This appendix contains supplementary Border Gateway Protocol (BGP) information and covers the following topics:

- BGP Route Summarization
- Redistribution with IGPs
- Policy Control and Prefix Lists
- Communities
- Route Reflectors

APPENDIX E

BGP Supplement

This appendix provides you with some additional information on the Border Gateway Protocol (BGP).

BGP Route Summarization

This section reviews classless interdomain routing (CIDR) and describes how BGP supports CIDR and summarization of addresses. Both the **network** and **aggregate-address** commands are described.

CIDR and Aggregate Addresses

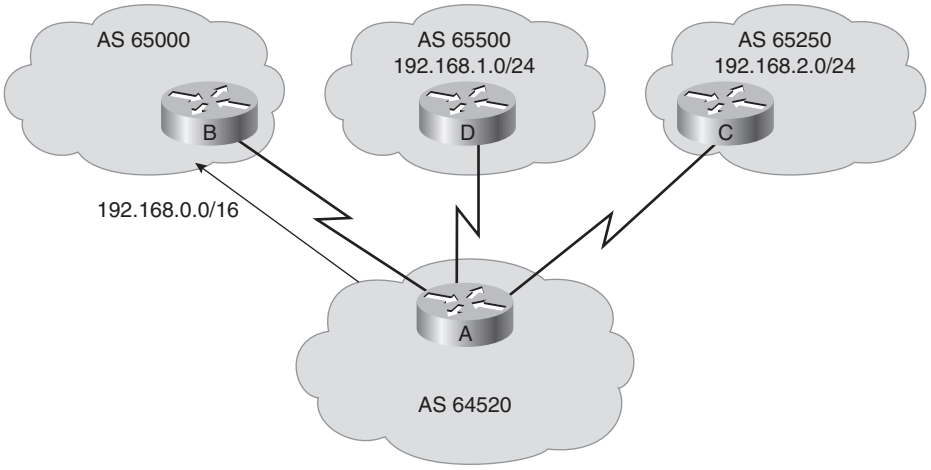
As discussed in Appendix C, "IPv4 Supplement," CIDR is a mechanism developed to help alleviate the problem of exhaustion of IP addresses and the growth of routing tables. The idea behind CIDR is that blocks of multiple addresses (for example, blocks of Class C address) can be combined, or aggregated, to create a larger classless set of IP addresses. These multiple addresses can then be summarized in routing tables, resulting in fewer route advertisements.

Earlier versions of BGP did not support CIDR; BGP Version 4 (BGP4) does. BGP4 support includes the following:

- The BGP update message includes both the prefix and the prefix length. Previous versions included only the prefix; the length was assumed from the address class.
- Addresses can be aggregated when advertised by a BGP router.
- The autonomous system (AS)-path attribute can include a combined unordered list of all autonomous systems that all the aggregated routes have passed through. This combined list should be considered to ensure that the route is loop-free.

For example, in Figure E-1, router C is advertising network 192.168.2.0/24, and router D is advertising network 192.168.1.0/24. Router A could pass those advertisements to router B; however, router A could reduce the size of the routing tables by aggregating the two routes into one (for example, 192.168.0.0/16).

Figure E-1 *Using CIDR with BGP*



NOTE In Figure E-1, the aggregate route that router A is sending covers more than the two routes from routers C and D. The example assumes that router A also has jurisdiction over all the other routes covered by this aggregate route.

Two BGP attributes are related to aggregate addressing:

- **Atomic aggregate**—A well-known discretionary attribute that informs the neighbor autonomous system that the originating router has aggregated the routes
- **Aggregator**—An optional transitive attribute that specifies the BGP router ID and autonomous system number of the router that performed the route aggregation

KEY POINT

Aggregate Routes

By default, the aggregate route is advertised as coming from the autonomous system that did the aggregation and has the atomic aggregate attribute set to show that information might be missing. The autonomous system numbers from the nonaggregated routes are *not* listed.

You can configure the router to include the unordered list of all autonomous systems contained in all paths that are being summarized.

NOTE Indications are that aggregate addresses are not used in the Internet as much as they could be because autonomous systems that are multihomed (connected to more than one Internet service provider [ISP]) want to make sure that their routes are advertised without being aggregated into a summarized route.

In Figure E-1, by default the aggregated route 192.168.0.0/16 has an autonomous system path attribute of (64520). If router A were configured to include the combined unordered list, it would include the set {65250 65500} as well as (64520) in the autonomous system path attribute; the autonomous system path would be the unordered set {64520 65250 65500}.

Network Boundary Summarization

BGP was originally not intended to be used to advertise subnets. Its intended purpose was to advertise classful, or better, networks. *Better* in this case means that BGP can summarize blocks of individual classful networks into a few large blocks that represent the same address space as the individual network blocks—in other words, CIDR blocks. For example, 32 contiguous Class C networks can be advertised individually as 32 separate entries, with each having a network mask of /24. Or it might be possible to announce these same networks as a single entry with a /19 mask.

Consider how other protocols handle summarization. The Routing Information Protocol Version 1 (RIPv1), Routing Information Protocol Version 2 (RIPv2), and Enhanced Interior Gateway Routing Protocol (EIGRP) protocols all summarize routes on the classful network boundary by default. In contrast, Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS) do not summarize by default, but you can configure summarization manually.

You can turn off autosummarization for RIPv2 and EIGRP. For example, if you are assigned a portion of a Class A, B, or C address, summarization needs to be turned off; otherwise, you risk claiming ownership of the whole Class A, B, or C address.

NOTE The Internet Assigned Numbers Authority (IANA) is reclaiming Class A addresses from organizations that no longer need them. IANA breaks these Class A addresses into blocks of /19 address space, which are assigned to various ISPs to be given out in place of Class C addresses. This process has helped make the Internet a classless environment.

BGP works differently than the other protocols. As discussed in Chapter 8, "Configuring the Border Gateway Protocol," the **network** *network-number* [**mask** *network-mask*] [**route-map** *map-tag*] router configuration command for BGP permits BGP to advertise a network if it is present in the IP routing table. This command allows classless prefixes; the router can advertise individual subnets, networks, or supernets. The default mask is the classful mask and results in only the classful network number being announced. Note that at least one subnet of the specified major network must be present in the IP routing table for BGP to start announcing the classful network. However, if you specify the *network-mask*, an exact match to the network (both address and mask) must exist in the routing table for the network to be advertised.

The BGP **auto-summary** command determines how BGP handles redistributed routes. The **no auto-summary** router configuration command turns off BGP autosummarization. When

summarization is enabled (with **auto-summary**), all redistributed subnets are summarized to their classful boundaries in the BGP table. When summarization is disabled (with **no auto-summary**), all redistributed subnets are present in their original form in the BGP table. For example, if an ISP assigns a network of 10.100.50.0/24 to an autonomous system, and that autonomous system then uses the **redistribute connected** command to introduce this network into BGP, BGP announces that the autonomous system owns 10.0.0.0/8 if the **auto-summary** command is on. To the Internet, it appears as if this autonomous system owns all the Class A network 10.0.0.0/8, which is not true. Other organizations that own a portion of the 10.0.0.0/8 address space might have connectivity problems because of this autonomous system claiming ownership for the whole block of addresses. This outcome is undesirable if the autonomous system does not own the entire address space. Using the **network 10.100.50.0 mask 255.255.255.0** command rather than the **redistributed connected** command ensures that this assigned network is announced correctly.

CAUTION Recall that in Cisco IOS Release 12.2(8)T, the default behavior of the **auto-summary** command was changed to disabled. In other words

- Before 12.2(8)T, the default is **auto-summary**.
- Starting in 12.2(8)T, the default is **no auto-summary**.

BGP Route Summarization Using the network Command

To advertise a simple classful network number, use the **network** *network-number* router configuration command without the **mask** option. To advertise an aggregate of prefixes that originate in this autonomous system, use the **network** *network-number* [**mask** *network-mask*] router configuration command with the **mask** option (but remember that the prefix must exactly match [both address and mask] an entry in the IP routing table for the network to be advertised).

When BGP has a **network** command for a classful address and it has at least one subnet of that classful address space in its routing table, it announces the classful network and not the subnet. For example, if a BGP router has network 172.16.22.0/24 in the routing table as a directly connected network, and a BGP **network 172.16.0.0** command, BGP announces the 172.16.0.0/16 network to all neighbors. If 172.16.22.0 is the only subnet for this network in the routing table and it becomes unavailable, BGP will withdraw 172.16.0.0/16 from all neighbors. If instead the command **network 172.16.22.0 mask 255.255.255.0** is used, BGP will announce 172.16.22.0/24 and not 172.16.0.0/16.

KEY POINT

The network Command

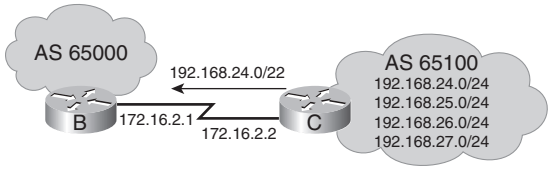
The **network** command requires that there be an exact match in the routing table for the prefix or mask that is specified. This exact match can be accomplished by using a static route with a null 0 interface, or it might already exist in the routing table, such as because of the Interior Gateway Protocol (IGP) performing the summarization.

Cautions When Using the network Command for Summarization

The **network** command tells BGP what to advertise but not how to advertise it. When using the BGP **network** command, the network number specified must also be in the IP routing table before BGP can announce it.

For example, consider router C in Figure E-2; it has the group of addresses 192.168.24.0/24, 192.168.25.0/24, 192.168.26.0/24, and 192.168.27.0/24 already in its routing table. The configuration in Example E-1 is put on router C.

Figure E-2 *BGP Network for Summarization Examples*



Example E-1 *Sample BGP Configuration for Router C in Figure E-2*

```
router bgp 65100
network 192.168.24.0
network 192.168.25.0
network 192.168.26.0
network 192.168.27.0
network 192.168.24.0 mask 255.255.252.0
neighbor 172.16.2.1 remote-as 65000
```

Each of the four Class C networks is announced because each already exists in the routing table. These networks are summarized with the **network 192.168.24.0 mask 255.255.252.0** command on router C; however, with this command the 192.168.24.0/22 route is *not* announced by default because that route is not in the routing table. If the IGP supports manual summarization (as EIGRP or OSPF do), and the same summarization is performed by the IGP command, BGP announces that summarized route. If route summarization is not performed with the IGP, and BGP is required to announce this route, a static route should be created that allows this network to be installed in the routing table.

The static route should point to the null 0 interface (using the command **ip route 192.168.24.0 255.255.252.0 null0**). Remember that 192.168.24.0/24, 192.168.25.0/24, 192.168.26.0/24, and 192.168.27.0/24 addresses are already in the routing table. This command creates an additional entry of 192.168.24.0/22 as a static route to null 0. If a network, such as 192.168.25.0/24, becomes unreachable, and packets arrive for 192.168.25.1, the destination address is compared to the current entries in the routing table using the longest-match criteria. Because 192.168.25.0/24 no longer exists in the routing table, the best match is 192.168.24.0/22, which points to the null 0 interface. The packet is sent to the null 0 interface, and an Internet Control Message Protocol

(ICMP) unreachable message is generated and sent to the packet's originator. Dropping these packets prevents traffic from using up bandwidth following a default route that is either deeper into your autonomous system or (in a worst-case scenario) back out to the ISP (when the ISP would route it back to the autonomous system because of the summarized route advertised to the ISP, causing a routing loop).

In this example, five networks are announced using **network** commands: the four Class C plus the summary route. The purpose of summarization is to reduce the advertisement's size, as well as the size of the Internet routing table. Announcing these more specific networks along with the summarized route actually increases the table's size.

Example E-2 shows a more efficient configuration. A single entry represents all four networks, and a static route to null 0 installs the summarized route in the IP routing table so that BGP can find a match. By using this **network** command, the autonomous system 65100 router advertises a summarized route for the four Class C addresses (192.168.24.0/24, 192.168.25.0/24, 192.168.26.0/24, and 192.168.27.0/24) assigned to the autonomous system. For this new **network** command (192.168.24.0/22) to be advertised, it must first appear in the local routing table. Because only the more specific networks exist in the IP routing table, a static route pointing to null 0 has been created to allow BGP to announce this network (192.168.24.0/22) to autonomous system 65000.

Example E-2 *More-Efficient BGP Configuration for Router C in Figure E-2*

```
router bgp 65100
  network 192.168.24.0 mask 255.255.252.0
  neighbor 172.16.2.1 remote-as 65000
ip route 192.168.24.0 255.255.252.0 null 0
```

Although this configuration works, the **network** command itself was not designed to perform summarization; the **aggregate-address** command, described in the next section, was designed to perform summarization.

Creating a Summary Address in the BGP Table Using the aggregate-address Command

The **aggregate-address** *ip-address mask* [**summary-only**] [**as-set**] router configuration command is used to create an aggregate, or summary, entry in the BGP table, as described in Table E-1.

Table E-1 **aggregate-address** *Command Description*

Parameter	Description
ip-address	Identifies the aggregate address to be created.
mask	Identifies the mask of the aggregate address to be created.

Table E-1 **aggregate-address** *Command Description (Continued)*

Parameter	Description
summary-only	(Optional) Causes the router to advertise only the aggregated route. The default is to advertise both the aggregate and the more specific routes.
as-set	(Optional) Generates autonomous system path information with the aggregate route to include all the autonomous system numbers listed in all the paths of the more specific routes. The default for the aggregate route is to list only the autonomous system number of the router that generated the aggregate route.

KEY POINT

aggregate-address Versus network Commands

The **aggregate-address** command aggregates only networks that are already in the *BGP table*. This is different from the requirement for advertising summaries with the BGP **network** command, in which case the network must exist in the *IP routing table*.

When you use this command without the **as-set** keyword, the aggregate route is advertised as coming from your autonomous system, and the atomic aggregate attribute is set to show that information might be missing. The atomic aggregate attribute is set unless you specify the **as-set** keyword.

Without the **summary-only** keyword, the router still advertises the individual networks. This can be useful for redundant ISP links. For example, if one ISP is advertising only summaries, and the other is advertising a summary plus the more specific routes, the more specific routes are followed. However, if the ISP advertising the more specific routes becomes inaccessible, the other ISP advertising only the summary is followed.

When the **aggregate-address** command is used, a BGP route to null 0 is automatically installed in the IP routing table for the summarized route.

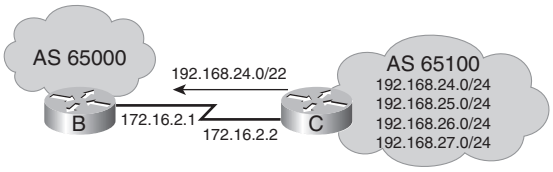
If any route already in the BGP table is within the range indicated by the **aggregate-address**, the summary route is inserted into the BGP table and is advertised to other routers. This process creates more information in the BGP table. To get any benefits from the aggregation, the more-specific routes covered by the route summary should be suppressed using the **summary-only** option. When the more specific routes are suppressed, they are still present in the BGP table of the router doing the aggregation. However, because the routes are marked as suppressed, they are never advertised to any other router.

For BGP to announce a summary route using the **aggregate-address** command, at least one of the more specific routes must be in the BGP table; this is usually a result of having **network** commands for those routes.

If you use only the **summary-only** keyword on the **aggregate-address** command, the summary route is advertised, and the path indicates only the autonomous system that did the summarization (all other path information is missing). If you use only the **as-set** keyword on the **aggregate-address** command, the set of autonomous system numbers is included in the path information (and the command with the **summary-only** keyword is deleted if it existed). However, you may use *both* keywords on one command; this causes only the summary address to be sent and all the autonomous systems to be listed in the path information.

Figure E-3 illustrates a sample network (it is the same network as in Figure E-2, repeated here for your convenience). Example E-3 shows the configuration of router C using the **aggregate-address**.

Figure E-3 *BGP Network for Summarization Examples*



Example E-3 *Configuration for Router C in Figure E-3 Using the* **aggregate-address** *Command*

```
router bgp 65100
  network 192.168.24.0
  network 192.168.25.0
  network 192.168.26.0
  network 192.168.27.0
  neighbor 172.16.2.1 remote-as 65000
aggregate-address 192.168.24.0 255.255.252.0 summary-only
```

This configuration on router C shows the following:

- **router bgp 65100**—Configures a BGP process for autonomous system 65100.
- **network** commands—Configure BGP to advertise the four Class C networks in autonomous system 65100.
- **neighbor 172.16.2.1 remote-as 65000**—Specifies the router at this address (router B) as a neighbor in autonomous system 65000. This part of the configuration describes *where* to send the advertisements.
- **aggregate-address 192.168.24.0 255.255.252.0 summary-only**—Specifies the aggregate route to be created but suppresses advertisements of more specific routes to all neighbors. This part of the configuration describes *how* to advertise. Without the **summary-only** option,

the new summarized route would be advertised along with the more specific routes. In this example, however, router B receives only one route (192.168.24.0/22) from router C. The **aggregate-address** command tells the BGP process to perform route summarization and automatically installs the null route representing the new summarized route.

KEY POINT

BGP Commands

The following summarizes the differences between the main BGP commands:

- The **network** command tells BGP *what* to advertise.
- The **neighbor** command tells BGP *where* to advertise.
- The **aggregate-address** command tells BGP *how* to advertise the networks.

The **aggregate-address** command does not replace the **network** command; at least one of the more specific routes to be summarized must be in the BGP table. In some situations, the more-specific routes are injected into the BGP table by other routers, and the aggregation is done in another router or even in another autonomous system. This approach is called *proxy aggregation*. In this case, the aggregation router needs only the proper **aggregate-address** command, not the **network** commands, to advertise the more specific routes.

The **show ip bgp** command provides information about route summarization and displays the local router ID, the networks recognized by the BGP process, the accessibility to remote networks, and autonomous system path information. In Example E-4, notice the *s* in the first column for the lower four networks. These networks are being suppressed. They were learned from a **network** command on this router; the next-hop address is 0.0.0.0, which indicates that this router created these entries in BGP. Notice that this router also created the summarized route 192.168.24.0/22 in BGP (this route also has a next hop of 0.0.0.0, indicating that this router created it). The more-specific routes are suppressed, and only the summarized route is announced.

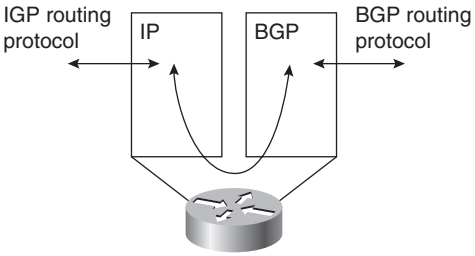
Example E-4 **show ip bgp** *Command Output with Routes Suppressed*

```
RouterC#show ip bgp
BGP table version is 28, local router ID is 172.16.2.1
Status codes: s = suppressed, * = valid, > = best, and i = internal
Origin codes : i = IGP, e = EGP, and ? = incomplete
Network              Next Hop         Metric   LocPrf     Weight     Path
*>192.168.24.0/22    0.0.0.0             0                32768     i
s>192.168.24.0       0.0.0.0             0                32768     i
s>192.168.25.0       0.0.0.0             0                32768     i
s>192.168.26.0       0.0.0.0             0                32768     i
s>192.168.27.0       0.0.0.0             0                32768     i
```

Redistribution with IGPs

Chapter 7, "Manipulating Routing Updates," discusses route redistribution and how it is configured. This section examines the specifics of when redistribution between BGP and IGPs is appropriate. As noted in Chapter 8, and as shown in Figure E-4, a router running BGP keeps a table of BGP information, separate from the IP routing table. The router can be configured to share information between the BGP table and the IP routing table.

Figure E-4 *Router Running BGP Keeps Its Own Table, Separate from the IP Routing Table*



Advertising Networks into BGP

Route information is sent from an autonomous system into BGP in one of the following ways:

- **Using the network command**—As discussed, the **network** command allows BGP to advertise a network that is already in the IP table. The list of **network** commands must include all the networks in the autonomous system you want to advertise.
- **By redistributing static routes to interface null 0 into BGP**—Redistribution occurs when a router running different protocols advertises routing information received by one protocol to the other protocol. Static routes in this case are considered a protocol, and static information is advertised to BGP (the use of the null 0 interface is discussed in the earlier section "Cautions When Using the network Command for Summarization").
- **By redistributing dynamic IGP routes into BGP**—This solution is not recommended because it might cause instability.

Redistributing from an IGP into BGP is not recommended because any change in the IGP routes—for example, if a link goes down—might cause a BGP update. This method could result in unstable BGP tables.

If redistribution is used, care must be taken that only local routes are redistributed. For example, routes learned from other autonomous systems (that were learned by redistributing BGP into the IGP) must not be sent out again from the IGP; otherwise, routing loops could result. Configuring this filtering can be complex.

Using a **redistribute** command into BGP results in an incomplete origin attribute for the route, as indicated by the **?** in the **show ip bgp** command output.

Advertising from BGP into an IGP

Route information may be sent from BGP into an autonomous system by redistributing the BGP routes into the IGP.

Because BGP is an external routing protocol, care must be taken when exchanging information with internal protocols because of the amount of information in BGP tables.

For ISP autonomous systems, redistributing from BGP normally is not required. Other autonomous systems may use redistribution, but the number of routes means that filtering normally is required. Each of these situations is examined in the following sections.

ISP: No Redistribution from BGP into IGP Is Required

An ISP typically has all routers in the autonomous system (or at least all routers in the transit path within the autonomous system) running BGP. Of course, this would be a full-mesh Internal BGP (IBGP) environment, and IBGP would be used to carry the External BGP (EBGP) routes across the autonomous system. All the BGP routers in the autonomous system would be configured with the **no synchronization** command (which is on by default in Cisco IOS Software Release 12.2(8)T and later), because synchronization between IGP and BGP is not required. The BGP information then would not need to be redistributed into the IGP. The IGP would need to route only information local to the autonomous system and routes to the next-hop addresses of the BGP routes.

One advantage of this approach is that the IGP protocol does not have to be concerned with all the BGP routes; BGP takes care of them. BGP also converges faster in this environment because it does not have to wait for the IGP to advertise the routes.

Non-ISP: Redistribution from BGP into IGP Might Be Required

A non-ISP autonomous system typically does not have all routers in the autonomous system running BGP, and it might not have a full-mesh IBGP environment. If this is the case, and if knowledge of external routes is required inside the autonomous system, redistributing BGP into the IGP is necessary. However, because of the number of routes that would be in the BGP tables, filtering normally is required.

As discussed in the “BGP Multihoming Options” section in Chapter 8, an alternative to receiving full routes from BGP is that the ISP could send only default routes, or default routes and some external routes, to the autonomous system.

NOTE An example of when redistributing into an IGP might be necessary is in an autonomous system that is running BGP only on its border routers and that has other routers in the autonomous system that do not run BGP but that require knowledge of external routes.

Policy Control and Prefix Lists

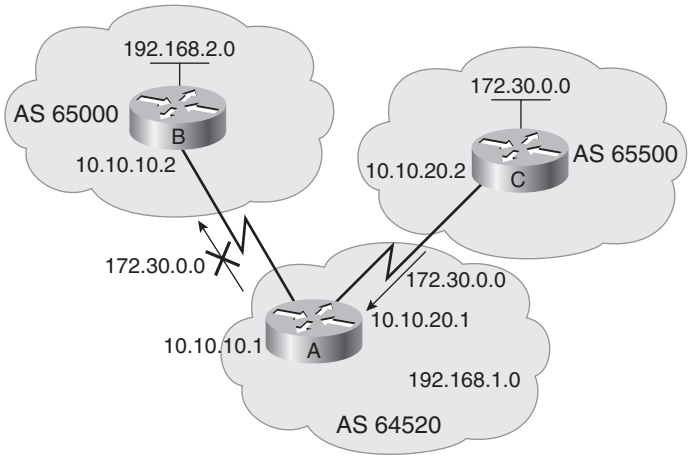
This section describes how a routing policy is applied to a BGP network using prefix lists.

If you want to restrict the routing information that the Cisco IOS Software learns or advertises, you can filter BGP routing updates to and from particular neighbors. To do this, you can define either an access list or a prefix list, and then apply it to the updates.

In earlier Cisco IOS releases, distribute lists were used; distribute lists use access lists to specify what routing information is to be filtered. Distribute lists for BGP have now been made obsolete by prefix lists in the Cisco IOS Software; prefix lists are available in Cisco IOS Release 12.0(3)T and later.

Figure E-5 shows an example where prefix lists might be used. In this figure, router C is advertising network 172.30.0.0 to router A. If you wanted to stop those updates from propagating to autonomous system 65000 (to router B), you could apply a prefix list on router A to filter those updates when router A is talking to router B.

Figure E-5 *Example Where Prefix Lists May Be Used*



Prefix List Characteristics

Access lists were originally designed to do packet filtering. Prefix lists can be used as an alternative to access lists in many BGP route filtering commands. The advantages of using prefix lists include the following:

- A significant performance improvement over access lists in loading and route lookup of large lists.
- Support for incremental modifications. Compared to a normal access list in which one **no** command erases the whole access list, prefix list entries can be modified incrementally.
- A more user-friendly command-line interface. The command-line interface for using extended access lists to filter BGP updates is difficult to understand and use.
- Greater flexibility.

Filtering with Prefix Lists

Filtering by prefix list involves matching the prefixes of routes with those listed in the prefix list, similar to using access lists.

Whether a prefix is permitted or denied is based on the following rules:

- An empty prefix list permits all prefixes.
- If a prefix is permitted, the route is used. If a prefix is denied, the route is not used.
- Prefix lists consist of statements with sequence numbers. The router begins the search for a match at the top of the prefix list, which is the statement with the lowest sequence number.
- When a match occurs, the router does not need to go through the rest of the prefix list. For efficiency, you might want to put the most common matches (permits or denies) near the top of the list by specifying a lower sequence number.
- An implicit **deny** is assumed if a given prefix does not match any entries in a prefix list.

Configuring Prefix Lists

The **ip prefix-list** {*list-name* | *list-number*} [**seq** *seq-value*] {**deny** | **permit**} *network/length* [**ge** *ge-value*] [**le** *le-value*] global configuration command is used to create a prefix list, as described in Table E-2.

Table E-2 **ip prefix-list** *Command Description*

Parameter	Description
list-name	The name of the prefix list that will be created (it is case sensitive).
list-number	The number of the prefix list that will be created.
seq-value	A 32-bit sequence number of the **prefix-list** statement, used to determine the order in which the statements are processed when filtering. Default sequence numbers are in increments of 5 (5, 10, 15, and so on).
deny | **permit**	The action taken when a match is found.
network/length	The prefix to be matched and the length of the prefix. The network is a 32-bit address; the length is a decimal number.
ge-value	The range of the prefix length to be matched for prefixes that are more specific than *network/length*. The range is assumed to be from *ge-value* to 32 if only the **ge** attribute is specified.
le-value	The range of the prefix length to be matched for prefixes that are more specific than *network/length*. The range is assumed to be from *length* to *le-value* if only the **le** attribute is specified.

The **ge** and **le** keywords are optional. They can be used to specify the range of the prefix length to be matched for prefixes that are more specific than *network/length*. The value range is

length < *ge-value* < *le-value* <= 32

An exact match is assumed when neither **ge** nor **le** is specified.

Prefix list entries can be reconfigured incrementally. In other words, an entry can be deleted or added individually.

The **neighbor** {*ip-address* | *peer-group-name*} **prefix-list** *prefix-listname* {**in** | **out**} router configuration command is used to filter BGP neighbor information as specified in a prefix list, as described in Table E-3.

Table E-3 **neighbor prefix-list** *Command Description*

Parameter	Description
ip-address	The IP address of the BGP neighbor for which routes will be filtered
peer-group-name	The name of a BGP peer group
prefix-listname	The name of the prefix list that will be used to filter the routes

Table E-3 **neighbor prefix-list** *Command Description (Continued)*

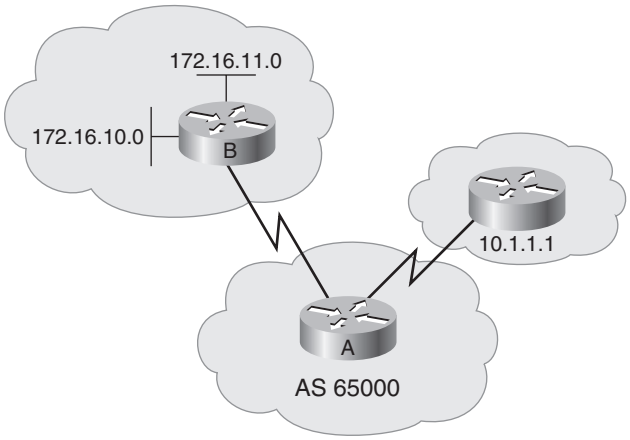
Parameter	Description
in	Indicates that the prefix list is to be applied to incoming advertisements from the neighbor
out	Indicates that the prefix list is to be applied to outgoing advertisements to the neighbor

ip prefix-list Command Options

The use of the **ge** and **le** options in the **ip prefix-list** command can be confusing. The following are results of some testing done to understand these keywords.

Three routers were used in this testing: router B, router A, and its neighbor 10.1.1.1, as illustrated in Figure E-6.

Figure E-6 *Network Used in Prefix List Option Testing*



Before configuring the prefix list, router A learns the following routes (from router B):

```
172.16.0.0 subnetted:
    172.16.10.0/24
    172.16.11.0/24
```

Five scenarios were tested:

Scenario 1—In this scenario, the following is configured on router A:

```
router bgp 65000
    aggregate-address 172.16.0.0 255.255.0.0
    neighbor 10.1.1.1 prefix-list tenonly out
ip prefix-list tenonly permit 172.16.10.0/8 le 24
```

When you view the router's configuration with the **show run** command, you see that the router automatically changes the last line of this configuration to the following:

```
ip prefix-list tenonly permit 172.0.0.0/8 le 24
```

Neighbor 10.1.1.1 learns about 172.16.0.0/16, 172.16.10.0/24, and 172.16.11.0/24.

Scenario 2—In this scenario, the following is configured on router A:

```
router bgp 65000
    aggregate-address 172.16.0.0 255.255.0.0
    neighbor 10.1.1.1 prefix-list tenonly out
ip prefix-list tenonly permit 172.0.0.0/8 le 16
```

Neighbor 10.1.1.1 learns only about 172.16.0.0/16.

Scenario 3—In this scenario, the following is configured on router A:

```
router bgp 65000
    aggregate-address 172.16.0.0 255.255.0.0
    neighbor 10.1.1.1 prefix-list tenonly out
ip prefix-list tenonly permit 172.0.0.0/8 ge 17
```

Neighbor 10.1.1.1 learns only about 172.16.10.0/24 and 172.16.11.0/24 (in other words, it ignores the **/8** parameter and treats the command as if it has the parameters **ge 17 le 32**).

Scenario 4—In this scenario, the following is configured on router A:

```
router bgp 65000
    aggregate-address 172.16.0.0 255.255.0.0
    neighbor 10.1.1.1 prefix-list tenonly out
ip prefix-list tenonly permit 172.0.0.0/8 ge 16 le 24
```

Neighbor 10.1.1.1 learns about 172.16.0.0/16, 172.16.10.0/24, and 172.16.11.0/24 (in other words, it ignores the **/8** parameter and treats the command as if it has the parameters **ge 16 le 24**).

Scenario 5—In this scenario, the following is configured on router A:

```
router bgp 65000
    aggregate-address 172.16.0.0 255.255.0.0
    neighbor 10.1.1.1 prefix-list tenonly out
ip prefix-list tenonly permit 172.0.0.0/8 ge 17 le 24
```

Neighbor 10.1.1.1 learns about 172.16.10.0/24 and 172.16.11.0/24 (in other words, it ignores the **/8** parameter and treats the command as if it has the parameters **ge 17 le 24**).

The **no ip prefix-list** *list-name* global configuration command, where *list-name* is the name of a prefix list, is used to delete a prefix list.

The [**no**] **ip prefix-list** *list-name* **description** *text* global configuration command can be used to add or delete a text description for a prefix list.

Prefix List Sequence Numbers

Prefix list sequence numbers are generated automatically, unless you disable this automatic generation. If you do so, you must specify the sequence number for each entry using the *seq-value* argument of the **ip prefix-list** command.

A prefix list is an ordered list. The sequence number is significant when a given prefix is matched by multiple entries of a prefix list, in which case the one with the smallest sequence number is considered the real match.

Regardless of whether you use the default sequence numbers to configure a prefix list, you don't need to specify a sequence number when removing a configuration entry.

By default, a prefix list's entries have sequence values of 5, 10, 15, and so on. In the absence of a specified sequence value, a new entry is assigned a sequence number equal to the current maximum sequence number plus 5.

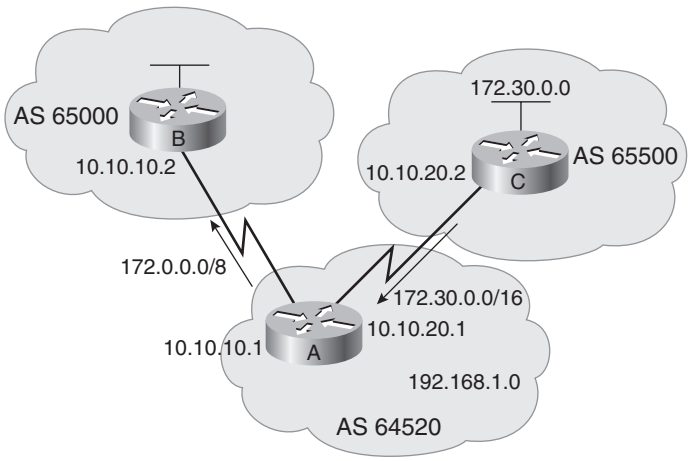
Prefix list **show** commands include the sequence numbers in their output.

The **no ip prefix-list sequence-number** global configuration command is used to disable the automatic generation of sequence numbers of prefix list entries. Use the **ip prefix-list sequence-number** global configuration command to reenable the automatic generation of sequence numbers.

Prefix List Example

The sample network shown in Figure E-7 illustrates the use of a prefix list. In this example, you want router A to send only the supernet 172.0.0.0/8 to autonomous system 65000; the route to the network 172.30.0.0/16 should not be sent. Example E-5 shows the configuration for router A.

Figure E-7 *Prefix List Example*



Example E-5 *Configuration of Router A in Figure E-7*

```
RtrA(config)#ip prefix-list superonly permit 172.0.0.0/8
RtrA(config)#ip prefix-list superonly description only permit supernet
RtrA(config)#router bgp 64520
RtrA(config-router)#network 192.168.1.0
RtrA(config-router)#neighbor 10.10.10.2 remote-as 65000
RtrA(config-router)#neighbor 10.10.20.2 remote-as 65500
RtrA(config-router)#aggregate-address 172.0.0.0 255.0.0.0
RtrA(config-router)#neighbor 10.10.10.2 prefix-list superonly out
RtrA(config-router)#exit
```

In this example, router A has two neighbors: router B (10.10.10.2 in autonomous system 65000) and router C (10.10.20.2 in autonomous system 65500). When router A sends updates to neighbor router B, the **neighbor prefix-list** statement specifies that it will use the prefix list called superonly to determine which updates are to be sent.

The **ip prefix-list superonly** command specifies that only the route 172.0.0.0/8 should be sent (it is permitted in the prefix list). No other routes are sent to router B, because prefix lists have an implicit **deny** any at the end.

Verifying Prefix Lists

The EXEC commands related to prefix lists are described in Table E-4. Use the **show ip prefix-list ?** command to see all of the **show** commands available for prefix lists.

Table E-4 *Commands Used to Verify Prefix Lists*

Command	Description
show ip prefix-list [**detail** | **summary**]	Displays information on all prefix lists. Specifying the **detail** keyword includes the description and the hit count (the number of times the entry matches a route) in the display.
show ip prefix-list [**detail** | **summary**] *prefix-list-name*	Displays a table showing the entries in a specific prefix list.
show ip prefix-list *prefix-list-name* [*network/length*]	Displays the policy associated with a specific *network/length* in a prefix list.
show ip prefix-list *prefix-list-name* [**seq** *sequence-number*]	Displays the prefix list entry with a given sequence number.
show ip prefix-list *prefix-list-name* [*network/length*] **longer**	Displays all entries of a prefix list that are more specific than the given network and length.

Table E-4 *Commands Used to Verify Prefix Lists (Continued)*

Command	Description
show ip prefix-list *prefix-list-name* [*network/length*] **first-match**	Displays the entry of a prefix list that matches the network and length of the given prefix.
clear ip prefix-list *prefix-list-name* [*network/length*]	Resets the hit count shown on prefix list entries.

Verifying Prefix Lists Example

The sample output of the **show ip prefix-list detail** command shown in Example E-6 is from router A in Figure E-7. Router A has a prefix list called superonly that has only one entry (sequence number 5). The hit count of 0 means that no routes match this entry.

Example E-6 **show ip prefix-list detail** *Command Output from Router A in Figure E-7*

```
RtrA #show ip prefix-list detail
Prefix-list with the last deletion/insertion: superonly ip prefix-list superonly:
   Description: only permit supernet
   count: 1, range entries: 0, sequences: 5 - 5, refcount: 1
seq 5 permit 172.0.0.0/8 (hit count: 0, refcount: 1)
```

Communities

As discussed in Chapter 8, BGP communities are another way to filter incoming or outgoing BGP routes. Distribute lists and prefix lists are cumbersome to configure for a large network with a complex routing policy. For example, individual neighbor statements and access lists or prefix lists have to be configured for each neighbor on each router involved in the policy.

The BGP communities function allows routers to tag routes with an indicator (the *community*) and allows other routers to make decisions (filter) based on that tag. BGP communities are used for destinations (routes) that share some common properties and that, therefore, share common policies; routers, therefore, act on the community, rather than on individual routes. Communities are not restricted to one network or autonomous system, and they have no physical boundaries.

Community Attribute

The community attribute is an optional transitive attribute. If a router does not understand the concept of communities, it passes it on to the next router. However, if the router does understand the concept, it must be configured to propagate the community; otherwise, communities are dropped by default.

Each network can be a member of more than one community.

The community attribute is a 32-bit number; it can have a value in the range 0 to 4,294,967,200. The upper 16 bits indicate the autonomous system number of the autonomous system that defined the community. The lower 16 bits are the community number and have local significance. The community value can be entered as one decimal number or in the format *AS*:*nn*, where *AS* is the autonomous system number, and *nn* is the lower 16-bit local number. The community value is displayed as one decimal number by default.

Setting and Sending the Communities Configuration

Route maps can be used to set the community attributes.

The **set community** {[*community-number*] [*well-known-community*] [**additive**]} | **none** route map configuration command is used within a route map to set the BGP community attribute, as described in Table E-5.

Table E-5 **set community** *Command Description*

Parameter	Description
community-number	The community number. Values are 1 to 4,294,967,200.
well-known-community	The following are predefined, well-known community numbers: • **internet**—Advertises this route to the Internet community and any router that belongs to it • **no-export**—Does not advertise to EBGP peers • **no-advertise**—Does not advertise this route to any peer • **local-AS**—Does not send outside the local autonomous system
additive	(Optional) Specifies that the community is to be added to the existing communities.
none	Removes the community attribute from the prefixes that pass the route map.

The **set community** command is used along with the **neighbor route-map** command to apply the route map to updates.

The **neighbor** {*ip-address* | *peer-group-name*} **send-community** router configuration command is used to specify that the BGP communities attribute should be sent to a BGP neighbor. Table E-6 explains this command.

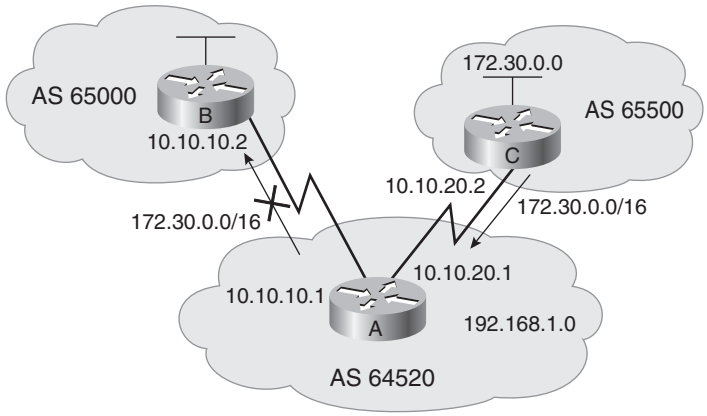
Table E-6 **neighbor send-community** *Command Description*

Parameter	Description
ip-address	The IP address of the BGP neighbor to which the communities attribute is sent.
peer-group-name	The name of a BGP peer group.

By default, the communities attribute is not sent to any neighbor (communities are stripped in outgoing BGP updates).

In Figure E-8, router C is sending BGP updates to router A, but it does not want router A to propagate these routes to router B.

Figure E-8 *Network for BGP Communities Example*



Example E-7 shows the configuration for router C in this example. Router C sets the community attribute in the BGP routes that it is advertising to router A. The **no-export** community attribute is used to indicate that router A should not send the routes to its external BGP peers.

Example E-7 *Configuration of Router C in Figure E-8*

```
router bgp 65500
  network 172.30.0.0
  neighbor 10.10.20.1 remote-as 64520
  neighbor 10.10.20.1 send-community
  neighbor 10.10.20.1 route-map SETCOMM out
```

continues

Example E-7 *Configuration of Router C in Figure E-8 (Continued)*

```
!
route-map SETCOMM permit 10
  match ip address 1
  set community no-export
!
access-list 1 permit 0.0.0.0 255.255.255.255
```

In this example, router C has one neighbor, 10.10.20.1 (router A). When communicating with router A, the community attribute is sent, as specified by the **neighbor send-community** command. The route map SETCOMM is used when sending routes to router A to set the community attribute. Any route that matches **access-list 1** has the community attribute set to **no-export**. Access list 1 permits any routes; therefore, all routes have the community attribute set to **no-export**.

In this example, router A receives all of router C's routes but does not pass them to router B.

Using the Communities Configuration

The **ip community-list** *community-list-number* {**permit** | **deny**} *community-number* global configuration command is used to create a community list for BGP and to control access to it, as described in Table E-7.

Table E-7 **ip community-list** *Command Description*

Parameter	Description
community-list-number	The community list number, in the range of 1 to 99
permit | **deny**	Permits or denies access for a matching condition.
community-number	The community number or well-known-community configured by a **set community** command

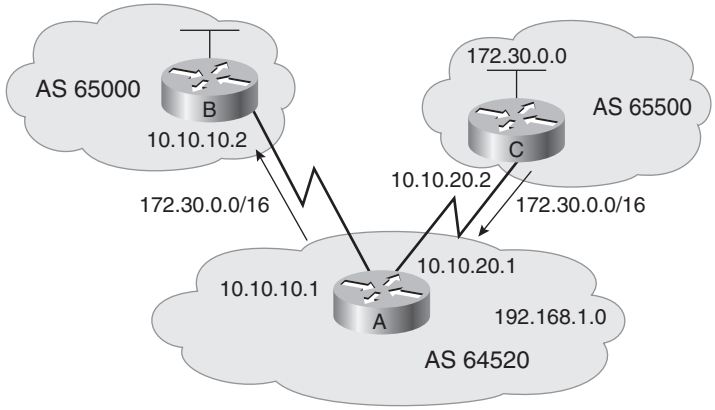
The **match community** *community-list-number* [**exact**] route map configuration command enables you to match a BGP community attribute to a value in a community list, as described in Table E-8.

Table E-8 **match community** *Command Description*

Parameter	Description
community-list-number	The community list number, in the range of 1 to 99, that is used to compare the community attribute.
exact	(Optional) Indicates that an exact match is required. All the communities and only those communities in the community list must be present in the community attribute.

In Figure E-9, router C is sending BGP updates to router A. Router A sets the weight of these routes based on the community value set by router C.

Figure E-9 *Network for BGP Communities Example Using Weight*



Example E-8 shows the configuration of router C in Figure E-9. Router C has one neighbor, 10.10.20.1 (router A).

Example E-8 *Configuration of Router C in Figure E-9*

```
router bgp 65500
  network 172.30.0.0
  neighbor 10.10.20.1 remote-as 64520
  neighbor 10.10.20.1 send-community
  neighbor 10.10.20.1 route-map SETCOMM out
!
route-map SETCOMM permit 10
  match ip address 1
  set community 100 additive
!
access-list 1 permit 0.0.0.0 255.255.255.255
```

In this example, the community attribute is sent to router A, as specified by the **neighbor send-community** command. The route map SETCOMM is used when sending routes to router A to set the community attribute. Any route that matches access list 1 has community 100 added to the existing communities in the route's community attribute. In this example, access list 1 permits any routes; therefore, all routes have 100 added to the list of communities. If the **additive** keyword in the **set community** command is not set, 100 replaces any old community that already exits. Because the keyword **additive** is used, the 100 is added to the list of communities that the route is part of.

Example E-9 shows the configuration of router A in Figure E-9.

Example E-9 *Configuration of Router A in Figure E-9*

```
router bgp 64520
  neighbor 10.10.20.2 remote-as 65500
  neighbor 10.10.20.2 route-map CHKCOMM in
!
route-map CHKCOMM permit 10
  match community 1
  set weight 20
route-map CHKCOMM permit 20
  match community 2
!
ip community-list 1 permit 100
ip community-list 2 permit internet
```

NOTE Other **router bgp** configuration commands for router A are not shown in Example E-9.

In this example, router A has a neighbor, 10.10.20.2 (router C). The route map CHKCOMM is used when receiving routes from router C to check the community attribute. Any route whose community attribute matches community list 1 has its weight attribute set to 20. Community list 1 permits routes with a community attribute of 100; therefore, all routes from router C (which all have 100 in their list of communities) have their weight set to 20.

In this example, any route that does not match community list 1 is checked against community list 2. Any route matching community list 2 is permitted but does not have any of its attributes changed. Community list 2 specifies the **internet** keyword, which means all routes.

The sample output shown in Example E-10 is from router A in Figure E-9. The output shows the details about the route 172.30.0.0 from router C, including that its community attribute is 100, and its weight attribute is now 20.

Example E-10 *Output from Router A in Figure E-9*

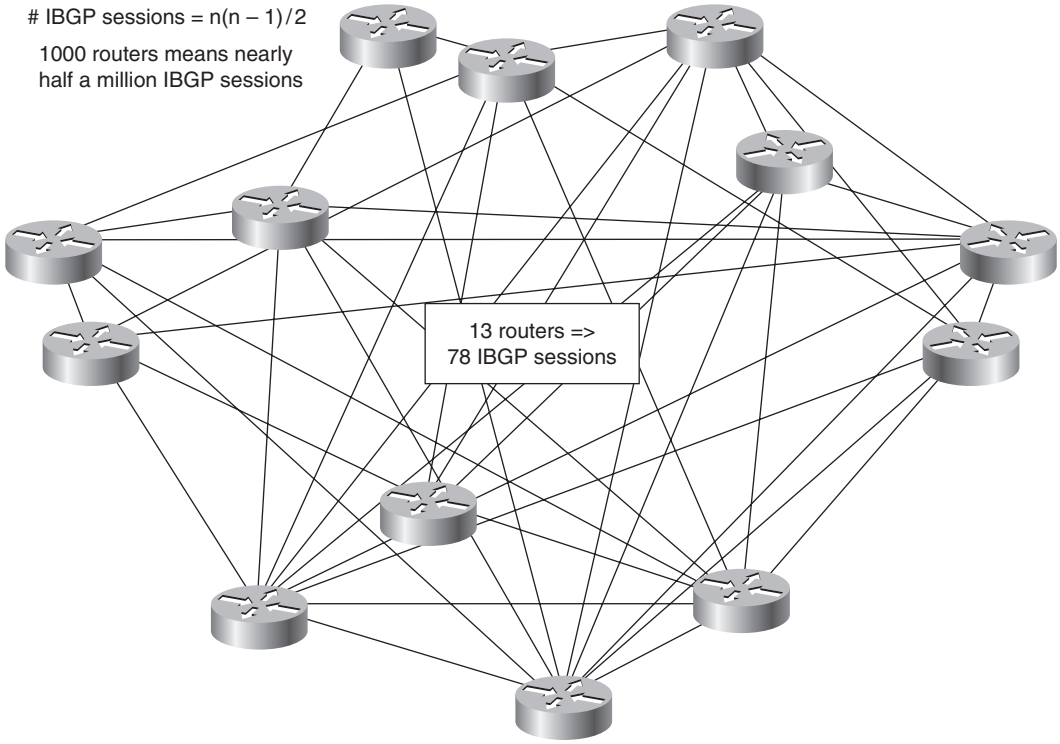
```
RtrA #show ip bgp 172.30.0.0/16
BGP routing table entry for 172.30.0.0/16, version 2
Paths: (1 available, best #1)
  Advertised to non peer-group peers:
    10.10.10.2
  65500
    10.10.20.2 from 10.10.20.2 (172.30.0.1)
      Origin IGP, metric 0, localpref 100, weight 20, valid, external, best, ref 2
 Community: 100
```

Route Reflectors

BGP specifies that routes learned via IBGP are never propagated to other IBGP peers. The result is that a full mesh of IBGP peers is required within an autonomous system. As Figure E-10 illustrates, however, a full mesh of IBGP is not scalable. With only 13 routers, 78 IBGP sessions would need to be maintained. As the number of routers increases, so does the number of sessions required, governed by the following formula, in which *n* is the number of routers:

$$\text{\# of IBGP sessions} = n(n-1)/2$$

Figure E-10 *Full-Mesh IBGP Requires Many Sessions and, Therefore, Is Not Scalable*

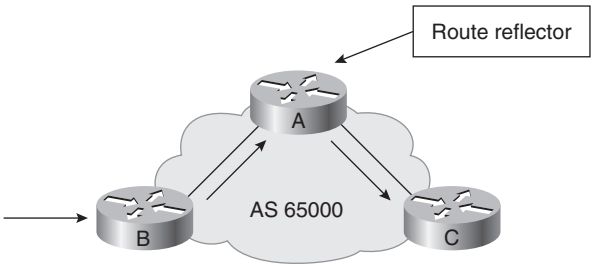


In addition to the number of BGP Transmission Control Protocol (TCP) sessions that must be created and maintained, the amount of routing traffic also might be a problem. Depending on the autonomous system topology, traffic might be replicated many times on some links as it travels to each IBGP peer. For example, if the physical topology of a large autonomous system includes some WAN links, the IBGP sessions running over those links might consume a significant amount of bandwidth.

A solution to this problem is the use of route reflectors (RRs). This section describes what an RR is, how it works, and how to configure it.

RRs modify the BGP rule by allowing the router configured as the RR to propagate routes learned by IBGP to other IBGP peers, as illustrated in Figure E-11.

Figure E-11 *When Router A Is a Route Reflector, It Can Propagate Routes Learned from Router B to Router C*



This saves on the number of BGP TCP sessions that must be maintained and also reduces the BGP routing traffic.

Route Reflector Benefits

With a BGP RR configured, a full mesh of IBGP peers is no longer required. The RR is allowed to propagate IBGP routes to other IBGP peers. RRs are used mainly by ISPs when the number of internal **neighbor** statements becomes excessive. Route reflectors reduce the number of BGP neighbor relationships in an autonomous system (thus, saving on TCP connections) by having key routers replicate updates to their RR clients.

Route reflectors do not affect the paths that IP packets follow; only the path that routing information is distributed on is affected. However, if RRs are configured incorrectly, routing loops might result, as shown in the example later in this appendix in the "Route Reflector Migration Tips" section.

An autonomous system can have multiple RRs, both for redundancy and for grouping to further reduce the number of IBGP sessions required.

Migrating to RRs involves a minimal configuration and does not have to be done all at one time, because routers that are not RRs can coexist with RRs within an autonomous system.

Route Reflector Terminology

A *route reflector* is a router that is configured to be the router allowed to advertise (or reflect) routes it learned via IBGP to other IBGP peers. The RR has a partial IBGP peering with other routers, which are called *clients*. Peering between the clients is not needed, because the route reflector passes advertisements between the clients.

The combination of the RR and its clients is called a *cluster.*

Other IBGP peers of the RR that are not clients are called *nonclients*.

The *originator ID* is an optional, nontransitive BGP attribute that is created by the RR. This attribute carries the router ID of the route's originator in the local autonomous system. If the update comes back to the originator because of poor configuration, the originator ignores it.

Usually a cluster has a single RR, in which case the cluster is identified by the RR's router ID. To increase redundancy and avoid single points of failure, a cluster might have more than one RR. When this occurs, all of the RRs in the cluster need to be configured with a *cluster ID*. The cluster ID allows route reflectors to recognize updates from other RRs in the same cluster.

A *cluster list* is a sequence of cluster IDs that the route has passed. When an RR reflects a route from its clients to nonclients outside the cluster, it appends the local cluster ID to the cluster list. If the update has an empty cluster list, the RR creates one. Using this attribute, an RR can tell whether the routing information is looped back to the same cluster because of poor configuration. If the local cluster ID is found in an advertisement's cluster list, the advertisement is ignored.

The originator ID, cluster ID, and cluster list help prevent routing loops in RR configurations.

Route Reflector Design

When using RRs in an autonomous system, you can divide the autonomous system into multiple clusters, each having at least one RR and a few clients. Multiple route reflectors can exist in one cluster for redundancy.

The RRs must be fully meshed with IBGP to ensure that all routes learned are propagated throughout the autonomous system.

An IGP is still used, just as it was before RRs were introduced, to carry local routes and next-hop addresses.

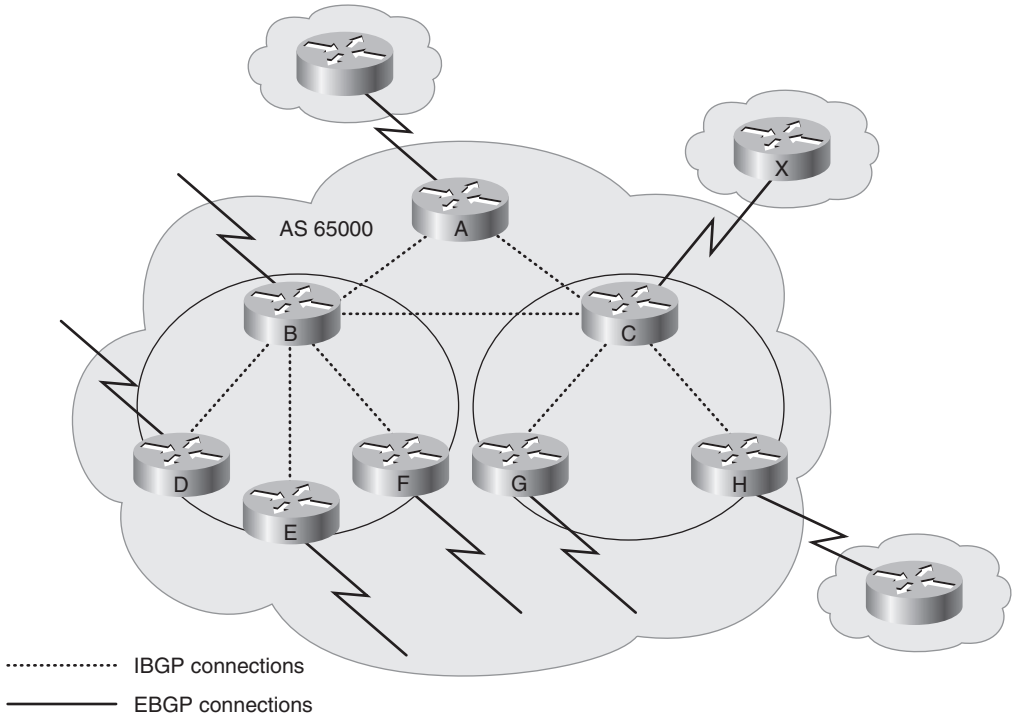
Split-horizon rules still apply between an RR and its clients; thus an RR that receives a route from a client does not advertise that route back to that client.

NOTE No defined limit applies to the number of clients an RR might have; it is constrained by the amount of router memory.

Route Reflector Design Example

Figure E-12 provides an example of a BGP RR design.

Figure E-12 *Example of a Route Reflector Design*



NOTE The physical connections within autonomous system 65000 are not shown in Figure E-12.

In Figure E-12, routers B, D, E, and F form one cluster. Routers C, G, and H form another cluster. Routers B and C are RRs. Routers A, B, and C are fully meshed with IBGP. Note that the routers within a cluster are not fully meshed.

Route Reflector Operation

When an RR receives an update, it takes the following actions, depending on the type of peer that sent the update:

- If the update is from a client peer, it sends the update to all nonclient peers and to all client peers (except the route's originator).
- If the update is from a nonclient peer, it sends the update to all clients in the cluster.
- If the update is from an EBGP peer, it sends the update to all nonclient peers and to all client peers.

For example, in Figure E-12, the following happens:

- If router C receives an update from router H (a client), it sends it to router G, and to routers A and B.
- If router C receives an update from router A (a nonclient), it sends it to routers G and H.
- If router C receives an update from router X (via EBGP), it sends it to routers G and H, and to routers A and B.

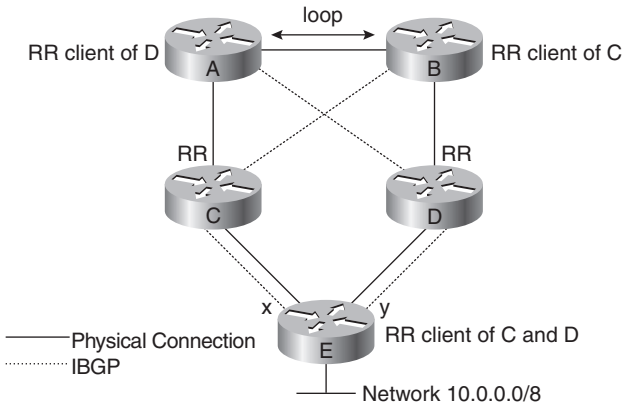
NOTE Routers also send updates to their EBGP neighbors as appropriate.

Route Reflector Migration Tips

When migrating to using RRs, the first consideration is which routers should be the reflectors and which should be the clients. Following the physical topology in this design decision ensures that the packet-forwarding paths are not affected. Not following the physical topology (for example, configuring RR clients that are not physically connected to the route reflector) might result in routing loops.

Figure E-13 demonstrates what can happen if RRs are configured without following the physical topology. In this figure, the lower router, router E, is an RR client for both RRs, routers C and D.

Figure E-13 *Bad Route Reflector Design*

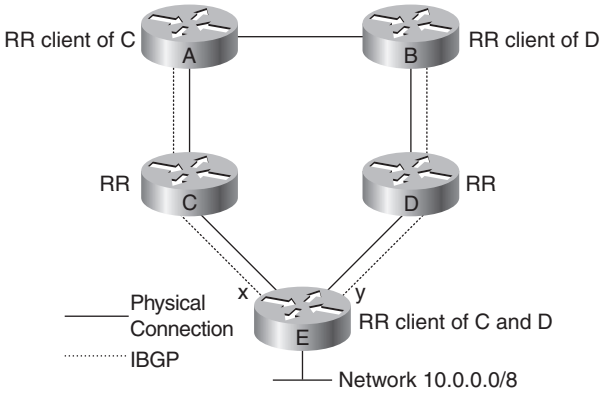


In this *bad design*, which does not follow the physical topology, the following happens:

- Router B knows that the next hop to get to 10.0.0.0 is *x* (because it learns this from its RR, router C).
- Router A knows that the next hop to get to 10.0.0.0 is *y* (because it learns this from its RR, router D).
- For router B to get to *x*, the best route might be through router A, so router B sends a packet destined for 10.0.0.0 to router A.
- For router A to get to *y*, the best route might be through router B, so router A sends a packet destined for 10.0.0.0 to router B.
- This is a routing loop.

Figure E-14 shows a better design (better because it follows the physical topology). Again, in this figure, the lower router, router E, is an RR client for both route reflectors.

Figure E-14 *Good Route Reflector Design*



In this *good design*, which follows the physical topology, the following are true:

- Router B knows that the next hop to get to 10.0.0.0 is *y* (because it learns this from its RR, router D).
- Router A knows that the next hop to get to 10.0.0.0 is *x* (because it learns this from its RR, router C).
- For router A to get to *x*, the best route is through router C, so router A sends a packet destined for 10.0.0.0 to router C, and router C sends it to router E.
- For router B to get to *y*, the best route is through router D, so router B sends a packet destined for 10.0.0.0 to router D, and router D sends it to router E.
- There is no routing loop.

When migrating to using RRs, configure one RR at a time, and then delete the redundant IBGP sessions between the clients. It is recommended that you configure one RR per cluster.

Route Reflector Configuration

The **neighbor** *ip-address* **route-reflector-client** router configuration command enables you to configure the router as a BGP RR and to configure the specified neighbor as its client. Table E-9 explains this command.

Table E-9 **neighbor route-reflector-client** *Command Description*

Parameter	Description
ip-address	The IP address of the BGP neighbor being identified as a client

Configuring the Cluster ID

To configure the cluster ID if the BGP cluster has more than one RR, use the **bgp cluster-id** *cluster-id* router configuration command on all the RRs in a cluster. You cannot change the cluster ID after the RR clients have been configured.

RRs cause some restrictions on other commands, including the following:

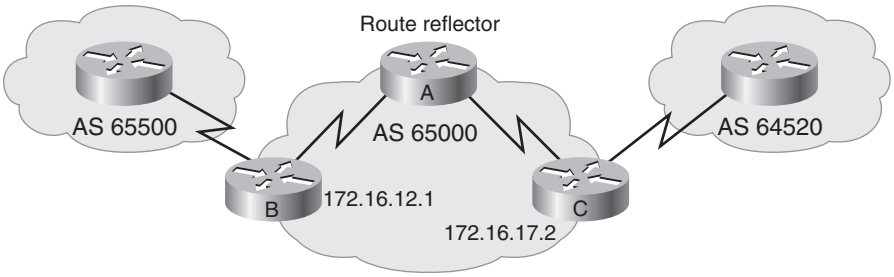
- When used on RRs, the **neighbor next-hop-self** command affects only the next hop of EBGP learned routes, because the next hop of reflected IBGP routes should not be changed.

- RR clients are incompatible with peer groups. This is because a router configured with a peer group must send any update to *all* members of the peer group. If an RR has all of its clients in a peer group and then one of those clients sends an update, the RR is responsible for sharing that update with all *other* clients. The RR must not send the update to the originating client, because of the split-horizon rule.

Route Reflector Example

The network in Figure E-15 illustrates a router configured as an RR in autonomous system 65000. Example E-11 shows the configuration for router A in this figure.

Figure E-15 *Router A Is a Route Reflector*



Example E-11 *Configuration of Router A in Figure E-15*

```
RTRA(config)#router bgp 65000
RTRA(config-router)#neighbor 172.16.12.1 remote-as 65000
RTRA(config-router)#neighbor 172.16.12.1 route-reflector-client
RTRA(config-router)#neighbor 172.16.17.2 remote-as 65000
RTRA(config-router)#neighbor 172.16.17.2 route-reflector-client
```

The **neighbor route-reflector-client** commands enable you to configure which neighbors are RR clients. In this example, both routers B and C are RR clients of router A, the RR.

Verifying Route Reflectors

The **show ip bgp neighbors** command indicates that a particular neighbor is an RR client. The sample output for this command, shown in Example E-12, is from router A in Figure E-15 and shows that 172.16.12.1 (router B) is an RR client of router A.

Example E-12 **show ip bgp neighbors** *Output from Router A in Figure E-15*

```
RTRA#show ip bgp neighbors
BGP neighbor is 172.16.12.1,  remote AS 65000, internal link
 Index 1, Offset 0, Mask 0x2
  Route-Reflector Client
  BGP version 4, remote router ID 192.168.101.101
  BGP state = Established, table version = 1, up for 00:05:42
  Last read 00:00:42, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 5 seconds
  Received 14 messages, 0 notifications, 0 in queue
  Sent 12 messages, 0 notifications, 0 in queue
  Prefix advertised 0, suppressed 0, withdrawn 0
  Connections established 2; dropped 1
  Last reset 00:05:44, due to User reset
  1 accepted prefixes consume 32 bytes
  0 history paths consume 0 bytes
--More--
```