Figure 1-13 illustrates the effects of this command.

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

Figure 1-13 Filtering RIP Updates



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*
no 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. The **no** version removes the neighbor.

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 ten 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 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* [*address mask*] [*access-list-number*]
no distance *weight* [*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 *address* and *mask* specifies a particular router or group of routers to which the weight value applies. The argument *address* is an Internet address that specifies a router, network, or subnet. The argument *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 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
Internal BGP	20
IGRP	100
OSPF`	110
RIP	120
Hello	130
EGP	140
External 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**]
> **no ip address** *address mask* [**secondary**]

Note: OSPF supports secondary address assignment.

The **no** version of this command removes the specified secondary-address association.

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 full syntax of the **ip route** global configuration 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. In this case, packets for network 10.0.0.0 will be routed via 131.108.3.4. if dynamic information with administrative distance less than 110 is not available.

```
ip route 10.0.0.0 131.108.3.4 110
```

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*
> **no ip default-network** *network-number*

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

If the router has a directly connected interface onto the specified 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 pseudonetwork 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. The **no** version of this command removes the specified default network.

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 readvertise IGRP-derived routes using the RIP protocol, or to readvertise 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). The following overview the router's automatic metric translations:

- 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 all dynamically-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 100, whichever is larger. Ethernets have a delay of 1 millisecond and serial links have a 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 IG-RP-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**
- **ospf**
- **igrp**

When you specify the **bgp, igrp** or **egp** keyword, use the optional argument *AS-number* 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 allowed {**in**|**out**}
no default-information allowed {**in**|**out**}

This subcommand controls the handling of default information between multiple processes.

The **no default-information allowed 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 allowed 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*
> **no 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 100 percent 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.

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

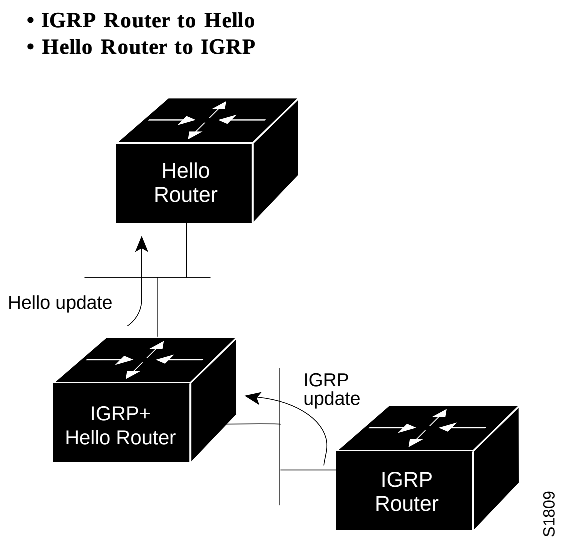
Example:

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 1-14 shows this type of redistribution.

Figure 1-14 Assigning Metrics for Redistribution



Redistributing Routes into OSPF

Routes from other OSPF routing domains and non-OSPF routing domains can be *redistributed* into a specific OSPF routing domain. This is accomplished with the **redistribute** router subcommand. The syntax for this command is as follows:

> **redistribute** *protocol* [*source-id*]
> [**metric** *metric-value*]
> [**metric-type** *type-value*]
> [**tag** *tag-value*]
> [**subnets**]

> **no redistribute** *protocol* [*source-id*]
> [**metric** *metric-value*]
> [**metric-type** *type-value*]
> [**tag** *tag-value*]
> [**subnets**]

The argument *protocol* is the source protocol from which routes are being redistributed. It can be one of the following keywords:

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

The optional argument *source-id* is either an AS (IGRP) or an appropriate OSPF process id from which routes are to be redistibuted. This value takes the form of either a positive integer. If the keywords **hello** or **rip** are used, then no *source-id* value is specified.

The optional keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the redistributed route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number. If a value is not specified for this option, and no value is specified using the **default-metric** router subcommand, the default metric value is 20.

Note: The **metric** value specified in the **redistribute** router subcommand supersedes the **metric** value specified using the **default-metric** router subcommand.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the OSPF routing domain. The *type-value* argument can assume one of two values:

- **1**—Type 1 external route
- **2**—Type 2 external route

If a **metric-type** is not specified, the router adopts a Type 2 external route.

The optional keyword/argument pair **tag** *tag-value* specifies a 32-bit decimal value attached to each external route. This is not used by the OSPF protocol itself. It may be used to communicate information between AS boundary routers. If none is specified, then the remote AS number is used for routes from BGP and EGP; for other protocols, zero (0) is used.

The optional keyword **subnets** specifies the scope of redistribution for the specified protocol. If **subnets** is not specified, only major networks are redistributed. If **subnets** is specified, major networks and subnets are redistributed. For the purposes of this discussion, a major network is any administratively assigned Class A, B, or C IP network.

Note: You can redistribute subnets from any IP routing protocol into OSPF and between different OSPF processes. In addition, if you do not specify the **subnets** keyword, routing information is effectively summarized at redistribution time. For example, assume subnet routes 10.1.0.0 and 10.2.0.0 are learned via IGRP. By omitting the **subnets** keyword, OSPF can be configured to redistribute those routes as a single net (10.0.0.0) route into the OSPF domain.

The **no redistribute** command disables redistribution for the protocol or OSPF process specified; it requires that all arguments be specified and disables redistribution only for specific protocols as routing processes. If the keywords **subnet** or **tag** are used, then the effects are isolated to processes associated with these arguments (for example, only subnet redistribution is disabled when the subnet keyword is included, while route redistribution into major nets continues).

Example

The following command example causes the specified IGRP process routes to be redistributed in to an OSPF domain. The IGRP-derived metric will be remapped to 100 and RIP routes to 200.

```
router ospf 109
redistribute igrp 108 metric 100 subnets
redistribute rip metric 200 subnets
```

Generating Default AS Boundary Router Routes for OSPF

Whenever you use the **redistribute** subcommand to redistribute routes into an OSPF routing domain, the router automatically becomes an AS boundary router. However, an AS boundary routers does not by default generate a *default route* into the OSPF routing domain. The **default-information** router subcommand allows you to force the AS boundary router do this. The **default-information** subcommand is always used with a **redistribute** command. The syntax of this router subcommand is as follows:

default-information originate metric *metric-value* **metric-type** *type-value*
no default-information originate metric *metric-value* **metric-type** *type-value*

The keyword **originate** causes the router to generate a default external route into an OSPF domain.

The keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the default route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the OSPF routing domain. The *type-value* argument can assume one of two values:

- **1**—Type 1 external route
- **2**—Type 2 external route

If a **metric-type** is not specified, the router adopts a Type 2 external route.

The **no default-information** command disables generation of a default route into the specified OSPF routing domain.

Example

The following example specifies a metric of 100 for the default route redistributed into the OSPF routing domain and an external metric type of Type 1.

```
router ospf 109
redistribute igrp 108 metric 100 subnets
default-information originate metric 100 metric-type 1
```

Redistributing OSPF Routes into Other Domains

Cisco's implementation of OSPF includes an extension to the **redistribute** subcommand that is specific to redistribution of routes from OSPF to other routing domains. The syntax for redistributing OSPF is:

redistribute ospf *ospf-process-id*
    [**metric** *metric-value*]
    [**match internal**|**external** *type-value* **external** *type-value*]

no redistribute ospf *ospf-process-id*
[**metric** *metric-value*]
[**match internal**|**external** *type-value* **external** *type-value*]

This subcommand only applies when redistributing OSPF routes to other routing protocols. You can select any combination of **internal** and/or **external** (Type 1 or Type 2) routes to redistribute. By default, if routes are redistributed into EGP or BGP, only **internal** routes are redistributed. Otherwise all routes are redistributed by default.

The argument *ospf-process-id* is the OSPF process id from which routes are to be redistibuted. This value takes the form of a decimal number.

The optional keyword/argument pair **metric** *metric-value* maps OSPF cost assigned to the redistributed route into the destination routing domain metric type. Use a value consistent with the destination protocol.

The optional keyword **match** specifies the criteria by which OSPF routes are redistributed into other routing domains. The keywords used are **internal** and **external**. The keyword **internal** refers to routes that are internal to a specific AS; the keyword **external** refers to routes that are external to the AS, but are to imported to OSPF as external routes.

The argument *type-value* specifies the external route type to be redistributed into other routing domains. The *type-value* argument can assume one of two values:

- **1**—Type 1 external route
- **2**—Type 2 external route

There is no default value.

Note: Any match variable is not exclusive; all may be specified. The example that follows illustrates the use of multiple matching criteria.

The **no redistribute** command disables redistribution for the OSPF process specified; you must specify the *ospf-process-id* and can only disable individual redistributions.

Example:

The following example illustrates the use of this version of the **redistribute** router subcommand, with the **match** keyword and its options enabled:

```
redistribute ospf 109 match internal external 1 external 2
```

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. The command syntax is:

ip route *network* {*address*|*interface*} [*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 argument *interface* is the name of the interface to use for 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.

Static routes that point to an interface (using the argument *interface*) are advertised via RIP and IGRP regardless of whether **redistribute static** commands were specified for those routing protocols. This is because static routes that point to an interface are considered in the routing table to be connected and hence lose their static nature. However, if you define a static route to an interface that is not one of the networks defined in a **network** command, neither RIP nor IGRP will advertise the route.

Example:

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
```

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 advertized 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, then 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 making a change in order to properly advertise routes. Remember: 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
```

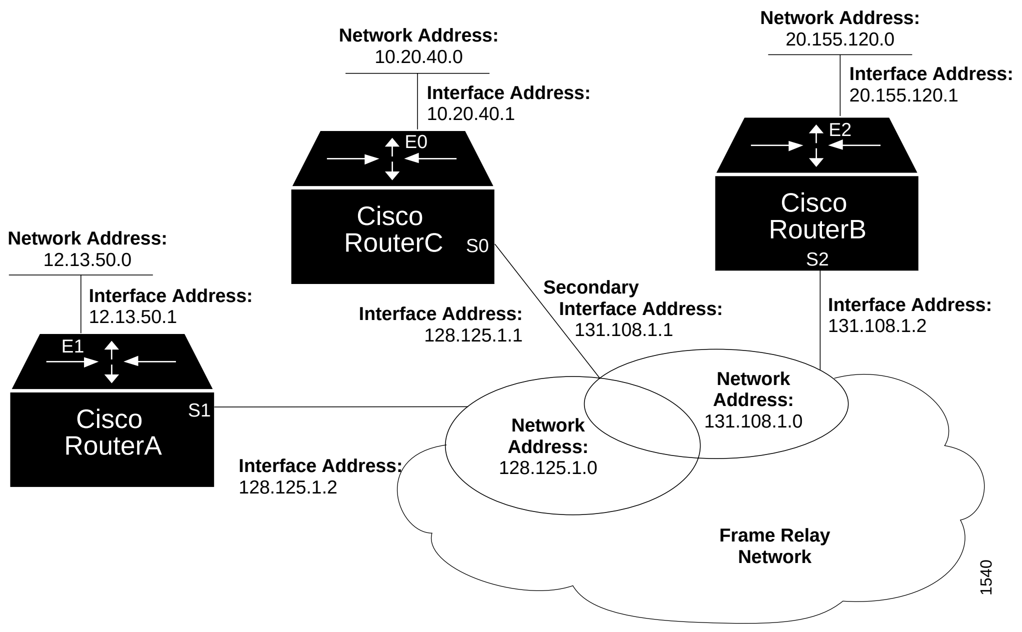
Example of Implicit Split Horizon Conditions

A typical situation in which the **no ip split-horizon** command would be useful is illustrated in Figure 1-15. This figure depicts two IP subnets that are both accessible via a serial interface on RouterC (connected to frame relay network). In this example, the serial interface on RouterC accommodates one of the subnets via the assignment of a secondary IP address.

The Ethernet interfaces for RouterA, RouterB, and RouterC (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 1-15 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 RouterC, 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 1-15 Disabled Split Horizon Example for Frame Relay Network



Example interface configuration for RouterA:

```
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 RouterB:

```
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 RouterC:

```
interface ethernet 0
ip address 10.20.40.1
!
interface serial 0
ip address 128.124.1.1
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 equals 0, the composite IGRP metric is computed according to the following formula:

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

If K5 does not equal 0, an additional operation is done:

```
metric = metric * [K5/(reliability + K4)]
```

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 bandwidth of the path in bits per second scaled by a factor of 1e10. 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. Table 1-3 lists the default values used for several common media.

Table 1-3 Default Bandwidth Values by Media Type

Media Type	Delay	Bandwidth
Satellite	200,000 (2 sec)	20 (500 Mbit)
Ethernet	100 (1 ms)	1,000
1.544 Mbps	2000 (20 ms)	6,476
64 Kbps	2000	156,250
56 Kbps	2000	178,571
10 Kbps	2000	1,000,000
1 Kbps	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 Release 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 (time in seconds between updates) at which updates are sent. This is the fundamental timing parameter of the routing protocol.
- The argument *invalid* is an interval of time (in seconds) after which a route is declared invalid; it should be three times the value of *update*. A route becomes invalid when there is an absence of updates that refresh the route. The route is marked inaccessible and advertised as unreachable. However, the route is still used for forwarding packets.
- The argument *holddown* is the interval (in seconds) during which routing information regarding better paths is suppressed. It should be at least three times the value of *update*. A route enters into a holddown state when an update packet is received that indicates the route is unreachable. The route is marked inaccessible and advertised as unreachable. However, the route is still used for forwarding packets. When the holddown interval expires, routes advertised by other sources are accepted and the route is no longer inaccessible.
- The argument *flush* is the amount of time (in seconds) that must pass before the route is removed from the routing table; the interval specified for *flush* must be at least the sum of *invalid* and *holddown*. If it is less than this sum, the proper holddown interval cannot elapse, which results in a new route being accepted before the holddown interval expires.

- The optional argument *sleeptime* is used to postpone IGRP routing updates for the specified number of milliseconds. Note that other timing values are specified in seconds. The *sleeptime* value should be less than the *update* time. If the *sleeptime* is greater than the *update* time, routing tables will become unsynchronized.

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 five seconds. If a router is not heard from in 15 seconds, the route is declared unusable. Further information is suppressed for an additional 15 seconds. At the end of the suppression period, the route is flushed from the routing table.

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

Note that by setting a short update period, you run the risk of congesting slow-speed serial lines; however, this is not much of a concern on faster-speed Ethernets and T1-rate serial lines. Also, if you have many routes in your updates, you can also cause the routers to spend an excessive amount of time processing updates.

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 a 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).

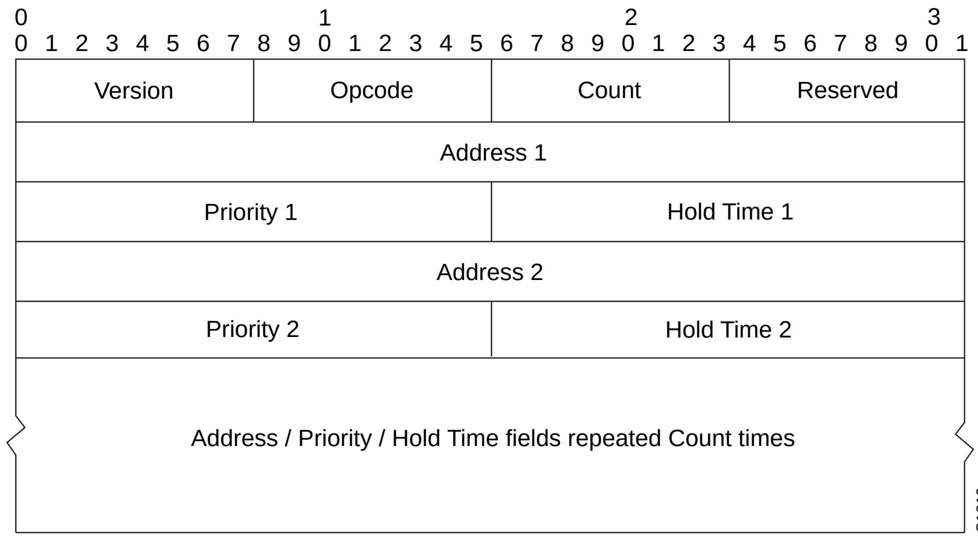
There are two types of GDP messages: *Report* and *Query.* On broadcast media, Report message packets are periodically sent to the IP broadcast address announcing that the router is present and functioning. By listening for these Report packets, a host can detect a vanishing or appearing router. If a 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 1-16 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.

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.

Figure 1-16 GDP Report Message Packet Format



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 interface 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 advertised. Do this by specifying the **redistribute static** subcommand, then specifying 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*, and 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
```

BGP Route Advertisement and Redistribution

The following examples illustrate configurations for advertising and redistributing BGP routes. The first example details the configuration for two neighboring routers that run IGRP within their respective ASes and that are configured to advertise their respective BGP routes between each other. The second example illustrates route redistribution of BGP into IGRP and IGRP into BGP.

Example 1: Simple BGP Route Advertisement

This example provides the required configuration for two routers (R1 and R2) that are intended to advertise BGP routes to each other and to redistribute BGP into IGRP.

```
! Router R1 Configuration:
```

```
! Assumes AS 1 has network number 131.108.0.0
router bgp 1
network 131.108.0.0
neighbor 131.108.1.1 remote-as 2
!
router igrp 1
network 131.108.0.0
redistribute bgp 1
! Note that IGRP is not redistributed into BGP
!
! Router R2 Configuration:
! Assumes AS 2 has network number 150.136.0.0:
router bgp 2
network 150.136.0.0
neighbor 131.108.1.2 remote-as 1
!
router igrp 2
network 150.136.0.0
redistribute bgp 2
```

Example 2: Mutual Route Redistribution

The most complex redistribution case is one in which *mutual* redistribution is required between an IGP (in this case IGRP) and BGP.

Suppose that EGP is running on a router somewhere else in AS 1, and that the EGP routes are injected into IGRP routing process 1. You must filter to ensure that the proper routes are advertised. The example configuration for router R1 illustrates use of access filters and a distribution list to filter routes advertised to BGP neighbors. This example also illustrates configuration commands for redistribution between BGP and IGRP.

```
! Configuration for router R1:
router bgp 1
network 131.108.0.0
neighbor 131.108.1.1 remote-as 2
neighbor 131.108.2.1 remote-as 1
! 131.108.2.1 an internal peer
neighbor 131.108.3.1 remote-as 3
! 131.108.3.1 is an external peer
redistribute igrp 1
distribute-list 1 out igrp 1
!
! All networks that should be
! advertised from R1 are
! controlled with access lists:
!
access-list 1 permit 131.108.0.0
access-list 1 permit 150.136.0.0
access-list 1 permit 128.125.0.0
!
router igrp 1
network 131.108.0.0
redistribute bgp 1
```

OSPF Routing and Route Redistribution

OSPF typically requires coordination among many internal routers, area border routers, and AS boundary routers. At a minium, OSPF-based routers can be configured with all default parameter values, with no authentication, and with interfaces assigned to areas.

If you intend to customize your environment, you must ensure coordinated configurations of all routers. Activities that require careful planning include:

- Establishing stub areas
- Enabling and defining OSPF authentication
- Attaching routers to nonbroadcast networks
- Modifying specific OSPF interface options
- Creating virtual links

Three examples follow:

- The first is a simple configuration illustrating basic OSPF commands.
- The second example illustrates a configuration for internal, area border, and AS boundary routers within a single, arbitrarily assigned, OSPF AS.
- The third example illustrates a more complex configuration and the application of various tools available for controlling OSPF-based routing environments.

Example 1: Basic OSPF Configuration

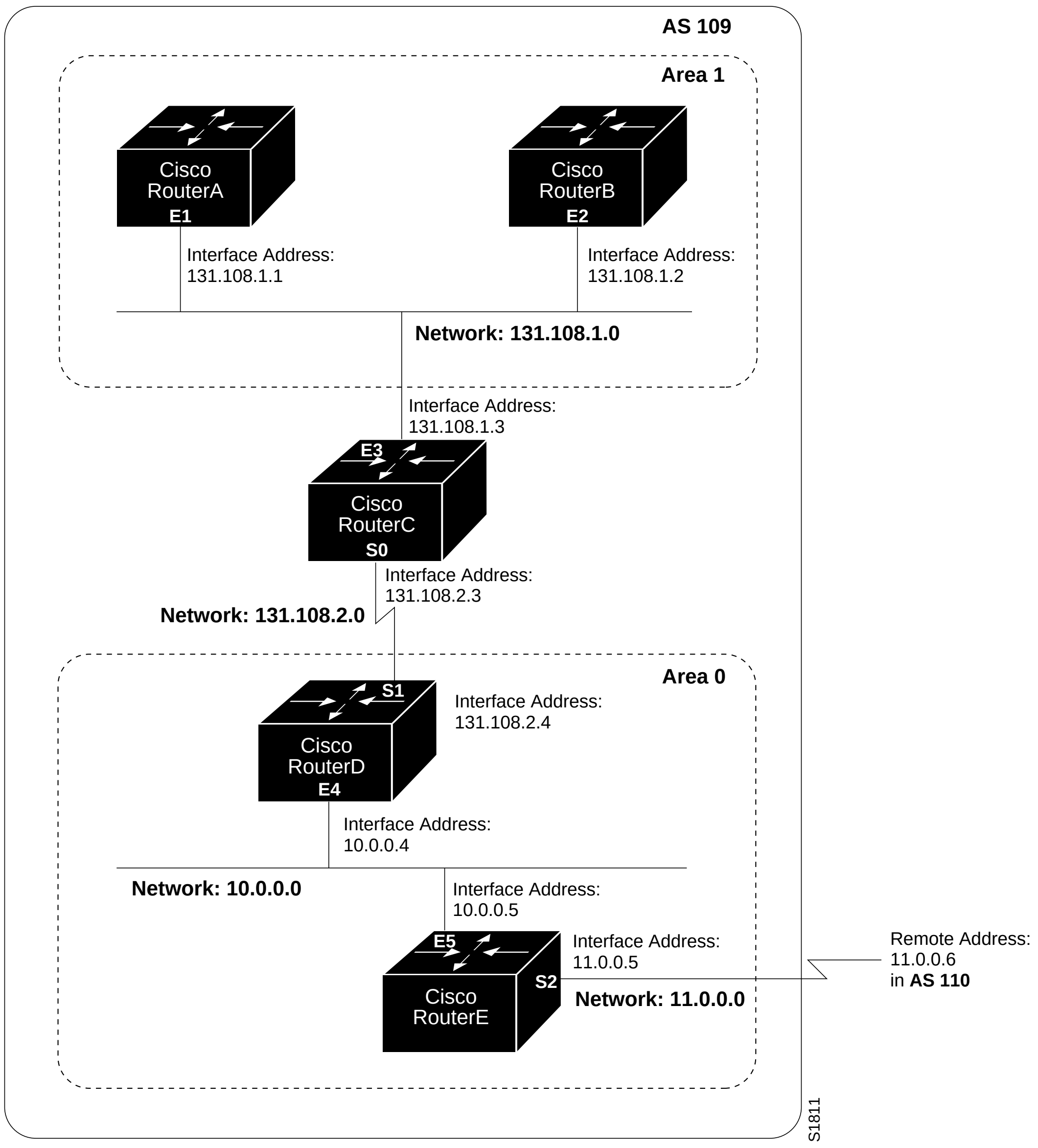
The following example illustrates a simple OSPF configuration that enables OSPF routing process 9000, attaches Ethernet 0 to area 0.0.0.0, and redistributes RIP into OSPF.

```
interface Ethernet0
ip address 130.93.1.1 255.255.255.0
ip ospf cost 1
!
interface Ethernet 1
ip address 130.94.1.1 255.255.255.0
!
router ospf 9000
network 130.93.0.0 0.0.255.255 area 0.0.0.0
redistribute rip metric 1 subnets
!
router rip
network 130.94.0.0
redistribute ospf 9000
default-metric 1
```

Example 2: Internal, Area Border and AS Boundary Routers

The following example outlines a configuration for several routers within a single OSPF autonomous system. Figure 1-17 provides a general network map that illustrates this example configuration.

Figure 1-17 Sample OSPF Autonomous System Network Map


AS 109
Area 1
Cisco RouterA
E1
Cisco RouterB
E2
Interface Address: 131.108.1.1
Interface Address: 131.108.1.2
Network: 131.108.1.0
Interface Address: 131.108.1.3
E3
Cisco RouterC
S0
Interface Address: 131.108.2.3
Network: 131.108.2.0
Area 0
S1
Interface Address: 131.108.2.4
Cisco RouterD
E4
Interface Address: 10.0.0.4
Network: 10.0.0.0
Interface Address: 10.0.0.5
E5
Interface Address: 11.0.0.5
S2
Cisco RouterE
Network: 11.0.0.0
Remote Address: 11.0.0.6 in AS 110
S1811

In this configuration, five routers are configured in OSPF autonomous system AS 109:

- Routers RouterA and RouterB are both internal routers, within Area 1.
- RouterC is an OSPF area border router; note that for RouterC, Area 1is assigned to subnet 1 and Area 0 is assigned to subnet 2.
- RouterD is an internal router in Area 0 (backbone area); in this case, both **network** router subcommands specify the same area (Area 0, or the backbone area).
- RouterE is an OSPF AS boundary router; note that BGP routes are redistributed into OSPF and that these route are advertised by OSPF.

Note: It is not necessary to include definitions of all areas in an OSPF AS in the configuration of all routers in the AS. You need only define the *directly* connected areas. In the example that follows, routes in Area 0 are learned by the routers in Area 1 (RouterA and RouterB) when the area border router (*RouterC*) injects summary link state advertisements (LSAs) into Area 1.

AS 109 is connected to the outside world via the BGP link to the external peer at IP address 11.0.0.6.

```
! RouterA - internal router
interface Ethernet 1
ip address 131.108.1.1 255.255.255.0

router ospf 109
network 131.108.0.0 0.0.255.255 area 1

! RouterB - internal router
interface Ethernet 2
ip address 131.108.1.2 255.255.255.0

router ospf 109
network 131.108.0.0 0.0.255.255 area 1
!
! RouterC - area border router
interface Ethernet 3
ip address 131.108.1.3 255.255.255.0

interface Serial 0
ip address 131.108.2.3 255.255.255.0

router ospf 109
network 131.108.1.0 0.0.0.255 area 1
network 131.108.2.0 0.0.0.255 area 0
!
! RouterD - internal router
interface Ethernet 4
ip address 10.0.0.4 255.0.0.0

interface Serial 1
ip address 131.108.2.4 255.255.255.0
```

```
router ospf 109
network 131.108.2.0 0.0.0.255 area 0
network 10.0.0.0 0.255.255.255 area 0
!
! RouterE - AS boundary router
interface Ethernet 5
ip address 10.0.0.5 255.0.0.0

interface Serial 2
ip address 11.0.0.5 255.0.0.0

router ospf 109
network 10.0.0.0 0.255.255.255 area 0
redistribute bgp 109 metric 1 metric-type 1

router bgp 109
network 131.108.0.0
network 10.0.0.0
neighbor 11.0.0.6 remote-as 110
```

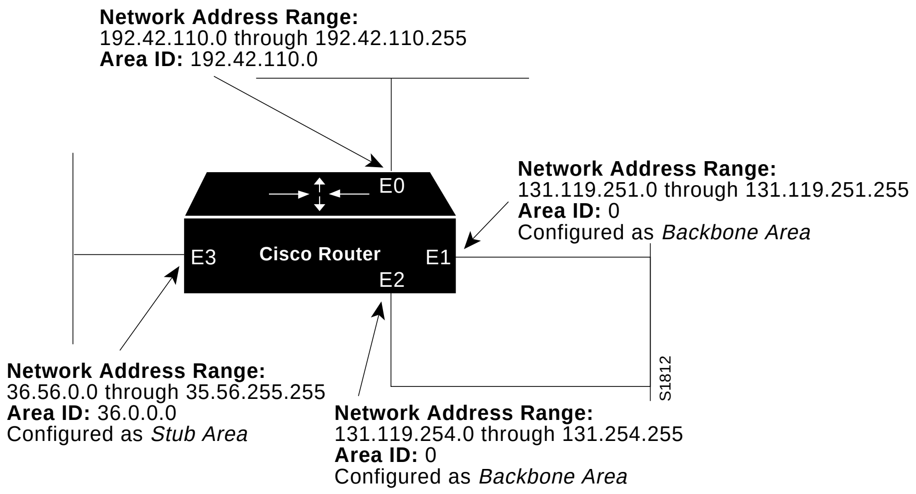
Example 3: Complex OSPF Configuration

The following example configuration accomplishes several tasks in setting up an area border router. These tasks can be split into two general categories:

- Basic OSPF configuration
- Route redistribution

The specific tasks outlined in this configuration are detailed briefly in the following descriptions. Figure 1-18 illustrates the network address ranges and area assignments for the interfaces.

Figure 1-18 Interface and Area Specifications for OSPF Example Configuration


Network Address Range:
192.42.110.0 through 192.42.110.255
Area ID: 192.42.110.0
E0
Network Address Range:
131.119.251.0 through 131.119.251.255
Area ID: 0
Configured as *Backbone Area*
E3
Cisco Router
E1
E2
S1812
Network Address Range:
36.56.0.0 through 35.56.255.255
Area ID: 36.0.0.0
Configured as *Stub Area*
Network Address Range:
131.119.254.0 through 131.254.255
Area ID: 0
Configured as *Backbone Area*

The basic configuration tasks in this example are as follows:

- Configure address ranges for interfaces Ethernet 0 through Ethernet 3.
- Enable OSPF on each interface.
- Set up an OSPF authentication password for each area and network.
- Assign link state metrics and other OSPF interface configuration options.
- Create a *stub area* with area id 36.0.0.0. (Note that the **authentication** and **stub** options of the **area** router subcommand are specified with separate subcommand entries, but can be merged into a single **area** subcommand.)
- Specify the backbone area (Area 0).

Configuration tasks associated with redistribution are as follows:

- Redistribute IGRP and RIP into OSPF with various options set (including **metric-type**, **metric**, **tag**, and **subnet**).
- Redistribute IGRP and OSPF into RIP.

The following is an example OSPF configuration command listing:

```
interface Ethernet0
ip address 192.42.110.201 255.255.255.0
ip ospf authentication-key abcdefgh
ip ospf cost 10
!
interface Ethernet1
ip address 131.119.251.201 255.255.255.0
ip ospf authentication-key ijklmnop
ip ospf cost 20
ip ospf retransmit-interval 10
ip ospf transmit-delay 2
ip ospf priority 4
!
interface Ethernet2
ip address 131.119.254.201 255.255.255.0
ip ospf authentication-key abcdefgh
ip ospf cost 10
!
interface Ethernet3
ip address 36.56.0.201 255.255.0.0
ip ospf authentication-key ijklmnop
ip ospf cost 20
ip ospf dead-interval 80
!
! OSPF on network 131.119
!
router ospf 201
network 36.0.0.0 0.255.255.255 area 36.0.0.0
network 192.42.110.0 0.0.0.255 area 192.42.110.0
network 131.119.0.0 0.0.255.255 area 0
area 0 authentication
area 36.0.0.0 stub
area 36.0.0.0 authentication
area 36.0.0.0 default-cost 20
area 192.42.110.0 authentication
area 36.0.0.0 range 36.0.0.0 255.255.0.0
area 192.42.110.0 range 192.42.110.0 255.255.255.0
area 0 range 131.119.251.0 255.255.255.0
area 0 range 131.119.254.0 255.255.255.0

redistribute igrp 200 metric-type 2 metric 1 tag 200 subnets
redistribute rip metric-type 2 metric 1 tag 200
!
! IGRP AS 200 on 131.119.0.0
!
router igrp 200
network 131.119.0.0
!
! RIP for 192.42.110
!
router rip
network 192.42.110.0
redistribute igrp 200 metric 1
redistribute ospf 201 metric 1
```

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 an asterisk (*), 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, local router ID is 131.108.6.69
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 the 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 non-BGP 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 interautonomous 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. A question mark (`?`) 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 10, external link
BGP version 3, remote router ID 131.108.6.68
BGP state = Established, table version = 22, up for 0:00:13
Last read 0:00:12, hold time is 180, keepalive interval is 60 seconds
Received 24 messages, 0 notifications
Sent 28 messages, 4 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 its 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 next line specifies that the BGP version being used to communicate with the remote router is BGP version 3; the neighbor's router id (IP address) is also specified.

- 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 ensure 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 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 1-4 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 1-4 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

Field	Description
Poll	Interval between Poll/Update packets
j/k	Measure of reachability; 4 is perfect
Flags	• Perm—Permanent • Temp—Temporary (neighbor will be removed) • 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.
- The 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 hold-down 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 OSPF Routing Processes

To display general information about OSPF routing processes in a particular router, use the **show ip ospf** EXEC command. The command syntax is:

show ip ospf [*ospf-process-id*]

The optional argument *ospf-process-id* is a specific process id. If this argument is included, only information for the specified routing process is included.

The following is a partial sample **show ip ospf** output when entered without a specific OSPF process id:

```
gwtest>show ip ospf

 Routing Process "ospf 201" with ID 192.42.110.200
 Supports only single TOS(TOS0) routes
 It is an area border and autonomous system boundary router
 Summary Link update interval is 0:30:00 and the update due in 0:16:26
 External Link update interval is 0:30:00 and the update due in 0:16:27
 Redistributing External Routes from,
    igrp 200 with metric mapped to 2, includes subnets in redistribution
    rip with metric mapped to 2
    igrp 2 with metric mapped to 100
    igrp 32 with metric mapped to 1
 Number of areas in this router is 3
    Area 192.42.110.0
        Number of interfaces in this area is 1
        Area has simple password authentication
        SPF algorithm executed 6 times
        Area ranges are
           192.42.110.0  255.255.255.0 Active(1)
        Link State Update Interval is 0:30:00 and due in 0:16:25
        Link State Age Interval is 0:20:00 and due in 0:16:25
```

This display provides the following information:

- Sequential list of information for each routing process in the configuration
- Router id
- Number of Types of Service supported (Type 0 only)
- Type of OSPF router
- Link Update Interval
- Route redistribution specified in configuration
- Number of areas in the router and the type of areas
- Number of interfaces in the area
- Form of authentication
- Number of times the SPF algorithm has run since the software was initialized
- Area ranges
- Link State Update Interval and Link State Age Interval, and their expected time due

Displaying the OSPF Database

To display lists of information related to the OSPF database for a specific router, use the **database** option of the **show ip ospf** EXEC command. There are several versions of this command, with each delivering information about different OSPF link states advertisements. The syntax for the various versions of this command is as follows:

show ip ospf [*ospf-process-id area-id*] **database**
show ip ospf [*ospf-process-id area-id*] **database** [**router**] [*link-state-id*]
show ip ospf [*ospf-process-id area-id*] **database** [**network**] [*link-state-id*]
show ip ospf [*ospf-process-id area-id*] **database** [**summary**] [*link-state-id*]
show ip ospf [*ospf-process-id area-id*] **database** [**asb-summary**] [*link-state-id*]
show ip ospf [*ospf-process-id*] **database** [**external**] [*link-state-id*]

When entered with the optional keywords **router**, **network**, **summary**, and **asb-summary**, a different information is displayed. Samples and brief descriptions of each version follow.

Three optional arguments are valid for each of these command variations: *ospf-process-id*, *area-id*, and *link-state-id*.

- *ospf-process-id*—Internally used identification parameter. It is locally assigned and can be any positive integer number. The number used here is the number assigned administratively when enabling the OSPF routing process.
- *area-id*—Area number associated with the OSPF address range defined in the **network** command used to define the particular area.
- *link-state-id*—Identifies the portion of the Internet environment that is being described by the advertisement. The value entered depends on the advertisement's LS type. It must be entered in the form of an IP address.

When the link state advertisement is describing a network, the *link-state-id* can be one of two forms:

- The network's IP address (as in type 3 summary link advertisements and in AS external link advertisements)
- A derived address obtained from the link state id. (Note that masking a network links advertisement's link state id with the network's subnet mask yields the network's IP address.)

When the link state advertisement is describing a router, the link state id is always the described router's OSPF router id.

When an AS external advertisement (LS Type = 5) is describing a default route, its link state id is set to Default Destination (0.0.0.0).

The following is a sample output from the **show ip ospf database** general display with no optional arguments or keywords:

```
gwtest>show ip ospf database


       OSPF Router with id(190.20.239.66) (Autonomous system 300)


                Displaying Router Link States(Area 0.0.0.0)

  Link ID      ADV Router       Age      Seq#      Checksum  Link count
155.187.21.6   155.187.21.6    1731    0x80002CFB    0x69BC       8
155.187.21.5   155.187.21.5    1112    0x800009D2    0xA2B8       5
155.187.1.2    155.187.1.2     1662    0x80000A98    0x4CB6       9
155.187.1.1    155.187.1.1     1115    0x800009B6    0x5F2C       1
155.187.1.5    155.187.1.5     1691    0x80002BC     0x2A1A       5
155.187.65.6   155.187.65.6    1395    0x80001947    0xEEE1       4
155.187.241.5  155.187.241.5   1161    0x8000007C    0x7C70       1
155.187.27.6   155.187.27.6    1723    0x80000548    0x8641       4
155.187.70.6   155.187.70.6    1485    0x80000B97    0xEB84       6

                Displaying Net Link States(Area 0.0.0.0)

  Link ID       ADV Router      Age        Seq#       Checksum
155.187.1.3  192.20.239.66     1245    0x800000EC      0x82E

                Displaying Summary Net Link States(Area 0.0.0.0)

  Link ID       ADV Router       Age        Seq#        Checksum
155.187.240.0   155.187.241.5    1152      0x80000077      0x7A05
155.187.241.0   155.187.241.5    1152      0x80000070      0xAEB7
155.187.244.0   155.187.241.5    1152      0x80000071      0x95CB
```

Fields in this display provide the following information:

- Link id
- Advertising router's router id
- Link state age
- Link state sequence number (detects old or duplicate link state advertisements)

- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Link count (number of interfaces detected for router)

The following is a partial sample output from the **show ip ospf database router** display with no optional arguments:

```
gwtest>show ip ospf database router


       OSPF Router with id(190.20.239.66) (Autonomous system 300)


                Displaying Router Link States(Area 0.0.0.0)

  LS age: 1176
  Options: (No TOS-capability)
  LS Type: Router Links
  Link State ID: 155.187.21.6
  Advertising Router: 155.187.21.6
  LS Seq Number: 80002CF6
  Checksum: 0x73B7
  Length: 120
  AS Boundary Router
155   Number of Links: 8

    Link connected to: another Router (point-to-point)
     (link ID) Neighboring Router ID: 155.187.21.5
     (Link Data) Router Interface address: 155.187.21.6
      Number of TOS metrics: 0
       TOS 0 Metrics: 2
```

Fields in this display provide the following information:

- Router id number
- OSPF autonomous system number (OSPF process id)
- Link state age
- Type of Service options (Type 0 only)
- Link state type
- Link state id
- Advertising router’s router id
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)
- Definition of router type
- Number of active links
- Link type
- Link id

- Router interface address
- Type of Service metric (Type 0 only)

The following is a sample output from the **show ip ospf database network** display with no optional arguments:

```
gwtest>show ip ospf database network


              OSPF Router with id(190.20.239.66) (Autonomous system 300)


                       Displaying Net Link States(Area 0.0.0.0)

    LS age: 1367
    Options: (No TOS-capability)
    LS Type: Network Links
    Link State ID: 155.187.1.3 (address of Designated Router)
    Advertising Router: 190.20.239.66
    LS Seq Number: 800000E7
    Checksum: 0x1229
    Length: 52
    Network Mask: 255.255.255.0
          Attached Router: 190.20.239.66
          Attached Router: 155.187.241.5
          Attached Router: 155.187.1.1
          Attached Router: 155.187.54.5
          Attached Router: 155.187.1.5
          Attached Router: 155.187.1.2
          Attached Router: 155.187.21.5
```

Fields in this display provide the following information:

- Router id number
- OSPF autonomous system number (OSPF process id)
- Link state age
- Type of Service options (Type 0 only)
- Link state type
- Link state id of designated router
- Advertising router’s router id
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)
- Network mask implemented
- List of routers attached to the network (by IP address)

The following is a sample output from the **show ip ospf database summary** display with no optional arguments.

```
gwtest>show ip ospf database summary

       OSPF Router with id(190.20.239.66) (Autonomous system 300)

                Displaying Summary Net Link States(Area 0.0.0.0)

  LS age: 1401
  Options: (No TOS-capability)
  LS Type: Summary Links(Network)
  Link State ID: 155.187.240.0 (summary Network Number)
  Advertising Router: 155.187.241.5
  LS Seq Number: 80000072
  Checksum: 0x84FF
  Length: 28
  Network Mask: 255.255.255.0   TOS: 0  Metric: 1
```

Fields in this display provide the following information for each network:

- Router id number
- OSPF autonomous system number (OSPF process id)
- Link state age
- Type of Service options (Type 0 only)
- Link state type
- Link state id (summary network number)
- Advertising router's router id
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)
- Network mask implemented, Type of Service, and link state metric

The following is a sample output from the **show ip ospf database asb-summary** display with no optional arguments:

```
gwtest>show ip ospf database asb-summary

       OSPF Router with id(190.20.239.66) (Autonomous system 300)

                Displaying Summary ASB Link States(Area 0.0.0.0)

  LS age: 1463
  Options: (No TOS-capability)
  LS Type: Summary Links(AS Boundary Router)
  Link State ID: 155.187.245.1 (AS Boundary Router address)
  Advertising Router: 155.187.241.5
  LS Seq Number: 80000072
  Checksum: 0x3548
  Length: 28
  Network Mask: 0.0.0.0 TOS: 0  Metric: 1
```

Fields in this display provide the following AS boundary router information:

- Router id number
- OSPF autonomous system number (OSPF process id)
- Link state age
- Type of Service options (Type 0 only)
- Link state type
- Link state id (AS Boundary Router)
- Advertising router's router id
- Link state sequence number (detects old or duplicate link state advertisements)
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement)
- Length (length in bytes of the link state advertisement)
- Network mask implemented, Type of Service, and link state metric

The following is a sample output from the **show ip ospf database external** display with no optional arguments:

```
gwtest>show ip ospf database external


       OSPF Router with id(190.20.239.66) (Autonomous system 300)

       Displaying AS External Link States

  LS age: 280
  Options: (No TOS-capability)
  LS Type: AS External Link
  Link State ID: 143.105.0.0 (External Network Number)
  Advertising Router: 155.187.70.6
  LS Seq Number: 80000AFD
  Checksum: 0xC3A
  Length: 36
  Network Mask: 255.255.0.0
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 1
        Forward Address: 0.0.0.0
        External Route Tag: 0
```

Fields in this display provide the following external links information:

- Router id number.
- OSPF autonomous system number (OSPF process id).
- Link state age.
- Type of Service options (Type 0 only).
- Link state type.
- Link state id (External Network Number).

- Advertising router's router id.
- Link state sequence number (detects old or duplicate link state advertisements).
- LS checksum (the Fletcher checksum of the complete contents of the link state advertisement).
- Length (length in bytes of the link state advertisement).
- Network mask implemented.
- External type.
- Type of Service.
- Link state metric.
- Forwarding address (data traffic for the advertised destination will be forwarded to this address. If the forwarding address is set to 0.0.0.0, data traffic will be forwarded instead to the advertisement's originator.
- External route tag (a 32-bit field attached to each external route; this is not used by the OSPF protocol itself).

Displaying OSPF Interface Parameters

To display OSPF-related interface information, use the **show ip ospf interface** EXEC command. The command syntax is:

show ip ospf interface [*interface-name*]

The argument *interface-name* can be formed either as the one-word interface description (for example, serial0, ethernet0, or fddi1), or can be formed as the two-word *interface-type unit-number* specification (for example, serial 0, ethernet 1, or e 2,).

The following is a sample output from the **show ip ospf interface** display with Ethernet interface 0 specifically stated:

```
gwtest>show ip ospf interface ethernet 0

Ethernet 0 is up, line protocol is up
  Internet Address 131.119.254.202, Mask 255.255.255.0, Area 0.0.0.0
  AS 201, Router ID 192.77.99.1, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State OTHER, Priority 1
  Designated Router id 131.119.254.10, Interface address 131.119.254.10
  Backup Designated router id 131.119.254.28, Interface address 
131.119.254.28
  Timer intervals configured, Hello 10, Dead 60, Wait 40, Retransmit 5
    Hello due in 0:00:05
  Neighbor Count is 8, Adjacent neighbor count is 2
     Adjacent with neighbor 131.119.254.28  (Backup Designated Router)
     Adjacent with neighbor 131.119.254.10  (Designated Router)
```

Fields in this display provide the following interface-related information:

- Status of physical link and operational status of protocol
- Interface IP address, subnet mask, and area address

- AS number (OSPF process id), router id, network type, link state cost
- Transmit delay, interface state, and router priority
- Designated router id and respective interface IP address
- Backup designated router id and respective interface IP address
- Configuration of timer intervals
- Anticipated arrival of next Hello packet
- Count of network neighbors and list of adjacent neighbors

Displaying OSPF Neighbor Information

To display OSPF-neighbor information on a per-interface basis, use the **show ip ospf neighbor** EXEC command. The command syntax is:

show ip ospf neighbor [*interface-name*]

The argument *interface-name* can be formed either as the one-word interface description (for example, serial0, ethernet0, or fddi1), or can be formed as the two-word *interface-type unit-number* specification (for example, serial 0, ethernet 1, or e 2).

The following is a partial sample output from the **show ip ospf neighbor** display with no interface argument:

```
gwtest>show ip ospf neighbor serial0

 Neighbor 155.187.241.5, interface address 155.187.1.240
    In the area 0.0.0.0 via interface Ethernet0
    Neighbor priority is 1, State is FULL
    Options 2
    Dead timer due in 0:00:59
 --More--
 Neighbor 155.187.1.1, interface address 155.187.1.1
    In the area 0.0.0.0 via interface Ethernet0
    Neighbor priority is 1, State is FULL
    Options 2
    Dead timer due in 0:00:55
```

Fields in this display provide the following interface-related information:

- Neighbor router id and interface address
- Area and interface through which OSPF neighbor is known
- Router priority of neighbor, neighbor state
- Hello packet options field contents (E-bit only; possible values are 0 and 2; 2 indicates area is not a stub; 0 indicates area is a stub)
- Expected time before router will declare neighbor dead

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, O - OSPF
derived
       C - connected, S - static, E - EGP derived, B - BGP derived
       * - candidate default route, IA - OSPF inter area route
      E1 - OSPF external type 1 route, E2 - OSPF external type 2 route

Gateway of last resort is 131.119.254.240 to network 129.140.0.0

O E2 150.150.0.0 [160/5] via 131.119.254.6, 0:01:00, Ethernet2
E    192.67.131.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
O E2 192.68.132.0 [160/5] via 131.119.254.6, 0:00:59, Ethernet2
O E2 130.130.0.0 [160/5] via 131.119.254.6, 0:00:59, Ethernet2
E    128.128.0.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E    129.129.0.0 [200/129] via 131.119.254.240, 0:02:22, Ethernet2
E    192.65.129.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E    131.131.0.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E    192.75.139.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet2
E    192.16.208.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E    192.84.148.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet2
E    192.31.223.0 [200/128] via 131.119.254.244, 0:02:22, Ethernet2
E    192.44.236.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet2
E    140.141.0.0 [200/129] via 131.119.254.240, 0:02:22, Ethernet2
E    141.140.0.0 [200/129] via 131.119.254.240, 0:02:23, Ethernet2
```

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 may list certain protocol-specific information as defined in the display header.
- The third 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 fourth column specifies the address of the router that can build a route to the specified remote network.
- The fifth column specifies the last time the route was updated in hours:minutes:seconds.
- 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 follows:

- 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
  - 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)
  - `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, OSPF 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 **debug ip-bgp** command provides debug messages about BGP events, including inbound updates.

debug ip-bgp-events

The **debug ip-bgp-events** command provides debug messages about BGP events, including BGP state machine changes and outbound updates.

debug ip-egp [*IP-address*]
debug ip-egp-events [*IP-address*]

These EXEC commands allow for the presentation of debugging information relating to a particular neighbor. The argument *IP-address* is optional. If omitted, debugging information about all neighbors. 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-ospf-adj

The **debug ip-ospf-adj** command provides information concerning database synchronization and the election of designated routers.

debug ip-ospf-events

The **debug ip-ospf-events** command provides information concerning OSPF related events, such as adjacencies, flooding information, how designated routers are selected, and how SPF is calculated.

debug ip-ospf-flood

The **debug ip-ospf-flood** command provides information about link state advertisement flooding.

debug ip-ospf-packets

The **debug ip-ospf-packet** command displays debugging information on OSPF packet traffic. This display confirms the receipt of each OSPF-related packet (Hello, link state request, and so on).

debug ip-ospf-spf

The **debug ip-ospf-spf** command provides information about how the SPF tree is built and how routes are derived.

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 as-path access-list** *list* {**permit**|**deny**} *as-regular-expression*

Defines a BGP-related access list. The *list* argument is a number between 1 and 99.The **permit** and **deny** keywords specify the type of access allowed. The argument *as-regular-expression* allows use of special characters in specifying AS in access list.

[**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.

ip route *network* {*address*|*interface*} [*distance*]

Establishes static routes. 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.

[**no**] **router ospf** *ospf-process-id*

Enables OSPF for the router. The argument *ospf-process-id* is an internally used identification parameter. It is locally assigned and can be any positive integer number. A unique value is assigned for each OSPF routing process. You can specify multiple OSPF routing processes in each router.

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

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

[**no**] **router egp 0**

Allows a specific router to have an EGP process that will enable a router to act as a peer with any reachable autonomous system. This defines the router as a *core gateway*. Only one core gateway process may be configured in a router.

Router Subcommand Summary

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

[**no**] **area** *area-id* **authentication**

Enables authentication for an area. The argument *area-id* is the specific area id of the area for which authentication is to be enabled. The authentication *type* (AuType 0 or AuType 1) must be the same for all routers in an area. The authentication *key* (password) for all OSPF routers on a network must be the same if they are to communicate with each other via OSPF. Use the **ip ospf authentication-key** interface subcommand to specify this password.

[**no**] **area** *area-id* **range** *address mask*

Consolidates or summarizes routes. The result is that a single summary route is advertized to other areas by the area border router. This command is only used with area border routers.

The argument *area-id* is the specific area id for the area about which routes are to be summarized. The argument *area-id* can be specified as either a decimal value or as an IP address.

The *address* argument is a standard IP address.

The *mask* argument is a standard IP mask.

[**no**] **area** *area-id* **stub**
[**no**] **area** *area-id* **default-cost** *cost*

Defines an area as a stub area. These commands are used only on an area border router attached to a stub.

The argument *area-id* is the specific area id for the stub area. The argument *area-id* can be specified as either a decimal value or as an IP address.

The **stub** option is used to enable the stub area.

The **default-cost** *cost* keyword/argument pair assigns a specific cost for the default external route used for the stub area. The acceptable value is a 24-bit number.

[**no**] **area** *area-id* **virtual-link** *router-id* [**hello-interval** *number-of-seconds*]
[**retransmit-interval** *number-of-seconds*]
[**transmit-delay** *number-of-seconds*]
[**dead-interval** *number-of-seconds*]
[**authentication-key** *8-bytes-of-password*]

Defines virtual links.

The argument *area-id* is the area id assigned to the transit area for the virtual link.

The argument *router-id* is the router id associated with the virtual link neighbor.

There are no default value for the *area-id* or *router-id* arguments for this command.

Specify the length of time, in seconds, between the Hello packets that the router sends on the interface with the **hello-interval** option of the **area** *area-id* **virtual-link** router subcommand. The argument *number of-seconds* is an unsigned integer value. This value is advertised in the router's Hello packets. It must be the same for all routers attached to a common network. The smaller the Hello interval, the faster topological changes will be detected, but more routing traffic will ensue. The default is ten seconds.

The number of seconds between link state advertisement retransmissions for adjacencies belonging to the interface is specified with the **retransmit-interval** option of the **area** *area-id* **virtual-link** router subcommand. The value for the *number-of-seconds* argument is a integer number, that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links. The default value five seconds.

The estimated number of seconds it takes to transmit a link state update packet on the interface is specified with the **transmit-delay** option of the **area** *area-id* **virtual-link** router subcommand. Link state advertisements in the update packet must have their age incremented by this amount before transmission. The value assigned should take into account the transmission and propagation delays for the interface. The argument *number-of-seconds* is a integer value that must be greater than zero. The default value is one second.

Set the number of seconds that a router's Hello packets have not been seen before its neighbors declare the router down with the **dead-interval** option of the **area** *area-id* **virtual-link** router subcommand. The argument *number of-seconds* is an unsigned integer value. The default is four times the HelloInterval. As with the HelloInterval, this value must be the same for all routers attached to a common network.

Assign a specific password to be used by neighboring routers with the **authentication-key** option of the **area** *area-id* **virtual-link** router subcommand. The argument *8-bytes-of-password* is any continuous string of characters that you can enter from the keyboard up to eight bytes in length. This configured data allows the authentication procedure to generate and/or verify the authentication field in the OSPF header. There is no default value.

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

Controls the handling of default information between multiple IGRP processes. The **no default-information allowed 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 allowed out** subcommand causes IGRP exterior routes to be suppressed in updates. Default information is normally passed between IGRP processes.

[**no**] **default-information originate**

Explicitly configures EGP to generate a default route. If the next hop for the default route can be advertised as a third party, it will be included as a third party.

[**no**] **default-information originate metric** *metric-value* **metric-type** *type-value*

Allows you to force the AS Boundary Router redistribute OSPF routes. Always use this command with a **redistribute** command.

The keyword **originate** causes the router to generate a default external route into an OSPF domain.

The keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the default route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the OSPF routing domain. The *type-value* argument can assume one of two values:

- **1**—Type 1 external route
- **2**—Type 2 external route

If a **metric-type** is not specified, the router adopts Type 2 external route.

The **no default-information** command disables generation of a default route into the specified OSPF routing domain.

[**no**] **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. The **no default-metric** command causes the current routing protocol to return to using the built-in, automatic metric translations.

[**no**] **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. The **no default-metric** command causes the current routing protocol to return to using the built-in, automatic metric translations.

[**no**] **distance bgp** *external-distance internal-distance local-distance*

Specifies administrative distance.

The argument *external-distance* specifies the value for BGP external routes. External routes are routes those for which the best path is learned from a neighbor external to the autonomous system. Acceptable values are positive, nonzero integers.

The argument *internal-distance* specifies the value for BGP internal routes. Internal routes are routes that are learned from another BGP entity within the same autonomous system. Acceptable values are positive, nonzero integers.

The argument *local-distance* specifies the value for BGP local routes. Local routes are those networks listed with a **network** command (possibly as backdoors) for that router or networks that are being redistributed from another process.

By default, the administrative distances are as follows

- *external-distance*—20
- *internal-distance*—200
- *local-distance*—200

[**no**] **distance** *weight* [*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 *address* and *mask* specifies a particular router or group of routers to which the *weight* applies. The argument *address* is an Internet address that specifies a router, network, or subnet. The argument *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 specify the interface. When redistributing networks, you may specify a routing process name.

[**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.

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

Control how an EGP process determines which neighbors will be treated as peers; used with the **router egp 0** router subcommand.

If the *list* argument is specified, the neighbor *must* be accepted by the access list to be allowed to peer with the EGP process.

The keyword/argument pair **third-party** *address* allows the specified address to be used as the next hop in EGP advertisements.

The optional keywords **internal** or **external** indicate whether the third-party router should be listed in the internal or external section of the EGP update.

[**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-list** *list* **in**
[**no**] **neighbor** *address* **filter-list** *list* **out**
[**no**] **neighbor** *address* **filter-list** *list* **weight** *weight*

Establishes filters using access lists defined with the **ip as-path access-list** command.

The argument *address* is the address of the neighbor.

The argument *list* is a predefined BGP access list number.

The keywords **in** and **out** specify whether you are applying the access list to incoming or outgoing routes.

The keyword/argument pair **weight** *weight* assigns a relative importance to a specific list. Any number weight filters are allowed on a per neighbor basis, but only one in or out filter is allowed

[**no**] **neighbor** *address* **interface** *type unit-number* [**priority** *8-bit-number*]
[**poll-interval** *number-of-seconds*]

Configures OSPF-based routers interconnecting to nonbroadcast networks.

The argument *address* is the specific interface IP address of the neighbor.

The keyword/argument sequence **interface** *type unit-number* identifies the specific router interface for this **neighbor** command.The interface must be connected to a non-broadcast, multiaccess type network. In addition, the neighbor specified must be eligible to be a designated router or backup designated router.

The keyword/argument pair **priority** *8-bit number* is the router priority value of the nonbroadcast neighbor associated with the IP address specified.

If a neighboring router has become inactive (Hello packets have not been seen for Router DeadInterval period), it may still be necessary to send Hello packets to the dead neighbor. These Hello packets will be sent at a reduced rate called the *Poll Interval* (PollInterval).

Specify this interval with the keyword/argument pair **poll-interval** *number-of-seconds*. The argument *number-of-seconds* is an unsigned integer value. RFC 1247 recommends that this value should be much larger than the HelloInterval. The default is 2 minutes (120 seconds).

neighbor *address* **remote-as** *number*
no neighbor *address*

Adds a neighbor entry to the routing table for BGP. The keywords *address* and *number* 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**] **neighbor** *ip-address*

Adds to a list of neighbor routers. 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.

[**no**] **neighbor** *ip-address* **version** *value*

Configures router to handle only handle a specific version of BGP.

The argument *ip-address* is the address of the BGP-speaking neighbor; the version *value* can be set to 2 to force the router to only use Version 2 with the specified neighbor. The default is to speak Version 3 of BGP and dynamically negotiate down to Version 2 if requested. The **no neighbor** *ip-address* **version** *value* command returns the version to the default state for that neighbor.

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

Specifies the list of networks with the **network** router configuration subcommand. 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. Use this version of the command for BGP, EGP, RIP, Hello, and IGRP protocols. The **no** version of this command removes the specified network number.

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

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

[**no**] **network** *address wildcard-mask* **area** *area-id*

Specifies a range of IP addresses for any area in which OSPF is to be used as a routing protocol. The *address* and *wildcard-mask* pair together define a range of IP addresses to be associated with a specific OSPF area. The argument *address* is formed as an IP address. The argument *wildcard-mask* is an IP-address-type mask that includes “don’t care” bits.

The keyword/variable argument pair **area** *area-id* specifies an area to be associated with the OSPF address range as defined in the same **network** command. The argument *area-id* can be specified as either a decimal value or as an IP address. If you intend to associate areas with IP subnets, you can specify a subnet address as the *area-id*.

[**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 specified with the *list* argument (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**, **bgp, egp**, **hello**, **igrp**. Use the optional argument *AS-number* when you specify the **bgp, igrp** or **egp** keyword to specify the autonomous system number.

[**no**] **redistribute** *protocol* [*source-id*]
[**metric** *metric-value*]
[**metric-type** *type-value*]
[**tag** *tag-value*]
[**subnets**]

Redistributes routes from other OSPF routing domains and non-OSPF routing domains into a specific OSPF routing domain. The argument *protocol* is the source protocol from which routes are being redistributed. It can be one of the following keywords:

- **bgp**
- **egp**
- **hello**
- **igrp**
- **ospf**

- **rip**
- **static**

The optional argument *source-id* is either and Autonomous System (IGRP) or an appropriate OSPF process id from which routes are to be redistributed. This value takes the form of either a positive integer. If the keywords **hello** or **rip** are used, then no *source-id* value is specified.

The optional keyword/argument pair **metric** *metric-value* specifies the link state cost to be assigned to the redistributed route. The *metric-value* argument is a dimensionless link state cost, formed as a 24-bit decimal number. If a value is not specified for this option, and no value is specified using the **default-metric** router subcommand, the default **metric** value is 20.

The keyword/argument pair **metric-type** *type-value* specifies the external link type associated with the default route advertised into the OSPF routing domain. The *type-value* argument can assume one of two values:

- **1**—Type 1 external route
- **2**—Type 2 external route

If a **metric-type** is not specified, the router adopts Type 2 external route.

The optional keyword/argument pair **tag** *tag-value* specifies a 32-bit decimal value attached to each external route. The optional keyword **subnets** specifies the scope of redistribution for the specified protocol.

[**no**] **redistribute ospf** *ospf-process-id*
    [**metric** *metric-value*]
    [**match internal**|**external** *type-value*|**external** *type-value*]

Redistributes routes from OSPF to other routing domains. You can select any combination of **internal** and/or **external** (Type 1 or Type 2) routes to redistribute. By default, if routes are redistributed into EGP or BGP, only **internal** routes are redistributed. Otherwise all routes are redistributed by default.

The argument *ospf-process-id* is the OSPF process id from which routes are to be redistributed. This value takes the form of either a decimal number or an IP address.

The optional keyword/argument pair **metric** *metric-value* maps OSPF cost assigned to the redistributed route into the destination routing domain metric type. Use a value consistent with the destination protocol.

The optional keyword **match** specifies the criteria by which OSPF routes are redistributed into other routing domains. The keywords used are **internal** and **external**. The keyword **internal** refers to routes that are internal to a specific AS; the keyword **external** refers to routes that are external to the AS, but are to imported to OSPF as external routes.

The argument *type-value* specifies the external route type to be redistributed into other routing domains. The *type-value* argument can assume one of two values:

- **1**—Type 1 external route

- **2**—Type 2 external route

There is no default value.

[**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 the 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 Hello messages 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.

[**no**] **variance** *multiplier*

Controls load balancing in an IGRP-based internet. The argument *multiplier* defines the amount of metric variance that will be accepted. Acceptable values are nonzero, positive integers. By default, the amount of variance is set to one. The **no** version resets variance to one.

IP Routing Interface Subcommands

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

[**no**] **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. The **no** version of the command removes the specified secondary-address association.

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

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

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

Enables or disables GDP routing, specifying hold time 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 5 seconds for Ethernets and the hold time 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 ospf authentication-key** *8-bytes-of-password*

Assigns a specific password to be used by neighboring routers on a wire that are using OSPF's simple password authentication.

The argument *8-bytes-of-password* is any continuous string of characters up to eight bytes in length that you can enter from the keyboard.

[**no**] **ip ospf cost** *cost*

Explicitly specifies the cost of sending a packet on an interface. The argument *cost* is expressed as the link state metric. It is a dimensionless integer value, that is always greater than zero. This value is advertized as the link cost in the router's router links advertisement. Cisco does not support Type of Service (TOS), so you can assign only one cost per interface.

[**no**] **ip ospf dead-interval** *number-of-seconds*

Sets the number of seconds that a router's Hello packets have not been seen before its neighbors declare the router down. This value is advertised in the router's Hello packets in the *DeadInt* field.

The argument *number of-seconds* is an unsigned integer value.

The default is four times the HelloInterval. As with the HelloInterval, this value must be the same for all routers attached to a common network.

[**no**] **ip ospf hello-interval** *number-of-seconds*

Specifies the length of time, in seconds, between the Hello packets that the router sends on the interface.

The argument *number of-seconds* is an unsigned integer value. This value is advertised in the router's Hello packets. It must be the same for all routers attached to a common network. The smaller the Hello interval, the faster topological changes will be detected, but more routing traffic will ensue.

[**no**] **ip ospf priority** *8-bit-number*

Sets the router priority, which helps determine the designated router for a network.

The argument *8-bit-number* is an 8-bit unsigned integer.

The default router priority is 1.

[**no**] **ip ospf retransmit-interval** *number-of-seconds*

Sets the number of seconds between link state advertisement retransmissions for adjacencies belonging to the interface.

The value for the *number-of-seconds* argument is an integer number that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links.

The default value is five seconds.

[**no**] **ip ospf transmit-delay** *number-of-seconds*

Sets the estimated number of seconds it takes to transmit a link state update packet on the interface.

Link state advertisements in the update packet must have their age incremented by this amount before transmission. The value assigned should take into account the transmission and propagation delays for the interface.

The argument *number-of-seconds* is an integer value that must be greater than zero. The default value is one second.

[**no**] **ip split-horizon**

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

[**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.