In this chapter:

CHAPTER 8
Administrative Controls

Chapter 1 classified the routing processes running on every router into three categories: (1) the processes responsible for running the routing protocols (EIGRP, OSPF, etc.); (2) the processes that take routing information from these routing protocols and build the routing table, exchange (redistribute) routing information between routing protocols, and filter routing information between peers; and (3) the processes involved with the forwarding of IP packets.

The bulk of this book is devoted to the description of routing protocols, which constitute the first set of processes. The third set of processes uses the rules of longest prefix match and classful versus classless route-lookup behavior, which I have already discussed at length.

The second set of processes is constituted of the controls that an administrator can exert over the routing process. This chapter describes these controls, which span all routing protocols. Instead of discussing these controls separately in the context of each routing protocol, I have reserved this discussion for this chapter, where I will talk about these controls just once.

The most common administrative control is the filtering of routing information between peers, over interfaces, or between routing protocols. Routing information may be filtered for any number of reasons: to stop sending routing updates to servers, to partition the network, to prevent routing loops, etc. These administrative controls are described in the section "Filter Routing Information."

If a router learns a route via multiple sources, it uses a default hierarchy of administrative distances to assign preference to one source over another (as discussed in

Chapter 1). This hierarchy can be adjusted as described later, in "Rate the Trustworthiness of a Routing Information Source." This control is used to create *floating static routes*.

For various reasons, networks often run multiple routing protocols. This requires the exchange of routing information between the routing protocols so as to present a cohesive, integrated network. The controls for this exchange of routing information are described in the section "Redistribute Routes."

When a router knows of multiple equal-cost paths to a given destination, it will install all the paths in its routing table, up to a default maximum. This default maximum number of paths to a single destination can be adjusted as described in "Maximum Number of Paths."

Filter Routing Information

The administrative control over updates entering and leaving a routing process has common elements across all routing protocols. As Figure 8-1 shows, both incoming and outgoing updates may be filtered.

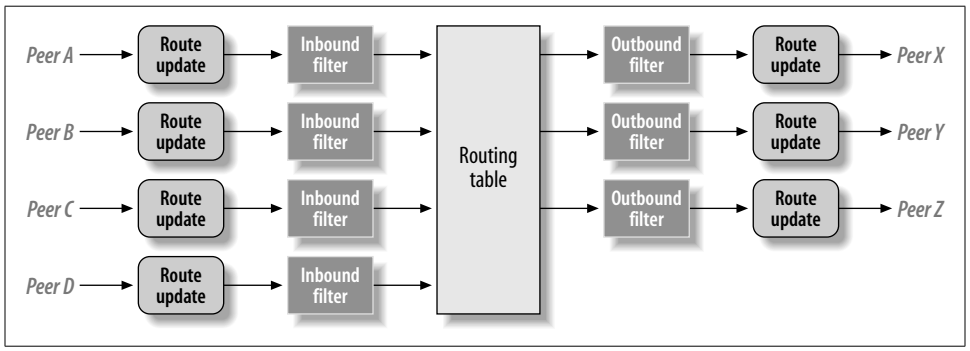

Figure 8-1. Filter incoming and outgoing updates

There are several approaches to filtering routing information. We will describe these approaches, along with a potential use of each, in the following sections.

Block All Updates on an Interface

The *passive-interface* command blocks all updates from being sent on the specified interface(s). The syntax of the command (in router configuration mode) is:

```
passive-interface type number
```

Consider the example of router *R*:

```
hostname R
...
interface Ethernet0
```

```
description * EIGRP Network *
...
interface Ethernet1
description * RIP Network *
...
router rip
passive-interface ethernet0
...
router eigrp 109
passive-interface ethernet1
```

This command allows router *R* to participate in a RIP process with routers connected on *Ethernet1* and an EIGRP process with routers on *Ethernet0*.

This command cannot be used with BGP.

Filter the Routes Sent out in Updates

The *distribute-list out* command may be used to filter the routes announced in updates to peers. The syntax of the command (in router configuration mode) is:

```
distribute-list access-list-number out [interface-name | routing-process]
```

There are two options: updates can be filtered when sent out of an interface or when redistributed between routing processes. An access list (as specified by *access-list-number*) is used in either case to specify the routes to be permitted or denied.

The following is an example of the first option. Router *R* is applying access list 1 to filter the RIP updates being sent out on *Ethernet1*:

```
hostname R
...
interface Ethernet1
description * RIP Network *
...
router rip
distribute-list 1 out interface Ethernet1
```

Our second example shows how routes can be filtered when conveyed to another routing protocol. Router *R* is running EIGRP and RIP. The routes learned from RIP are being redistributed into EIGRP. Access list 2 specifies that networks `146.100.0.0` and `11.0.0.0` are permitted to cross over from RIP into EIGRP; all other network numbers are blocked.

```
hostname R
...
interface Ethernet0
description * EIGRP Network *
...
interface Ethernet1
description * RIP Network *
...
router rip
passive-interface ethernet0
```

```
...
router eigrp 109
redistribute rip
distribute-list 2 out rip
...
access-list 2 permit 146.100.0.0
access-list 2 permit 11.0.0.0
```

Filter the Routes Received in Updates

It is not possible to control the routes being advertised by a peer, but it is possible to restrict the routes that are installed in the routing table. You can do so by applying the *distribute-list in* command to a routing process, using an access list to specify the routes to be permitted or denied:

```
distribute-list access-list-number in [interface-name]
```

In the following example, router *R* will not install `146.100.0.0` when it is received in an EIGRP update on *Ethernet0*:

```
hostname R
...
interface Ethernet0
description * EIGRP Network *
...
interface Ethernet1
description * RIP Network *
...
router rip
passive-interface ethernet0
...
router eigrp 109
 distribute-list 3 in Ethernet0
...
access-list 3 deny 146.100.0.0
access-list 3 permit 0.0.0.0 255.255.255.255
```

Apply an Offset to a Routing Metric

When there are several paths to a destination and one path is less desirable, an offset may be applied to (increase) the metric on the less favorable path. This subcommand applies to RIP and IGRP updates only:

```
offset-list [access-list-number] {in | out} offset [interface-type number]
```

The command must specify whether the offset applies to incoming or outgoing updates (using the *in* or the *out* keywords).

You can associate an access list with the command to specify the routes to which the offset applies. Optionally, you can also specify an interface type and number to indicate that the offset applies only to updates sent/received from a specific interface.

Rate the Trustworthiness of a Routing Information Source

As I discussed in detail in Chapter 1, administrative weight is the trustworthiness of a routing information source. When a route is known via more than one source, the source with the lower administrative distance is installed in the routing table. The following subcommand:

```
distance weight [[ip-source-address ip-address-mask] [access-list-number]]
```

is used to specify administrative weight. Without the options, the command applies to all routes known via the routing protocol. Thus, in the following example, a distance of 10 is attached to all RIP-derived routes and a distance of 20 is attached to all EIGRP-derived routes:

```
hostname R
...
router rip
distance 10
...
router eigrp 109
distance 20
```

You can use the optional filters to attach an administrative weight to only the routes derived from routing sources that pass the filters. Thus, the *distance* command in the following code:

```
router rip
network 11.0.0.0
distance 160 11.1.1.0 0.0.0.255
```

attaches a distance of 160 to RIP routes derived from sources in the IP address range `11.1.1.0` through `11.1.1.255`. The use of the optional filters in the *distance* command is discouraged; unless it is carefully planned, it can cause problems with routing loops.

Redistribute Routes

Ideally, you should run only a single IGP in any given network. However, as networks evolve they often end up running multiple routing protocols. How does this happen? After all, shouldn't the routing engineer select one routing protocol and stick with it as the network grows?

Consider a network running RIP. The network is to be extended to support a new business area, and the routing engineers decide not to use RIP for the extension because of its long convergence times. Instead, they deploy EIGRP on the extension, while continuing to use RIP on the *legacy* network.

In another scenario, consider two corporations that merge and ask their network engineers to join their networks. One network may have been running OSPF and the other IGRP. The two routing domains in this scenario are described in Figure 8-2.

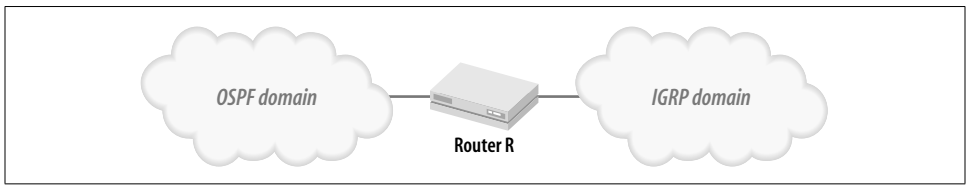

Figure 8-2. A network with two routing domains

For the network in Figure 8-2 to remain cohesive—i.e., for end-stations in one domain to reach end-stations in the other domain—router *R* must perform some kind of "translation," taking the routes from the OSPF domain and conveying them into IGRP, and vice versa.

This "translation" of routing information from one domain (or routing protocol) to another is known as *route redistribution*. Note that route redistribution is a one-way translation of routing information from one routing protocol to another. The two-way translation of routing information from one routing protocol to another and vice versa is referred to as *mutual redistribution*.

The translation of routing information during route redistribution is akin to the translation of texts between languages, such as French and English. A French-to-English translator must know both languages; a router doing route redistribution must run both routing protocols. Thus, router *R* in Figure 8-2 must run IGRP and OSPF if it is to redistribute between IGRP and OSPF.

Every translation between languages leads to a loss in nuance, feeling, and depth: how can you ever translate "Shalom"? Route redistribution usually results in some loss of routing information: how can an OSPF IA route with a metric of 1,575 be represented in IGRP? This can lead to problems. In fact, the careless use of route redistribution commands is a sure recipe for disaster. In the following section, I'll describe the commands used in route redistribution and some common pitfalls.

How to Redistribute

This section describes the Cisco IOS commands used in route redistribution. Route redistribution commands allow the network engineer to (a) specify which routing protocol to redistribute into which other protocol, (b) specify which routes to translate between the routing protocols, and (c) specify the attributes of the routes in the new routing protocol. So, for instance, if the routes are being imported into OSPF, it should be possible to specify that the redistributed routes should be AS-external type 2 with a metric of 100.

The *redistribute* command appears as follows in router configuration mode:

```
redistribute protocol [process-id] {level-1 | level-1-2 | level-2}
[metric metric-value] [metric-type type-value]
[match internal | external type-value] [route-map map-tag]
[weight weight] [subnets]
```

The *protocol* keyword specifies the source protocol from which routes are being redistributed. The source protocol may be bgp, igrp, ospf, static, connected, rip, isis, etc.

The *process-id* specifies the autonomous system number of the routing process. Note that no *process-id* value is needed for RIP.

The *{level-1 | level-1-2 | level-2}* keyword is used only for isis routes.

The *metric-value* specifies the metric to attach to the redistributed routes. Remember that route metrics do not translate between routing protocols. It is usual to assign a fixed metric to all routes when redistributing them into another routing protocol. In the upcoming example, a metric of 100 is attached to the routes redistributed from RIP. If a metric-value is not specified in the command, a default value of 0 is assumed.

The *type-value* applies to OSPF, which defines two types of external routes: type 1 and type 2.

The *match* keywords apply only when OSPF routes are being redistributed into another protocol. The keywords specify which types of OSPF routes to redistribute: *internal*, *external*, etc.

You can use a *route map* to control details of the redistribution or to specify the attributes of routes when translating between protocols. In the following example, RIP is being redistributed into OSPF. The route map *only-2-hop-routes* is used to enforce the policy that only two-hop routes be redistributed into OSPF. These routes are accepted into OSPF with a metric of 100 and as type 1 external routes.

```
router ospf 2
redistribute rip route-map only-2-hop-routes
!
route-map only-2-hop-routes permit
match metric 2
set metric-type 1
set metric 100
```

The *weight* keyword is used only when redistributing into BGP, to specify the Weight attribute of the redistributed route.

The *subnets* keyword is used when redistributing into OSPF to specify which routes to import from the specified protocol.

If you are experienced with RIP, you may have noticed an exception. RIP automatically redistributes all static routes with a metric of 1. In other words, static routes appear to RIP to be directly connected.

Since IGRP's metric is a vector, the specification of the metric of the redistributed routes takes on a variation:

```
default-metric bandwidth delay reliability loading mtu
```

Many Pitfalls...

Route redistribution exposes the network to the risks of routing anomalies. Consider the network in Figure 8-3. Routers *R1* and *R2* perform mutual redistribution, exchanging routes between the two domains. Let's say that domain 1 is using EIGRP, and domain 2 is using OSPF. *R1* and *R2* are redistributing EIGRP routes into OSPF and OSPF routes into EIGRP. The network engineer has selected two routers for this redistribution for redundancy.

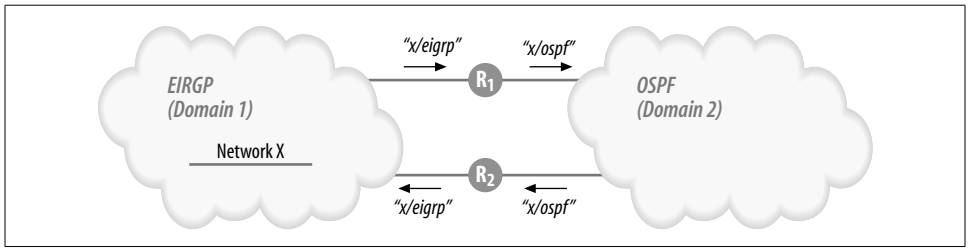

Figure 8-3. A routing anomaly as a result of route redistribution

Now, consider network X in domain 1. *R1* advertises X into domain 2. *R2* learns X (via OSPF) and redistributes this information into domain 1. It may appear to routers in domain 1 that X is reachable via *R2*! Such routing anomalies sometimes take the shape of routing loops.

Other routing anomalies include nonoptimal routing, "black holes," and missing routes. These problems are often a result of carelessly redistributing routes without paying attention to the details of the differences between routing protocols. For example, RIP Version 1 is a classful protocol and cannot carry subnet mask information. If an OSPF domain is using VLSM, how will the OSPF routes look to RIP? Or, if you are redistributing IGRP into OSPF, how should the IGRP metric be translated into the OSPF metric?

... and a Couple of Strategies

Do not run multiple routing protocols!

If you have to run multiple routing protocols (and use route redistribution), there are a few guidelines to follow:

1. Do not run multiple routing protocols on overlapping topologies.
2. Redistribute in one direction only.

3. Use distribute lists to control which routes to accept—there are examples of this in the earlier section "Filter Routing Information."
4. Avoid mutual redistribution (a common cause of routing loops). When mutual redistribution seems necessary, see if you can redistribute in one direction and use a default route in the opposite direction. Thus, in the example in Figure 8-3, redistribute EIGRP into OSPF and (in place of redistributing OSPF into EIGRP) define a default route in EIGRP pointing to the OSPF domain.
5. Avoid mutual redistribution at multiple routers like the plague.
6. Whatever redistribution strategy you decide to undertake, try to test it in a lab environment before inflicting it on a production network.

Maximum Number of Paths

If the path to a destination is known via more than one equal-cost path, the routing table will install multiple paths in the routing table and balance traffic over those paths. To override the default maximum number of paths (which is 6 for the current IOS releases) that can be installed in the routing table, use the following command in router configuration mode:

```
maximum-paths maximum
```

Note that this command does not apply to BGP, which installs only one route to a destination.

Summing Up

The administrative controls described in this chapter are useful tools for the network engineer to have in her back pocket. To ensure that the network engineer is not being constantly called on to use these tools to "patch" the network, it is important that the overall routing architecture be simple and elegant. Toward this end, it may be prudent for the network engineer to pay attention to the following:

- Hierarchy of design, reflecting a hierarchy of IP addresses
- Route summarization, reducing the size of the routing table
- Using only a small number of routing protocols in the network, as it is difficult to be familiar with the vagaries of several different routing protocols or to handle multiple route redistributions

Further, since all IOS versions exhibit bugs, limit the number of IOS versions used in the network.