CHAPTER 14

Route Maps

Route maps are similar to access lists; they both have criteria for matching the details of certain packets and an action of permitting or denying those packets. Unlike access lists, though, route maps can add to each "match" criterion a "set" criterion that actually changes the packet in a specified manner, or changes route information in a specified manner. Route maps also allow you more options for matching a given packet. Altogether, route maps are a powerful tool for creating customized routing policies in your network.

Basic Uses of Route Maps

Route maps can be used for both redistribution and for policy routing. Although redistribution has been covered extensively in previous chapters, this chapter introduces the topic of policy routing. And most commonly, they are an essential tool in large-scale Border Gateway Protocol (BGP) implementations. The use of route maps to implement BGP routing policies is outside the scope of this volume, and is covered in "Routing TCP/IP," Volume II (CCIE Professional Development) (1-57870-089-2).

Policy routes are nothing more than sophisticated static routes. Whereas static routes forward a packet to a specified next hop based on the destination address of the packet, policy routes can forward a packet to a specified next hop based on the source of the packet or other fields in the packet header. Policy routes can also be linked to extended IP access lists so that routing might be based on such things as protocol types and port numbers. Like a static route, a policy route influences the routing only of the router on which it is configured.

Figure 14-1 shows an example of a typical policy routing application. AbnerNet is connected to two Internet service providers via router Dogpatch. AbnerNet's corporate policy dictates that some users' Internet traffic should be sent via ISP 1 and other users' Internet traffic should be sent via ISP 2. If either ISP should become unavailable, the traffic normally using that provider would be sent to the other provider. A policy route at Dogpatch can distribute Internet traffic in accordance with local policy. The distribution of traffic might be based on subnet, specific user, or even user applications.

Figure 14-1 *Policy routing allows traffic from AbnerNet to be routed to one of its two Internet service providers based on parameters such as source address, source/destination address combinations, packet size, or application-level ports.*

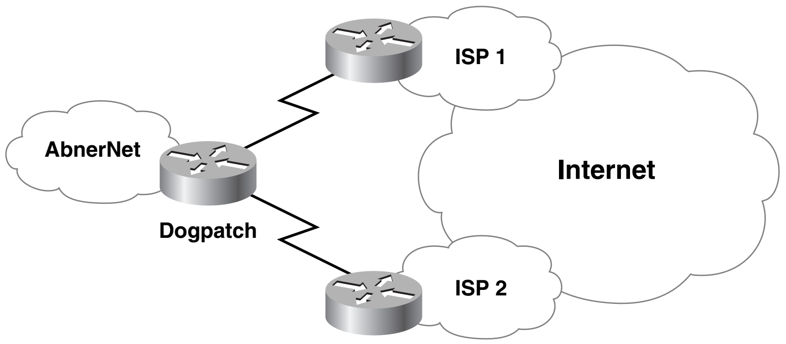
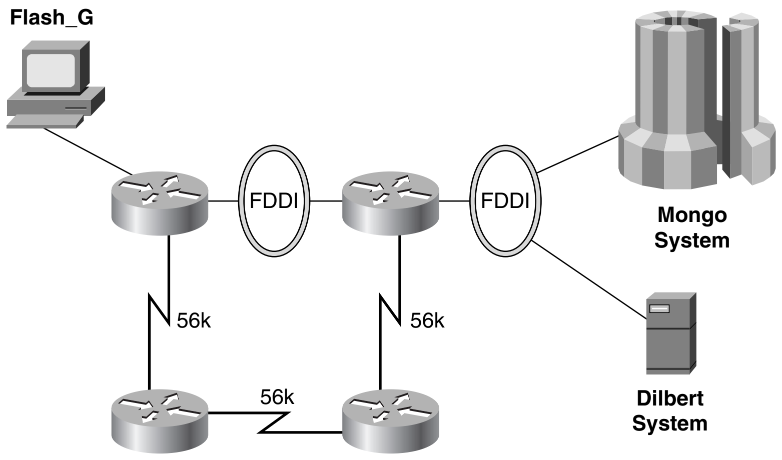


Figure 14-2 shows another use for policy routing. One of the systems on the right watches for invasion forces from the planet Mongo, while the other stores back copies of *Dilbert* comic strips. Policy routes can be configured to route the critical traffic from the Mongo System to Flash_G over the FDDI (Fiber Distributed Data Interface) link and to route the lower-priority Dilbert traffic over the 56K links—or vice versa.

Figure 14-2 *Policy routing allows high-priority traffic from the Mongo System to be routed over the FDDI link while low-priority traffic from the Dilbert System is routed over the 56K links.*



Tables 14-1 and 14-2 show the **match** and **set** commands that can be used with redistribution, and Tables 14-3 and 14-4 show the **match** and **set** commands that can be used with policy routing.

Table 14-1 **match** *commands that can be used with redistribution.*

Command	Description
match interface *type number* [...*type number*]	Matches routes that have their next hop out one of the interfaces specified
match ip address {*access-list-number* | *name*} [...*access-list-number* | *name*]	Matches routes that have a destination address specified by one of the access lists
match ip next-hop {*access-list-number* | *name*} [...*access-list-number* | *name*]	Matches routes that have a next-hop router address specified by one of the access lists
match ip route-source {*access-list-number* | *name*} [...*access-list-number* | *name*]	Matches routes that have been advertised by routers at the addresses specified in the access lists
match metric *metric-value*	Matches routes with the specified metric
match route-type {**internal** | **external** [**type-1** | **type-2**] | **level-1** | **level-2**}	Matches OSPF, EIGRP, or IS-IS routes of the specified type
match tag *tag-value* [...*tag-value*]	Matches routes with the specified tags

Table 14-2 **set** *commands that can be used with redistribution.*

Command	Description
set level {**level-1** | **level-2** | **level-1-2** | **stub-area** | **backbone**}	Sets the IS-IS level or the OSPF area into which a matched route is to be redistributed
set metric {*metric-value* | *bandwidth delay reliability loading mtu*}	Sets the metric value for a matched route
set metric-type {**internal** | **external** | **type-1** | **type-2**}	Sets the metric type for a matched route being redistributed into IS-IS or OSPF
set next-hop *next-hop*	Sets the next-hop router address for a matched route
set tag *tag-value*	Sets a tag value for a matched route

Table 14-3 **match** *commands that can be used with policy routing.*

Command	Description
match ip address {*access-list-number* | *name*} [...*access-list-number* | *name*]	Matches a packet with the characteristics specified in the standard or extended access lists
match length *min max*	Matches the Layer 3 length of a packet

Table 14-4 **set** *commands that can be used with policy routing.*

Command	Description
set default interface *type number* [...*type number*]	Sets the outgoing interface for matched packets when there is no explicit route to the destination
set interface *type number* [...*type number*]	Sets the outgoing interface for matched packets when there is an explicit route to the destination
set ip default next-hop *ip-address* [...*ip-address*]	Sets the next-hop router address for matched packets when there is no explicit route to the destination
set ip next-hop *ip-address* [...*ip-address*]	Sets the next-hop router address for matched packets when there is an explicit route to the destination
set ip precedence *precedence*	Sets the precedence bits in the Type of Service field of matched IP packets
set ip tos *type-of-service*	Sets the TOS bits in the Type of Service field of matched packets

Configuring Route Maps

Like access lists (see Appendix B, "Tutorial: Access Lists"), route maps by themselves affect nothing; they must be "called" by some command. The command will most likely be either a policy routing command or a redistribution command. Policy routing will send packets to the route map, whereas redistribution will send routes to the route map. The case studies in this section demonstrate the use of route maps for both redistribution and policy routing.

Route maps are identified by a name. For example, the route map in Example 14-1 is named Hagar.

Example 14-1 *A route map named Hagar is defined in this configuration.*

```
route-map Hagar permit 10
match ip address 110
set metric 100
```

Each route map statement has a "permit" or "deny" action and a sequence number. This route map shows a permit action and a sequence number of 10. These settings are the defaults—that is, if no action or sequence number is specified when the route map is configured, the route map will default to a permit and a sequence number of 10.

The sequence number allows the identification and editing of multiple statements. Consider the configuration steps in Example 14-2.

Example 14-2 *Route map Hagar is modified in this configuration.*

```
route-map Hagar 20
 match ip address 111
 set metric 50
route-map Hagar 15
 match ip address 112
 set metric 80
```

Here, a second and third set of route map statements, each with their own **match** and **set** statements, have been added to route map Hagar. Notice that a sequence number of 20 was configured first and then a sequence number of 15. In the final configuration, the IOS has placed statement 15 before 20 even though it was entered later, as shown in Example 14-3.[1]

Example 14-3 *IOS places the commands in sequential order.*

```
route-map Hagar permit 10
 match ip address 110
 set metric 100
!
route-map Hagar permit 15
 match ip address 112
 set metric 80
!
route-map Hagar permit 20
 match ip address 111
 set metric 50
```

The sequence numbers also allow for the elimination of individual statements. For example, the statement

```
Linus(config)#no route-map Hagar 15
```

deletes statement 15 and leaves the other statements intact, as shown in Example 14-4.

Example 14-4 *Route-map Hagar after the match/set statements associated with sequence 15 have been removed.*

```
route-map Hagar permit 10
 match ip address 110
 set metric 100
!
route-map Hagar permit 20
 match ip address 111
 set metric 50
```

Be careful when editing route maps. In this example, if **no route-map Hagar** had been typed, without specifying a sequence number, the entire route map would have been deleted. Likewise, if no sequence numbers had been specified when the additional **match** and **set** statements were added, they would have simply changed statement 10.[2]

[1] Notice also that no action was specified in the configuration steps, so the default **permit** appears in the final configuration.

[2] Depending upon your IOS version, trying to change a match or set route-map statement without specifying a sequence number might result in an IOS message "% Please specify the entry by its sequence," rather than assuming that sequence number 10 is to be changed.

A packet or route is passed sequentially through route map statements. If a match is made, any **set** statements are executed and the permit or deny action is executed. As with access lists, processing stops when a match is made and the specified action is executed; the route or packet is not passed to subsequent statements. Consider the route map in Example 14-5.

Example 14-5 *Route-map Sluggo.*

```
route-map Sluggo permit 10
 match ip route-source 1
 set next-hop 192.168.1.5
!
route-map Sluggo permit 20
 match ip route-source 2
 set next-hop 192.168.1.10
!
route-map Sluggo permit 30
 match ip route-source 3
 set next-hop 192.168.1.15
```

If a route does not match statement 10, it will be passed to statement 20. If a match is made at statement 20, the **set** command will be executed and the route will be permitted. The matched route will not be passed on to statement 30.

The behavior of a "deny" action depends on whether the route map is being used for policy routing or for redistribution. If a route map is being used for redistribution and a route matches a statement with a deny action, the route will not be redistributed. If the route map is being used for policy routing and a packet matches a statement with a deny action, the packet is not policy routed but is passed back to the normal routing process for forwarding.

Again as with access lists, there must be a default action for the route map to take in the event that a route or packet passes through every statement without a match. An implicit deny exists at the end of every route map. Routes that pass through a redistribution route map without a match are not redistributed, and packets that pass through a policy route map without a match are sent to the normal routing process.

If no **match** statement is configured under a route map statement, the default action is to match everything.

Each map statement might have multiple **match** and **set** statements, as shown in Example 14-6.

Example 14-6 *Route map Garfield contains multiple match and set statements for the map statement with sequence number 10.*

```
route-map Garfield permit 10
match ip route-source 15
match interface Serial0
set metric-type type-1
set next-hop 10.1.2.3
```

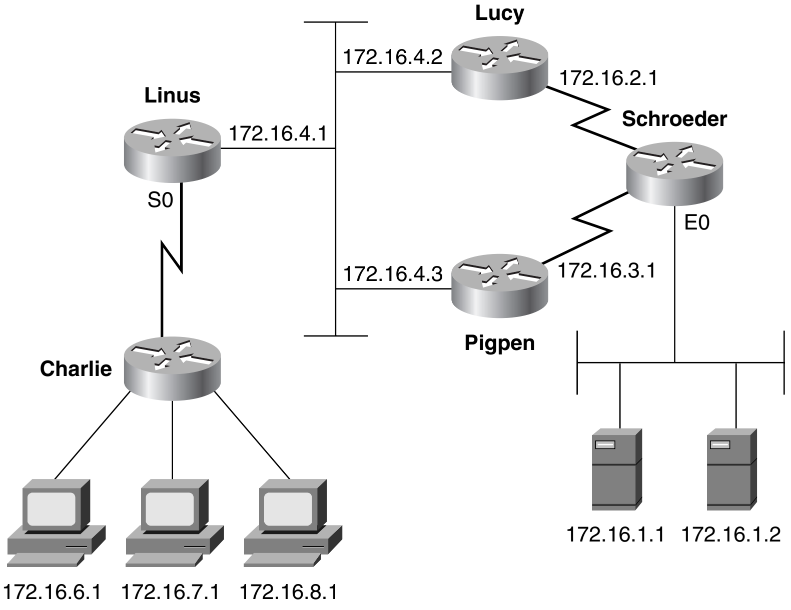
In a case such as this, every **match** statement must be matched for the **set** statements to be executed.

Case Study: Policy Routing

Policy routing is defined with the command **ip policy route-map**. The command is configured on an interface and affects incoming packets only.

Suppose a policy were to be implemented on Linus in Figure 14-3 such that traffic from subnet 172.16.6.0/24 would be forwarded to Lucy and traffic from subnet 172.16.7.0/24 would be forwarded to Pigpen. Linus's configuration is displayed in Example 14-7.

Figure 14-3 *Policy routes can be configured at Linus to route some packets through Lucy and other packets through Pigpen.*



Example 14-7 *Linus's policy routing configuration.*

```
interface Serial0
 ip address 172.16.5.1 255.255.255.0
 ip policy route-map Sally
!
access-list 1 permit 172.16.6.0 0.0.0.255
access-list 2 permit 172.16.7.0 0.0.0.255
!
route-map Sally permit 10
 match ip address 1
 set ip next-hop 172.16.4.2
!
route-map Sally permit 15
 match ip address 2
 set ip next-hop 172.16.4.3
```

The policy routing command on S0 sends incoming packets to route map Sally. Statement 10 of route map Sally uses access list 1 to identify source addresses from subnet 172.16.6.0/24. If a match is made, the packet is forwarded to Lucy, whose next-hop interface address is 172.16.4.2. If no match is made, the packet is sent to statement 15. That statement uses access list 2 to match source addresses from subnet 172.16.7.0/24. If a match is made at that statement, the packet is forwarded to Pigpen (172.16.4.3). Any packets that do not match statement 15, such as packets sourced from subnet 172.16.8.0/24, are routed normally. Example 14-8 shows the results of the policy route.[3]

Example 14-8 *The policy route configured on Linus's S0 interface routes packets from subnet 172.16.6.0/24 to Lucy (172.16.4.2) and routes packets from subnet 172.16.7.0/24 to Pigpen (172.16.4.3). Packets from subnet 172.16.8.0/24, which do not match the policy route, are routed normally (load balancing between Lucy and Pigpen).*

```
Linus#debug ip packet 5
IP packet debugging is on for access list 5
Linus#
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.6.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.6.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.6.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.6.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.8.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.8.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.8.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.8.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.3, len 60, forward
```

Suppose Lucy's Ethernet interface fails. Linus would still attempt to forward packets from 172.16.6.0 to Lucy's IP address. Because a match is made in the route map, even if the specified next-hop interface were down, no other matches would be attempted nor would the packet be routed normally. To force Linus to verify the availability of the next-hop address before attempting to forward the packet, use the command **set ip next-hop verify-availability**. Linus will search its CDP neighbors table to verify that the next-hop address is listed. If it is not, the policy is rejected and the packet is forwarded normally. Example 14-9 shows the output of the commands **debug ip policy** and **debug arp** while Lucy's Ethernet interface is down. Packets are matched and policy routed, even while the router is attempting to ARP for the next-hop address, without receiving an ARP reply. The packets are dropped.

Example 14-10 shows the output of the **debug ip policy** command with the addition of the **set ip next-hop verify-availability** command in the IP policy configuration. The example shows Linus's operation with Lucy's Ethernet interface still down. The packets are matched by the policy, but the policy is rejected because the next-hop address is no longer in Linus's CDP table. Since the policy is rejected, the packets are routed using the normal method.

[3] Note that the **debug ip packet** command references an access list 5. This access list permits only the subnets connected to router Charlie so that uninteresting traffic is not displayed by the debug function.

Example 14-9 **debug ip policy** *and* **debug arp** *on Linus shows packets are attempting to be routed according to the configured policy, even if the next hop specified by the policy is unavailable.*

```
Linus#debug arp
ARP packet debugging is on
Linus#debug ip policy
Policy routing debugging is on
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, policy match
IP: route map Sally, item 10, permit
IP: s=172.16.6.1 (Serial0), d=172.16.2.1 (Ethernet0), len 100, policy routed
IP: Serial0 to Ethernet0 172.16.4.2
IP ARP: sent req src 172.16.4.1 0004.c150.e700,
                 dst 172.16.4.2 0000.0000.0000 Ethernet0
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, policy match
IP: route map Sally, item 10, permit
IP: s=172.16.6.1 (Serial0), d=172.16.2.1 (Ethernet0), len 100, policy routed
IP: Serial0 to Ethernet0 172.16.4.2
IP ARP: sent req src 172.16.4.1 0004.c150.e700,
                 dst 172.16.4.2 0000.0000.0000 Ethernet0
IP ARP: creating incomplete entry for IP address: 172.16.4.2 interface Ethernet0
IP ARP: sent req src 172.16.4.1 0004.c150.e700,
                 dst 172.16.4.2 0000.0000.0000 Ethernet0
IP ARP throttled out the ARP Request for 172.16.4.2
```

Example 14-10 **debug ip policy** *on Linus with* **set ip next-hop verify-availability** *configured, shows the policy is rejected (the next hop is not in Linus's CDP neighbor table) and packets are routed normally (not policy routed).*

```
Linus#debug ip policy
Policy routing debugging is on

IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy rejected - normal
forwarding
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy rejected - normal
forwarding
```

Example 14-11 shows the output of the same debug command, **debug ip policy**, after Lucy's Ethernet interface is backed up. The output shows that the packets are successfully policy routed.

Example 14-11 **debug ip policy** *on Linus with* **set ip next-hop verify-availability** *configured. The packets are policy routed when the next hop is verified.*

```
Linus#
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, g=172.16.4.2, len 100, FIB policy routed
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, len 100, FIB policy match
IP: s=172.16.6.1 (Serial0), d=172.16.2.1, g=172.16.4.2, len 100, FIB policy routed
```

Standard IP access lists are used when policy routing by source address only. To route by both source and destination, an extended IP access list is used. The configuration in Example 14-12 causes packets from any subnet to host 172.16.1.1 to be forwarded to Lucy, whereas packets from host 172.16.7.1 to host 172.16.1.2 are forwarded to Pigpen. All other packets are routed normally.

Example 14-12 *Policy route maps can reference extended IP access-lists to specify a source and destination address pair to match, as shown in this configuration.*

```
interface Serial0
ip address 172.16.5.1 255.255.255.0
ip policy route-map Sally
!
access-list 101 permit ip any host 172.16.1.1
access-list 102 permit ip host 172.16.7.1 host 172.16.1.2
!
route-map Sally permit 10
match ip address 101
set ip next-hop 172.16.4.2
!
route-map Sally permit 15
match ip address 102
set ip next-hop 172.16.4.3
```

Route map Sally is again used, except the **match** statements now reference access lists 101 and 102. Example 14-13 shows the results.

Example 14-13 *Packets from any host to host 172.16.1.1 match statement 10 of route map Sally and are forwarded to Lucy. Packets from host 172.16.7.1 to host 172.16.1.2 are forwarded to Pigpen. Packets from another address on subnet 172.16.7.0/24 to host 172.16.1.2 are not matched by Sally and are routed normally.*

```
Linus#debug ip packet 5
IP packet debugging is on for access list 5
Linus#
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.1 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.1 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.254 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.254 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.2, len 60, forward
IP: s=172.16.7.254 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.3, len 60, forward
IP: s=172.16.7.254 (Serial0), d=172.16.1.2 (Ethernet0), g=172.16.4.2, len 60, forward
```

Next, suppose your policy states that FTP traffic from the servers on subnet 172.16.1.0/24 should be forwarded to Lucy and that Telnet traffic from the same servers should be forwarded to Pigpen. This plan allows the bulk FTP traffic and the bursty, interactive Telnet traffic to be segregated on the two serial links from Schroeder. Schroeder will have the configuration in Example 14-14.

Example 14-14 *Schroeder's policy route configuration forwarding FTP and Telnet traffic.*

```
interface Ethernet0
ip address 172.16.1.4 255.255.255.0
ip policy route-map Rerun
!
access-list 105 permit tcp 172.16.1.0 0.0.0.255 eq ftp any
access-list 105 permit tcp 172.16.1.0 0.0.0.255 eq ftp-data any
access-list 106 permit tcp 172.16.1.0 0.0.0.255 eq telnet any
!
route-map Rerun permit 10
match ip address 105
set ip next-hop 172.16.2.1
!
route-map Rerun permit 20
match ip address 106
set ip next-hop 172.16.3.1
```

Access lists 105 and 106 are examining not only the source and destination addresses, but also the source port. In Example 14-15, the **detail** option is used with **debug ip packet** to allow observation of the packet types being forwarded by Schroeder. An access list 10 limits the displayed packets to those from 172.16.1.1 to 172.16.6.1.

Example 14-15 *FTP packets (TCP ports 20 and 21) are being forwarded to Lucy, whereas Telnet packets (TCP port 23) with the same source and destination addresses are forwarded to Pigpen. Echo Reply packets (ICMP type 0), which do not find a match in the policy route, are routed normally.*

```
Schroeder#debug ip packet detail 10
IP packet debugging is on (detailed) for access list 10
Schroeder#
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1064, forward
    TCP src=20, dst=1047, seq=3702770065, ack=591246297, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 64, forward
    TCP src=21, dst=1046, seq=3662108731, ack=591205663, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1476, forward
    TCP src=20, dst=1047, seq=3702771089, ack=591246297, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forward
    TCP src=23, dst=1048, seq=3734385279, ack=591277873, win=14332 ACK
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 52, forward
    TCP src=23, dst=1048, seq=3734385279, ack=591277873, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forward
    TCP src=23, dst=1048, seq=3734385291, ack=591277876, win=14332 ACK
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
```

The purpose of segregating bulk and interactive traffic, as demonstrated in the last example, is so that the small packets characteristic of interactive traffic do not become delayed by the large packets characteristic of bulk traffic. The problem with the approach in this example is that if many types of traffic must be segregated, the access lists identifying the traffic by destination port might become prohibitively large.

If the objective is to segregate small packets from large packets, the length of the packet can be matched, as shown in Example 14-16.

Example 14-16 *Schroeder's policy route configuration forwards traffic based on packet length.*

```
interface Ethernet0
ip address 172.16.1.4 255.255.255.0 ip policy route-map Woodstock !
route-map Woodstock permit 20
match length 1000 1600
set ip next-hop 172.16.2.1
!
route-map Woodstock permit 30
match length 0 400
set ip next-hop 172.16.3.1
```

Here the **match length** statement specifies a minimum and a maximum packet size. Statement 20 of the route map causes all packets between 1000 and 1600 octets in length to be routed across the serial link to Lucy. Statement 30 causes all packets up to 400 octets in length to be routed across the serial link to Pigpen. Packets between 400 and 1000 octets are routed normally.

Example 14-17 shows the results of the new route map. Again there are FTP, Telnet, and Echo Reply packets from 172.16.1.2 to 172.16.6.1, but now the packets are routed according to their size instead of their addresses and ports.

Example 14-17 *Packets of 1000 octets or larger are routed to Lucy, whereas packets of 400 octets or less are routed to Pigpen. Any packets between 400 and 1000 octets are routed normally.*

```
Schroeder#debug ip packet detail 10
IP packet debugging is on (detailed) for access list 10
Schroeder#
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1476, forward
    TCP src=20, dst=1063, seq=1528444161, ack=601956937, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1476, forward
    TCP src=20, dst=1063, seq=1528442725, ack=601956937, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial0), g=172.16.2.1, len 1476, forward
    TCP src=20, dst=1063, seq=1528444161, ack=601956937, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 840, forward
    TCP src=20, dst=1063, seq=1528445597, ack=601956937, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forward
    TCP src=21, dst=1062, seq=1469372904, ack=601897901, win=14329 ACK
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 54, forward
    TCP src=21, dst=1062, seq=1469372904, ack=601897901, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forward
    TCP src=21, dst=1062, seq=1469372918, ack=601897901, win=14335 ACK FIN
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 44, forward
    TCP src=23, dst=1064, seq=1712116521, ack=602140570, win=14335 ACK SYN
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 43, forward
    TCP src=23, dst=1064, seq=1712116522, ack=602140570, win=14335 ACK PSH
```

Example 14-17 *Packets of 1000 octets or larger are routed to Lucy, whereas packets of 400 octets or less are routed to Pigpen. Any packets between 400 and 1000 octets are routed normally. (Continued)*

```
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 40, forward
    TCP src=23, dst=1064, seq=1712116525, ack=602140573, win=14332 ACK
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 52, forward
    TCP src=23, dst=1064, seq=1712116525, ack=602140573, win=14335 ACK PSH
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
IP: s=172.16.1.2 (Ethernet0), d=172.16.6.1 (Serial1), g=172.16.3.1, len 60, forward
    ICMP type=0, code=0
```

The policy routes demonstrated so far affect packets entering the router from a particular interface. But what about packets generated by the router itself? These can also be policy routed, with the command **ip local policy route-map**. Unlike the **ip policy route-map** command, which is configured on an interface, this command is configured globally on the router.

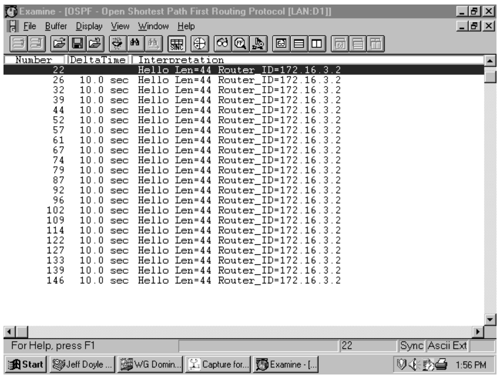
To apply the previously demonstrated policy to packets generated by Schroeder, the configuration is as displayed in Example 14-18.

Example 14-18 *Route policy configuration for packets generated by Schroeder.*

```
interface Ethernet0
 ip address 172.16.1.4 255.255.255.0
 ip policy route-map Woodstock
!
ip local policy route-map Woodstock
!
access-list 120 permit ip any 172.16.1.0 0.0.0.255
access-list 120 permit ospf any any
!
route-map Woodstock permit 10
 match ip address 120
!
route-map Woodstock permit 20
 match length 1000 1600
 set ip next-hop 172.16.2.1
!
route-map Woodstock permit 30
 match length 0 400
 set ip next-hop 172.16.3.1
```

Of particular interest is statement 10. This statement does not have a **set** statement, but merely permits packets that match access list 120. Access list 120, in turn, permits all packets destined for subnet 172.16.1.0/24 and all OSPF packets. Without the first line of the access list, some packets originated by Schroeder and destined for subnet 172.16.1.0/24 would be forwarded to the wrong interface by statement 20 or 30. Figure 14-4 shows why the second line of the access list is necessary. The length of Schroeder's OSPF Hellos is 44 octets. If statement 10 were not included, the OSPF Hellos would all match statement 30 and be forwarded to Pigpen, breaking the adjacency between Lucy and Schroeder. By matching statement 10, the OSPF packets are permitted with no changes and are forwarded normally.

Figure 14-4 *The length of the OSPF Hello packets is seen in this analyzer capture.*

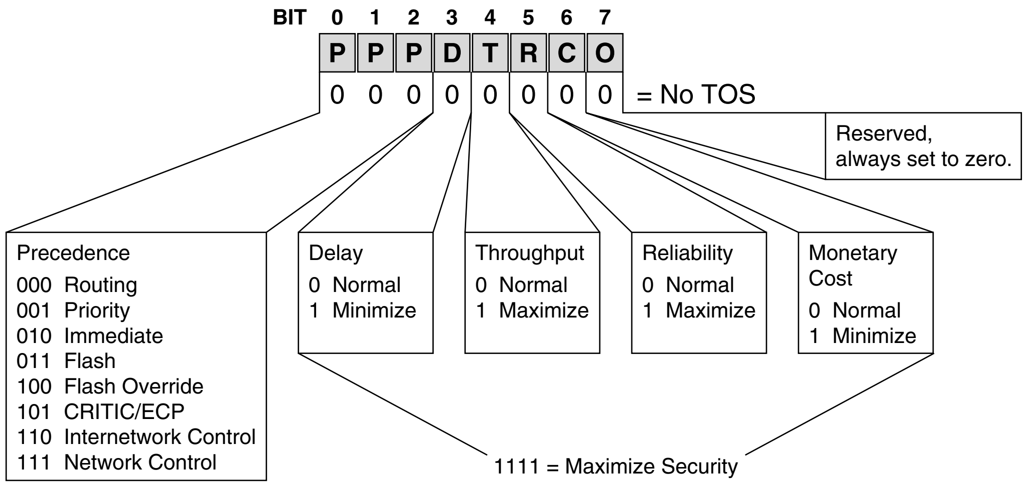

Examine - [OSPF - Open Shortest Path First Routing Protocol [LAN:D1]]
File Buffer Display View Window Help

```
 Number  |DeltaTime| Interpretation
      22            Hello Len=44 Router_ID=172.16.3.2
      26  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      32  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      39  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      44  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      52  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      57  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      61  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      67  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      74  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      79  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      87  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      92  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
      96  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
     102  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
     109  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
     114  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
     122  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
     127  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
     133  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
     139  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
     146  10.0 sec  Hello Len=44 Router_ID=172.16.3.2
```

Case Study: Policy Routing and Quality of Service Routing

Although quality of service (QoS) routing is outside the scope of this volume, it must be noted here that policy routing can be an integral part of QoS. Policy routing in conjunction with QoS is done by setting the Precedence or the Type of Service (TOS) bits of the TOS field in the IP headers of packets as they enter a router's interface. Figure 14-5 shows the bits of the TOS field. Although the TOS bits are seldom used in modern networks, the Precedence bits have found new life in QoS applications. The TOS bits are used to influence the path a router selects for a packet, whereas the Precedence bits are used to prioritize packets within a router.

Figure 14-5 *The Precedence and TOS bits of the Type of Service field of the IP header.*



The Precedence bits are set by using the **set ip Precedence** statement within a route map. The Precedence might be set by specifying the decimal equivalent of the three Precedence bits or by using keywords. Table 14-5 shows the decimal numbers and the keywords that can be used.

Table 14-5 *Precedence values and keywords used with the* **set ip precedence** *command.*

Bits	Number	Keyword
000	0	**routine**
001	1	**priority**
010	2	**immediate**
011	3	**flash**
100	4	**flash-override**
101	5	**critical**
110	6	**internet**
111	7	**network**

The TOS bits are set by using the **set ip tos** statement. Like the Precedence statement, the argument of the statement might be a number or a keyword, as shown in Table 14-6. Unlike Precedence, you might use a combination of TOS values. For example, specifying a TOS value of 12 (1100b) means *minimum delay* and *maximum throughput*. Only a single keyword can be used, so to set a combination of TOS values, a number must be specified.

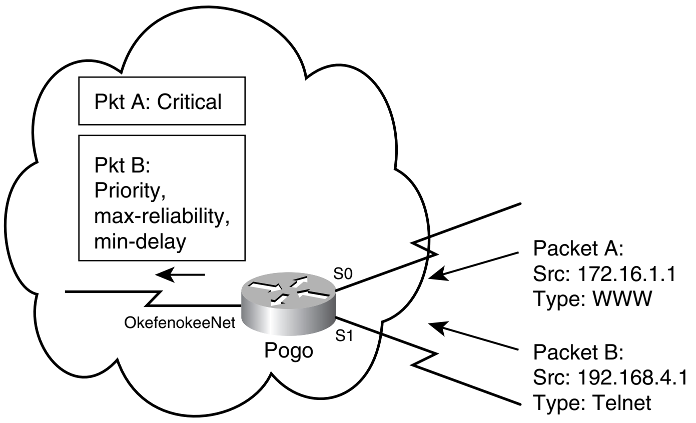
Table 14-6 *TOS values and keywords used with the* **set ip tos** *command.*

Bits	Number (0–15)	Keyword
0000	0	**normal**
0001	1	**min-monetary-cost**
0010	2	**max-reliability**
0100	4	**max-throughput**
1000	8	**min-delay**

Figure 14-6 shows an example of how policy routes can be used for QoS routing.

Here, router Pogo is at the "edge" of the Internet OkefenokeeNet. By configuring policy routes on Pogo's serial links, the Precedence or TOS bits of incoming packets can be changed so that IP traffic is divided into several traffic classes. See Example 14-19 for instance.

Figure 14-6 *Policy routes can be used to set the Precedence or TOS bits of packets entering a network. The routers within the network can then make QoS decisions based on the setting of these bits.*



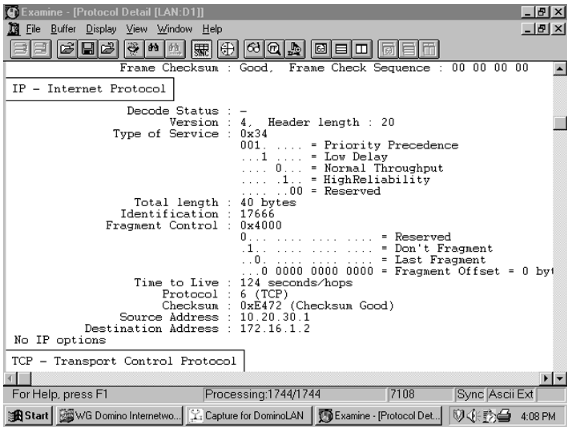
Example 14-19 *Pogo's configuration sets Precedence and TOS bits.*

```
interface Serial0
ip address 10.1.18.67 255.255.255.252
ip policy route-map Albert
!
interface Serial1
ip address 10.34.16.83 255.255.255.252
ip policy route-map Albert
!
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 110 permit tcp any eq www any
!
route-map Albert permit 10
match ip address 1 110
set ip precedence critical
!
route-map Albert permit 20
set ip tos 10
set ip precedence priority
```

Statement 10 says that if packets match both access lists 1 and 110, the Precedence will be set to *critical*. Notice that statement 20 has no **match** statement. This statement will match any packets that haven't been matched by statement 10. There are also two **set** statements under statement 20. These statements will set the TOS bits to *minimum delay* and *maximum reliability* and will set the Precedence bits to *priority*. Figure 14-7 shows a capture of a packet from somewhere inside OkefenokeeNet, which has been modified by the route map at Pogo.

After the Precedence or TOS bits have been set in packets entering the network, the routers within the Internet can make QoS decisions based in part or wholly on the class of service these bits define. For example, priority, custom, or weighted fair queuing might be configured to prioritize traffic according to the Precedence or TOS bits. In some implementations, Precedence can be used with congestion avoidance mechanisms such as Weighted Random Early Detection (WRED). Or a crude Class-of-Service routing can be implemented by configuring access lists that permit or deny packets across certain links based on the setting of their Precedence or TOS bits.

Figure 14-7 *Pogo's policy route has set the Precedence bits of this packet to priority (001b) and the TOS bits to minimum delay and maximum reliability (1010b).*



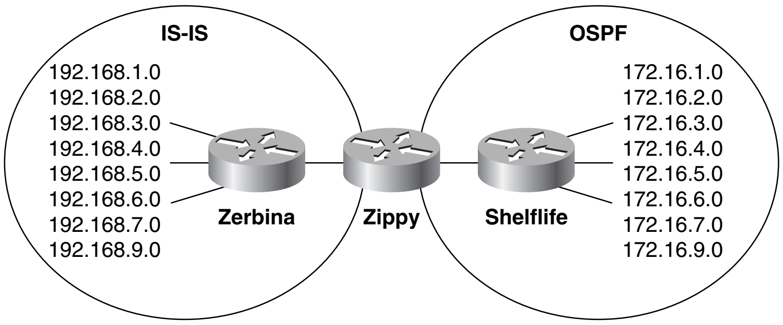
```
                Frame Checksum : Good,  Frame Check Sequence : 00 00 00 00
IP - Internet Protocol

                 Decode Status : -
                       Version : 4,  Header length : 20
               Type of Service : 0x34
                                 001. .... = Priority Precedence
                                 ...1 .... = Low Delay
                                 .... 0... = Normal Throughput
                                 .... .1.. = HighReliability
                                 .... ..00 = Reserved
                  Total length : 40 bytes
                Identification : 17666
              Fragment Control : 0x4000
                                 0... .... .... .... = Reserved
                                 .1.. .... .... .... = Don't Fragment
                                 ..0. .... .... .... = Last Fragment
                                 ...0 0000 0000 0000 = Fragment Offset = 0 byt
                  Time to Live : 124 seconds/hops
                      Protocol : 6 (TCP)
                      Checksum : 0xE472 (Checksum Good)
                Source Address : 10.20.30.1
           Destination Address : 172.16.1.2
 No IP options
TCP - Transport Control Protocol
```

Case Study: Route Maps and Redistribution

A route map can be used with redistribution for both IPv4 and IPv6 by adding a call to the route map in the **redistribute** command. Figure 14-8 shows a network in which IPv4 IS-IS and OSPF routes are being mutually redistributed at router Zippy. Of the network and subnet addresses listed in the illustration, only the ones whose third octet is odd-numbered are to be redistributed.

Figure 14-8 *The OSPF and IS-IS routes are being mutually redistributed. Route maps can be used with the* **redistribute** *command as simple route filters, or they can be used to modify characteristics of the redistributed routes.*



Zippy's configuration is displayed in Example 14-20.

Example 14-20 *Zippy is configured to redistribute only addresses with an odd numbered third octet.*

```
router ospf 1
 redistribute isis level-1 metric 20 subnets route-map Griffy
 network 172.16.10.2 0.0.0.0 area 5
!
router isis
 redistribute ospf 1 metric 25 route-map Toad metric-type internal level-2
 net 47.0001.1234.5678.9056.00
!
access-list 1 permit 192.168.2.0
access-list 1 permit 192.168.4.0
access-list 1 permit 192.168.6.0
access-list 2 permit 172.16.1.0
access-list 2 permit 172.16.3.0
access-list 2 permit 172.16.5.0
access-list 2 permit 172.16.7.0
access-list 2 permit 172.16.9.0
!
route-map Griffy deny 10
 match ip address 1
!
route-map Griffy permit 20
!
route-map Toad permit 10
 match ip address 2
```

Route maps Griffy and Toad perform the same functions, but with different logic. Griffy uses negative logic, identifying the routes that should not be redistributed, and Toad uses positive logic, identifying the routes that should be redistributed.

Statement 10 of Griffy denies any routes that are permitted by access list 1 (the addresses with an even third octet). Because the addresses with odd-numbered third octets do not find a match at statement 10, they are passed to statement 20. Statement 20 has no **match** statement, so the default is to match everything. Statement 20 has a permit action, so the odd routes are permitted. The result is shown in Example 14-21.

Example 14-21 *The only destinations within the IS-IS domain that are contained in Shelflife's route table are those with an odd-numbered third octet.*

```
Shelflife#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
O  E2  192.168.9.0 [110/20] via 172.16.10.2, 00:24:46, Ethernet0
O  E2  192.168.1.0 [110/20] via 172.16.10.2, 00:24:46, Ethernet0
O  E2  192.168.3.0 [110/20] via 172.16.10.2, 00:24:46, Ethernet0
```

Example 14-21 *The only destinations within the IS-IS domain that are contained in Shelflife's route table are those with an odd-numbered third octet. (Continued)*

```
O  E2  192.168.5.0 [110/20] via 172.16.10.2, 00:24:47, Ethernet0
O  E2  192.168.7.0 [110/20] via 172.16.10.2, 00:24:47, Ethernet0
       172.16.0.0 255.255.255.0 is subnetted, 9 subnets
C         172.16.9.0 is directly connected, Serial0
C         172.16.10.0 is directly connected, Ethernet0
O         172.16.4.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O         172.16.5.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O         172.16.6.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O         172.16.7.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O         172.16.1.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O         172.16.2.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
O         172.16.3.0 [110/159] via 172.16.9.2, 14:05:33, Serial0
Shelflife#
```

Route map Toad has a single statement that permits routes that have been permitted by access list 2 (addresses with an odd third octet). The addresses with an even third octet do not find a match at access list 2. The default route map statement when redistributing is to deny all routes, so the addresses that are not matched by access list 2 are not redistributed. Example 14-22 shows the results of route map Toad.

Example 14-22 *The only destinations within the OSPF domain that are contained in Zerbina's route table are those with an odd-numbered third octet.*

```
Zerbina#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
Gateway of last resort is not set
C    192.168.9.0/24 is directly connected, Serial0
C    192.168.10.0/24 is directly connected, Ethernet0
i L1 192.168.1.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.2.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.3.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.4.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.5.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.6.0/24 [115/15] via 192.168.9.2, Serial0
i L1 192.168.7.0/24 [115/15] via 192.168.9.2, Serial0
     172.16.0.0/24 is subnetted, 5 subnets
i L2    172.16.9.0 [115/35] via 192.168.10.2, Ethernet0
i L2    172.16.5.0 [115/35] via 192.168.10.2, Ethernet0
i L2    172.16.7.0 [115/35] via 192.168.10.2, Ethernet0
i L2    172.16.1.0 [115/35] via 192.168.10.2, Ethernet0
i L2    172.16.3.0 [115/35] via 192.168.10.2, Ethernet0
Zerbina#
```

Other configurations will achieve the same ends. For instance, route map Toad will have the same effect with the access list in Example 14-23.

Example 14-23 *An alternate configuration for route-map Toad on Zippy.*

```
access-list 2 deny 172.16.2.0
access-list 2 deny 172.16.4.0
access-list 2 deny 172.16.6.0
access-list 2 permit any
```

Although route maps work fine as simple route filters, their strength lies in their ability to change the routes in various ways. Consider the configuration in Example 14-24 of Zippy in Figure 14-8.

Example 14-24 *Zippy's route-map configuration sets the metric type, metric and level of certain redistributed routes.*

```
router ospf 1
 redistribute isis level-1 metric 20 subnets route-map Griffy
 network 172.16.10.2 0.0.0.0 area 5
!
router isis
 redistribute ospf 1 metric 25 route-map Toad metric-type internal level-2
 net 47.0001.1234.5678.9056.00
!
ip classless
access-list 1 permit 192.168.2.0
access-list 1 permit 192.168.4.0
access-list 1 permit 192.168.6.0
access-list 2 permit 172.16.9.0
access-list 2 permit 172.16.5.0
access-list 2 permit 172.16.7.0
access-list 2 permit 172.16.1.0
access-list 2 permit 172.16.3.0
!
route-map Griffy permit 10
 match ip address 1
 set metric-type type-1
!
route-map Griffy permit 20
!
route-map Toad permit 10
 match ip address 2
 set metric 15
 set level level-1
!
route-map Toad permit 20
```

Statement 10 of route map Griffy permits routes to the addresses in access list 1 and redistributes them into OSPF as type 1 external routes. Statement 20 permits all other routes, which will be redistributed with the default external type 2. Example 14-25 shows the results.

Example 14-25 *The routes to destinations in the IS-IS domain are E1 if the third octet of the address is even and E2 if the third octet is odd.*

```
Shelflife#show ip route
Codes:C -connected,S -static,I -IGRP,R -RIP,M -mobile,B -BGP
      D -EIGRP,EX -EIGRP external,O -OSPF,IA -OSPF inter area
      E1 -OSPF external type 1,E2 -OSPF external type 2,E -EGP
      i -IS-IS,L1 -IS-IS level-1,L2 -IS-IS level-2,*-candidate default
Gateway of last resort is not set
O E2 192.168.9.0 [110/20 ] via 172.16.10.2,,00:13:43,Ethernet0
O E2 192.168.1.0 [110/20 ] via 172.16.10.2,,00:13:43,Ethernet0
O E1 192.168.2.0 [110/30 ] via 172.16.10.2,,00:13:43,Ethernet0
O E2 192.168.3.0 [110/20 ] via 172.16.10.2,,00:13:44,Ethernet0
O E1 192.168.4.0 [110/30 ] via 172.16.10.2,,00:13:44,Ethernet0
O E2 192.168.5.0 [110/20 ] via 172.16.10.2,,00:13:44,Ethernet0
O E1 192.168.6.0 [110/30 ] via 172.16.10.2,,00:13:44,Ethernet0
O E2 192.168.7.0 [110/20 ] via 172.16.10.2,,00:13:44,Ethernet0
     172.16.0.0 255.255.255.0 is subnetted,9 subnets
C       172.16.9.0 is directly connected,Serial0
C       172.16.10.0 is directly connected,Ethernet0
O       172.16.4.0 [110/159 ] via 172.16.9.2,,15:49:29,Serial0
O       172.16.5.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.6.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.7.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.1.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.2.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
O       172.16.3.0 [110/159 ] via 172.16.9.2,,15:49:30,Serial0
Shelflife#
```

Statement 10 of route map Toad permits routes to addresses in access list 2 and redistributes them into IS-IS as level 1 routes. Their metric is set to 15. Statement 20 permits all other routes, which will be redistributed as level 2 and with a metric of 25, as specified by the **redistribute** command under the IS-IS configuration (see Example 14-26).

Example 14-26 *The routes to destinations in the OSPF domain are L2 if the third octet of the address is even and L1 if the third octet is odd. The "odds" are redistributed with a metric of 15, and the "evens" are redistributed with a metric of 25 (10 is added for the hop from Zippy to Zerbina).*

```
Zerbina#show ip route
Codes:C -connected,S -static,I -IGRP,R -RIP,M -mobile,B -BGP
      D -EIGRP,EX -EIGRP external,O -OSPF,IA -OSPF inter area
      N1 -OSPF NSSA external type 1,N2 -OSPF NSSA external type 2
      E1 -OSPF external type 1,E2 -OSPF external type 2,E -EGP
      i -IS-IS,L1 -IS-IS level-1,L2 -IS-IS level-2,*-candidate default
      U -per-user static route,o -ODR
Gateway of last resort is not set
C    192.168.9.0/24 is directly connected,Serial0
C    192.168.10.0/24 is directly connected,Ethernet0
i L1 192.168.1.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.2.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.3.0/24 [115/15 ] via 192.168.9.2,,Serial0
```

continues

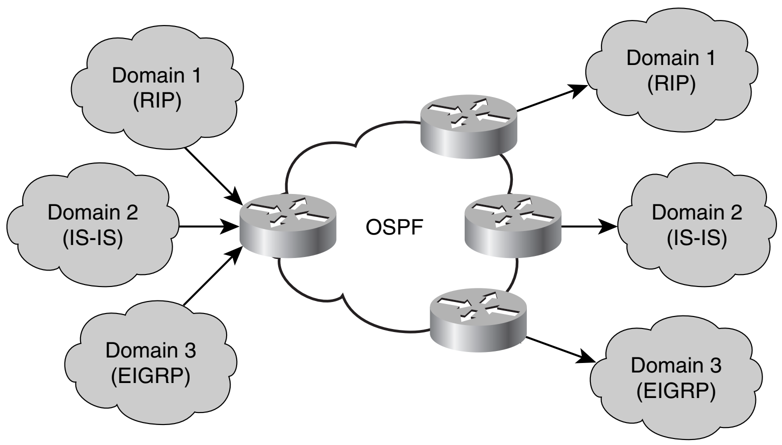
Example 14-26 *The routes to destinations in the OSPF domain are L2 if the third octet of the address is even and L1 if the third octet is odd. The "odds" are redistributed with a metric of 15, and the "evens" are redistributed with a metric of 25 (10 is added for the hop from Zippy to Zerbina). (Continued)*

```
i L1 192.168.4.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.5.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.6.0/24 [115/15 ] via 192.168.9.2,,Serial0
i L1 192.168.7.0/24 [115/15 ] via 192.168.9.2,,Serial0
     172.16.0.0/24 is subnetted,8 subnets
i L1    172.16.9.0 [115/25 ] via 192.168.10.2,,Ethernet0
i L2    172.16.4.0 [115/35 ] via 192.168.10.2,,Ethernet0
i L1    172.16.5.0 [115/25 ] via 192.168.10.2,,Ethernet0
i L2    172.16.6.0 [115/35 ] via 192.168.10.2,,Ethernet0
i L1    172.16.7.0 [115/25 ] via 192.168.10.2,,Ethernet0
i L1    172.16.1.0 [115/25 ] via 192.168.10.2,,Ethernet0
i L2    172.16.2.0 [115/35 ] via 192.168.10.2,,Ethernet0
i L1    172.16.3.0 [115/25 ] via 192.168.10.2,,Ethernet0
Zerbina#
```

Case Study: Route Tagging

Figure 14-9 shows a situation in which routes from several routing domains, each running a separate routing protocol, are being redistributed into a single transit domain running OSPF. On the other side of the OSPF cloud, the routes must be redistributed back into their respective domains. Route filters can be used at the egress points from the OSPF cloud into each domain to permit only the routes that belong to that domain. However, if each domain has many routes or if the routes within the domain change frequently, the route filters can become difficult to manage.

Figure 14-9 *Routes from each of the three domains on the left are redistributed into a transit network running OSPF. On the right, the routes for each domain must be redistributed back into their original domains.*

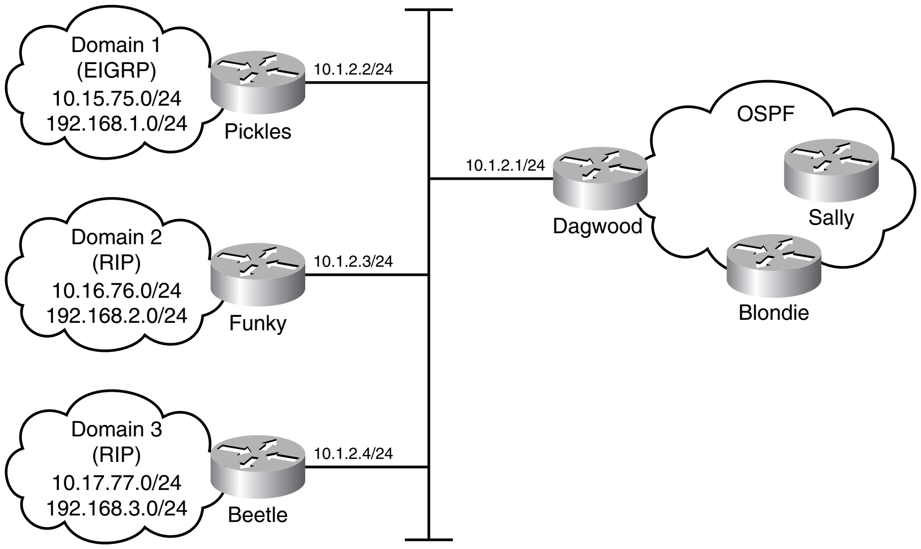


Another way of handling this problem is to tag the routes at their ingress points to the OSPF transit domain with a tag that is unique to each domain. At the egress points, the routes can be redistributed by their tags instead of by specific addresses. The routing protocol of the transit network does not necessarily use the tag, but merely conveys it to and from its external networks. RIPv2, EIGRP, Integrated IS-IS, and OSPF all support route tags. BGP also supports route tags. Tags are not supported by RIPv1. A case study in this section shows how a transit network running OSPF can use the route tags.

A re-examination of the packet formats in Chapter 6, "RIPv2, RIPng, and Classless Routing," Chapter 7, "Enhanced Interior Gateway Routing Protocol (EIGRP)," Chapter 8, "OSPFv2," and Chapter 10, "Integrated IS-IS" show that RIPv2 messages support 16-bit tags, IS-IS Inter-Domain Routing Protocol Information TLVs support 16-bit tags, and EIGRP external route TLVs and OSPF type 5 LSAs support 32-bit tags. These tags are expressed as decimal numbers, so tags carried by RIPv2 and IS-IS will be between 0 and 65,535, and tags carried by EIGRP and OSPF will be between 0 and 4,294,967,295.

In Figure 14-10, router Dagwood is accepting routes from three different routing domains and redistributing them into a domain running OSPF. The objective here is to tag the routes from each domain so that their source domain might be identified within the OSPF cloud. Routes from domain 1 will have a tag of 1, domain 2 will have a tag of 2, and so on.

Figure 14-10 *Dagwood is configured to tag the routes from each of the three routing domains as they are redistributed into OSPF.*



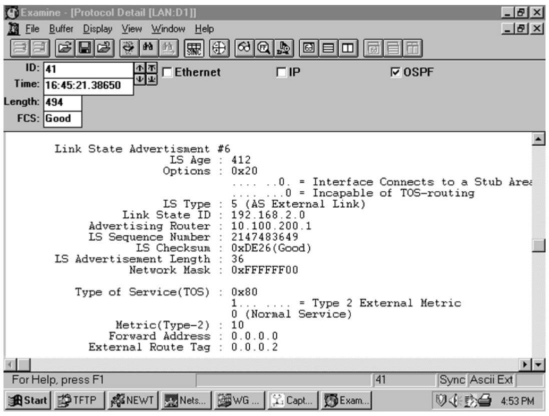
Dagwood's configuration is displayed in Example 14-27.

Example 14-27 *Dagwood is configured to tag routes as they are redistributed into OSPF from RIP and EIGRP.*

```
router ospf 1
  redistribute eigrp 1 metric 10 subnets tag 1
  redistribute rip metric 10 subnets route-map Dithers
  network 10.100.200.1 0.0.0.0 area 0
!
router rip
  network 10.0.0.0
!
router eigrp 1
  network 10.0.0.0
!
access-list 1 permit 10.1.2.3
access-list 2 permit 10.1.2.4
!
route-map Dithers permit 10
  match ip route-source 1
  set tag 2
!
route-map Dithers permit 20
  match ip route-source 2
  set tag 3
```

First, notice the **redistribute eigrp** command under OSPF. Dagwood is accepting routes from only one EIGRP domain, so the tag can be set to 1 directly on the **redistribute** command. However, routes are being learned from two RIP domains. Here a route map is needed. Route map Dithers sets the tag of the RIP routes to either 2 or 3, depending on whether the route was learned from Funky (10.1.2.3) or Beetle (10.1.2.4). Figure 14-11 shows an LSA advertising one of the RIP-learned routes, with the route tag set to 2.

Figure 14-11 *This type 5 LSA is advertising network 192.168.2.0, which is in domain 2, within the OSPF domain. The route tag is shown on the last line.*



ID: 41
Time: 16:45:21.38650
Length: 494
FCS: Good

☐ Ethernet ☐ IP ☑ OSPF

```
      Link State Advertisment #6
                      LS Age : 412
                     Options : 0x20
                               .... ..0. = Interface Connects to a Stub Area
                               .... ...0 = Incapable of TOS-routing
                     LS Type : 5 (AS External Link)
               Link State ID : 192.168.2.0
          Advertising Router : 10.100.200.1
          LS Sequence Number : 2147483649
                 LS Checksum : 0xDE26(Good)
      LS Advertisement Length : 36
                Network Mask : 0xFFFFFF00

         Type of Service(TOS) : 0x80
                               1... .... = Type 2 External Metric
                               0 (Normal Service)
              Metric(Type-2) : 10
             Forward Address : 0.0.0.0
          External Route Tag : 0.0.0.2
```

Examine - [Protocol Detail [LAN:D1]]
File Buffer Display View Window Help
For Help, press F1
41
Sync Ascii Ext
Start TFTP NEWT Nets... WG ... Capt... Exam...
4:53 PM

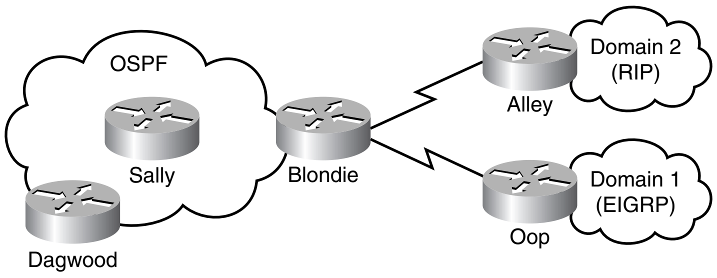
The route tags can also be observed in the OSPF link state database (see Example 14-28).

Example 14-28 *The OSPF link state database indicates the tags that were set for each of the external routes by Dagwood's redistribution processes.*

```
Blondie#show ip ospf database
       OSPF Router with ID (10.100.200.2)(Process ID 1)
               Router Link States (Area 0)
Link ID        ADV Router     Age    Seq#        Checksum  Link count
10.100.200.3   10.100.200.3   671    0x80000003  0x00A137  4
10.100.200.2   10.100.200.2   39     0x80000002  0x6FF5   3
10.100.200.1   10.100.200.1   40     0x80000033  0x33E1   3
               Net Link States (Area 0)
Link ID        ADV Router     Age    Seq#        Checksum
10.100.200.1   10.100.200.1   40     0x80000001  0xB0A7
               AS External Link States
Link ID        ADV Router    Age     Seq#        Checksum  Tag
192.168.2.0    10.100.200.1  641     0x80000028  0x904D   2
10.17.77.0     10.100.200.1  642     0x80000028  0xC817   3
192.168.3.0    10.100.200.1  642     0x80000028  0x9744   3
10.15.75.0     10.100.200.1  642     0x80000028  0xD213   1
10.1.2.0       10.100.200.1  642     0x80000028  0xA19B   1
10.16.76.0     10.100.200.1  642     0x80000028  0xCD15   2
192.168.1.0    10.100.200.1  644     0x80000028  0x8956   1
10.100.200.0   10.100.200.1  644     0x80000028  0x6EA4   1
Blondie#
```

In Figure 14-12, Blondie must redistribute only domain 2 routes to Alley and only domain 1 routes to Oop. Because the routes were tagged as they entered the OSPF transit domain, this is easily done. Blondie's configuration is shown in Example 14-29.

Figure 14-12 *Blondie is using route maps to redistribute routes according to their route tag.*



Example 14-29 *Blondie is configured to use route maps to redistribute routes with tag 1 to EIGRP and tag 2 to RIP.*

```
router ospf 1
  network 10.100.200.2 0.0.0.0 area 0
!
router rip
  redistribute ospf 1 match external 2 route-map Daisy
  passive-interface Ethernet0
  passive-interface Serial1
```

continues

Example 14-29 *Blondie is configured to use route maps to redistribute routes with tag 1 to EIGRP and tag 2 to RIP. (Continued)*

```
  network 10.0.0.0
  default-metric 5
!
router eigrp 1
  redistribute ospf 1 match external 2 route-map Herb
  passive-interface Ethernet0
  passive-interface Serial0
  network 10.0.0.0
  default-metric 10000 1000 255 1 1500
!
route-map Daisy permit 10
  match tag 2
!
route-map Herb permit 10
  match tag 1
```

Example 14-30 shows the resulting routes at Alley and Oop. One drawback to the use of route tags to filter routes is that there is no way to filter routes by interface. For example, if Blondie had to send routes to both domain 2 and domain 3, which both run RIP, route maps could not be configured to send some routes to one RIP process and other routes to another RIP process. The routes would have to be filtered by address with **distribute-list** commands.

Example 14-30 *The route tables of Alley and Oop in Figure 14-12 show the results of the redistribution configuration at Blondie.*

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

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 5 subnets, 3 masks
C       10.1.3.0/24 is directly connected, Serial0
R       10.1.5.4/30 [120/1] via 10.1.3.1, 00:00:25, Serial0
R       10.1.4.0/24 [120/1] via 10.1.3.1, 00:00:25, Serial0
R       10.16.76.0/24 [120/5] via 10.1.3.1, 00:00:25, Serial0
R       10.100.200.2/32 [120/1] via 10.1.3.1, 00:00:25, Serial0
R    192.168.2.0/24 [120/5] via 10.1.3.1, 00:00:25, Serial0
Alley#
```

```
Oop#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

Example 14-30 *The route tables of Alley and Oop in Figure 14-12 show the results of the redistribution configuration at Blondie. (Continued)*

```
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks
D       10.1.3.0/24 [90/2681856] via 10.1.4.1, 00:21:36, Serial0
D EX    10.1.2.0/24 [170/2425856] via 10.1.4.1, 00:08:22, Serial0
D       10.1.5.4/30 [90/2681856] via 10.1.4.1, 00:22:40, Serial0
C       10.1.4.0/24 is directly connected, Serial0
D EX    10.15.75.0/24 [170/2425856] via 10.1.4.1, 00:04:56, Serial0
D       10.100.200.2/32 [90/2297856] via 10.1.4.1, 00:22:40, Serial0
D EX 192.168.1.0/24 [170/2425856] via 10.1.4.1, 00:04:56, Serial0
Oop#
```

Case Study: Filtering Tagged Routes Out of OSPF Route Table

The network running OSPF in Figure 14-10 and Figure 14-12 is a transit network. If devices within that transit area do not need to send packets to any of the other domains, there is no need to maintain the addresses of those domains in the OSPF route tables. The tags that were added to the routes, along with distribute lists and route maps, can be applied to the OSPF routers to prevent the addresses from being added to the route tables, while not affecting the entries in the link-state database.

Sally, a router wholly within the OSPF domain, has been modified with the configuration in Example 14-31.

Example 14-31 *Sally's configuration uses tags to filter routes from the OSPF route table.*

```
router ospf 1
 network 10.100.200.1 0.0.0.0 area 0
 network 10.1.5.0 0.0.0.255 area 0
 distribute-list route-map Charlie in
!
route-map Charlie deny 10
 match tag  1 2 3
!
route-map Charlie permit 20
```

The route map Charlie denies addresses that are marked with tags 1, 2 or 3. These addresses are omitted from the IP route table by the **distribute-list in** command. Any other address is added to the route table, permitted by the route map sequence 20. The distribute list is not applied to Blondie and Dagwood, the edge routers that are performing redistribution. If an address is not in the route table, even if it is in the OSPF LSA database, it will not be redistributed into another routing protocol. Example 14-32 shows Sally's IP route table and OSPF LSA database.

Example 14-32 *The OSPF addresses marked with tags are filtered out of the IP route table. The addresses still exist in the OSPF LSA database.*

```
Sally#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 5 subnets, 2 masks
C       10.1.5.4/30 is directly connected, Serial0/0.2
C       10.1.5.0/30 is directly connected, Serial0/0.1
O       10.100.200.2/32 [110/65] via 10.1.5.6, 00:17:24, Serial0/0.2
C       10.100.200.3/32 is directly connected, Loopback0
O       10.100.200.1/32 [110/65] via 10.1.5.1, 00:17:24, Serial0/0.1
```

```
Sally#show ip ospf database

            OSPF Router with ID (10.100.200.3) (Process ID 1)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#     Checksum Link count
10.100.200.1    10.100.200.1    1183        0x80000004 0x003756 3
10.100.200.2    10.100.200.2    1181        0x80000004 0x00E1A1 3
10.100.200.3    10.100.200.3    1177        0x8000000A 0x00933E 4

                Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
10.1.2.0        10.100.200.1    94          0x80000003 0x00EB76 1
10.15.75.0      10.100.200.1    603         0x80000002 0x001FEC 1
10.16.76.0      10.100.200.1    346         0x80000002 0x001AEE 2
10.16.77.0      10.100.200.1    353         0x80000002 0x001FEE 2
192.168.1.0     10.100.200.1    603         0x80000002 0x00D530 1
192.168.2.0     10.100.200.1    346         0x80000002 0x00DC27 2
192.168.3.0     10.100.200.1    353         0x80000002 0x00E427 2
Sally#
```

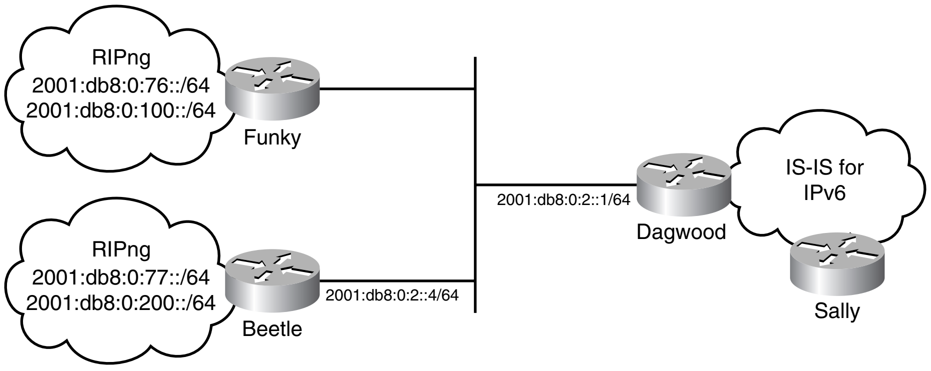
None of the tagged routes, tag 1, 2, or 3, exist in Sally's IP route table.

Case Study: IPv6 Redistribution with Route Maps

IPv6 routing protocols also support redistribution of routes using route maps. The configuration is almost identical to IPv4.

Figure 14-13 shows the addition of IPv6 addresses and routing protocols to the network shown in Figure 14-10. Funky and Beetle are each running RIPng. Dagwood is redistributing IPv6 prefixes between RIPng and IS-IS. Only IPv6 prefixes from Beetle are redistributed into IS-IS, and prefix 2001:db8:0:77::/64 is set with a metric of 10 while prefix 2001:db8:0:200::/64 has a metric of 100.

Figure 14-13 *IPv6 has been added to the network in Figure 14-10. IPv6 prefixes are redistributed between RIPng and IS-IS.*



Dagwood's configuration is displayed in Example 14-33.

Example 14-33 *Dagwood's IPv6 configuration.*

```
Interface ethernet 0/0
 ipv6 address 2001:db8:0:2::1/64
 ipv6 rip domain3 enable
!
interface serial 0/0.1 point-to-point
 ipv6 address 2001:db8:0:5::1/64
 ipv6 router isis
!
ipv6 router rip domain3
!
router isis
 net 00.0001.0000.5678.ef01.00
 Address-family ipv6
  Redistribute rip domain3 route-map Beetlefilter
!
route-map Beetlefilter permit 10
 match ipv6 route-source prefix-list 1
```

continues

Example 14-33 *Dagwood's IPv6 configuration. (Continued)*

```
 match ipv6 address prefix-list 3
 set metric 10
!
route-map Beetlefilter permit 20
 match ipv6 route-source prefix-list 1
 match ipv6 address prefix-list 2
 set metric 100
!
ipv6 prefix-list 1 permit 2001:db8:0:2::4/128
ipv6 prefix-list 2 permit 2001:db8:0:200::/64
ipv6 prefix-list 3 permit 2001:db8:0:77::/64
```

IPv6 prefix-lists are used to match the source of route information or to match specific addresses for redistribution. Statement 10 of Beetlefilter specifies that the route source must equal Beetle's IPv6 address and the prefix must equal 2001:db8:0:77::/64 for the metric to be set to 10. If the route source or the prefix does not match, statement 20 is executed. If the source is Beetle and the prefix is 2001:db8:0:200::/64, the metric is set to 100. If the source is not Beetle, or the prefix is not one of the ones specified, the route is not redistributed.

Sally's IS-IS database (Example 14-34) shows the redistributed prefixes.

Example 14-34 *The redistributed routes are seen in Sally's IS-IS database.*

```
Sally#show isis database detail Dagwood-00.00

IS-IS Level-1 LSP Dagwood.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Dagwood.00-00         0x00000004   0xCA8B        938               0/0/0
  Area Address: 00.0001
  NLPID:        0x8E
  Hostname: Dagwood
  IPv6 Address: 2001:DB8:0:5::1
  Metric: 10         IPv6 2001:DB8:0:5::/64
  Metric: 10         IS Sally.00

IS-IS Level-2 LSP Dagwood.00-00
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
Dagwood.00-00         0x00000008   0x28D7        1089              0/0/0
  Area Address: 00.0001
  NLPID:        0x8E
  Hostname: Dagwood
  IPv6 Address: 2001:DB8:0:5::1
  Metric: 10         IS Sally.00
  Metric: 10         IPv6 2001:DB8:0:5::/64
  Metric: 10         IPv6 2001:DB8:0:77::/64
  Metric: 100        IPv6 2001:DB8:0:200::/64
Sally#
```

Looking Ahead

This chapter concludes this book's in-depth look at routing TCP/IP with respect to interior gateway protocols. If you are preparing to become a CCIE, you certainly will want to know this book's topics thoroughly before taking the exam. Use the end-of-chapter questions and exercises to test your level of understanding and preparedness. And if you haven't studied routing TCP/IP with respect to Exterior Gateway Protocols, doing so is a next logical step in your preparation.

Summary Table: Chapter 14 Command Review

Command	Description
access-list *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*]	Defines a line of a standard IP access list
access-list *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**tos** *tos*] [**log**]	Defines a line of an extended IP access list
ip local policy route-map *map-tag*	Defines a policy route for packets originated by the router itself
ip policy route-map map-tag	Defines a policy route for packets transiting the router
match interface *type number* [*...type number*]	Matches routes that have their next hop out one of the interfaces specified
match ip address {*access-list-number* | *name*} [*...access-list-number* | *name*]	Matches routes that have a destination address specified by one of the access lists
match ip next-hop {*access-list-number* | *name*} [*...access-list-number* | *name*]	Matches routes that have a next-hop router address specified by one of the access lists
match ip route-source {*access-list-number* | *name*} [*...access-list-number* | *name*]	Matches routes that have been advertised by routers at the addresses specified in the access lists
match ipv6 address prefix-list {*name*}	Matches routes that have a destination address specified by the prefix-list
match ipv6 next-hop prefix-list {*name*}	Matches routes that have a next-hop router address specified by the prefix-lists
match ipv6 route-source prefix-list {*name*}	Matches routes that have been advertised by routers at the addresses specified in the prefix-list
match length *min max*	Matches the level 3 length of a packet

continues

(Continued)

Command	Description
match metric *metric-value*	Matches routes with the specified metric
match route-type {**internal** | **external** [**type-1** | **type-2**] | **level-1** | **level-2**}	Matches OSPF, EIGRP, or IS-IS routes of the specified type
match tag *tag-value* [...*tag-value*]	Matches routes with the specified tags
ipv6 prefix-list *list-name* [**seq** *seq-number*] {**deny** *ipv6-prefix/prefix-length* | **permit** *ipv6-prefix/prefix-length* | **description** *text*} [**ge** *ge-value*] [**le** *le-value*]	Defines an IPv6 prefix list
redistribute *protocol* [*process-id*] {**level-1** | **level-1-2** | **level-2**} [**metric** *metric-value*] [**metric-type** *type-value*] [**match** {**internal** | **external 1**| **external 2**}] [**tag** *tag-value*] [**route-map** *map-tag*] [**weight** *weight*] [**subnets**]	Configures redistribution into a routing protocol and specifies the source of the redistributed routes
set level {**level-1** | **level-2** | **level-1-2** | **stub-area** | **backbone**}	Sets the IS-IS level or the OSPF area into which a matched route is to be redistributed
set default interface *type number* [...*type number*]	Sets the outgoing interface for matched packets when there is no explicit route to the destination
set interface *type number* [...*type number*]	Sets the outgoing interface for matched packets when there is an explicit route to the destination
set ip default next-hop *ip-address* [...*ip-address*]	Sets the next-hop router address for matched packets when there is no explicit route to the destination
set ip next-hop *ip-address* [...*ip-address*]	Sets the next-hop router address for matched packets when there is an explicit route to the destination
set ip precedence *precedence*	Sets the precedence bits in the Type of Service field of matched IP packets
set ip tos *type-of-service*	Sets the TOS bits in the Type of Service field of matched packets
set metric {*metric-value* | *bandwidth delay reliability loading mtu*}	Sets the metric value for a matched route
set metric-type {**internal** | **external** | **type-1** | **type-2**}	Sets the metric type for a matched route being redistributed into IS-IS or OSPF
set next-hop *next-hop*	Sets the next-hop router address for a matched route
set tag *tag-value*	Sets a tag value for a matched route

Review Questions

1 How are route maps similar to access lists? How are they different?

2 What are policy routes?

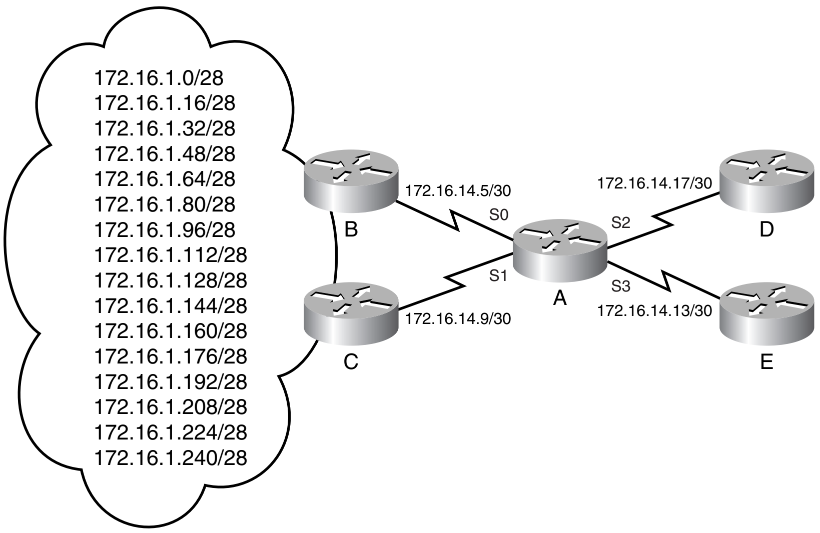
3 What are route tags?

4 In what way do route tags affect routing protocols?

Configuration Exercises

1 Configure policy routes for Router A in Figure 14-14 that forward packets from subnets 172.16.1.0/28 through 172.16.1.112/28 to Router D and forward packets from subnets 172.16.1.128/28 through 172.16.1.240/28 to Router E.

Figure 14-14 *The network for Configuration Exercises 1 through 3.*



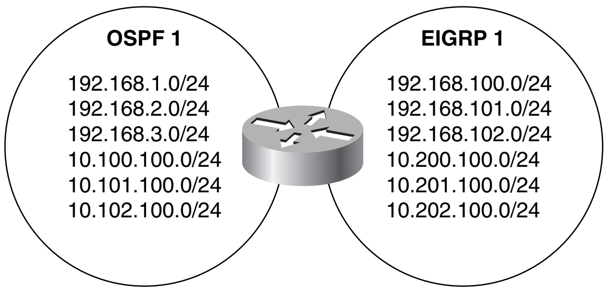
2 Configure policy routes for Router A in Figure 14-14 so that packets from subnets 172.16.1.64/28 through 172.16.1.112/28 are forwarded to Router D if they are received from Router C. If packets from the same subnets are received from Router B, forward them to Router E. All other packets should be forwarded normally.

3 Configure policy routes for Router A in Figure 14-14 that will forward any packets destined for subnets 172.16.1.0/28 through 172.16.1.240/28, sourced from an SMTP port, to Router C. Route any other UDP packets destined for the same subnets to Router B. No other packets should be forwarded to Routers C or B by either the policy routes or the normal routing protocol.

4 The OSPF and EIGRP configuration for the router in Figure 14-15 is

```
router eigrp 1
network 192.168.100.0
!
router ospf 1
network 192.168.1.0 0.0.0.255 area 16
```

Figure 14-15 *The router for Configuration Exercises 4 and 5.*



Configure the router to redistribute internal EIGRP routes into OSPF as E1 routes with a metric of 10 and to redistribute external EIGRP routes into OSPF as E2 routes with a metric of 50. Of the networks and subnets shown in the EIGRP domain, all should be redistributed except 10.201.100.0/24.

5 Configure the router in Figure 14-15 to redistribute internal OSPF routes into EIGRP with a lower delay than external OSPF routes. Allow only the three Class C networks shown in the OSPF domain to be redistributed.

Troubleshooting Exercise

1 Given the following configuration:

```
interface TokenRing1
ip address 192.168.15.254 255.255.255.0
ip policy route-map Ex1
!
access-list 1 permit 192.168.0.0 0.0.255.255
access-list 101 permit host 192.168.10.5 any eq telnet
!
route-map Ex1 permit 5
match ip address 1
set ip next-hop 192.168.16.254
```

```
!
route-map Ex1 permit 10
match ip address 101
set ip next-hop 192.168.17.254
```

The intention is to policy route all packets whose source address prefix is 192.168.0.0 to 192.168.255.255. The exception is that packets originating from the Telnet port of host 192.168.10.5 should be forwarded to 192.168.17.254. There are two errors in this configuration that are preventing the policy route from working correctly. What are they?