Optimal Routing Design

Techniques for optimizing large-scale IP routing operation and managing network growth

Russ White, CCIE® No. 2653
Don Slice, CCIE No. 1929
Alvaro Retana, CCIE No. 1609

ciscopress.com

Optimal Routing Design

Russ White, CCIE No. 2635
Don Slice, CCIE No. 1929
Alvaro Retana, CCIE No. 1609

Cisco Press
800 East 96th Street
Indianapolis, IN 46240 USA

because of the redistribution from EIGRP. This means that the alternative path is working fine. Unfortunately, because the hop count on the redistribution is set to 1 because of the default metric, when Router E receives the real route back from Router B, it does not use it because the one it received from Router D is better. This is not what you want to happen.

This is a classic redistribution routing loop. How do you solve it? The easiest thing to do is to filter the destinations that are redistributed from RIP into EIGRP and from EIGRP into RIP.

Using Distribute Lists to Prevent Redistribution Routing Loops

The first, and simplest, way to handle this problem is to set up a distribute list specifically blocking the routes that you do not want to redistribute. For example, on Router D, you could build the distribute list in Example 3-25.

Example 3-25 *Using a Distribution List to Block Redistribution Routing Loops*

```
access-list 10 deny 172.16.20.0 0.0.0.255
access-list 10 permit any
!
router rip
 redistribute eigrp 100
 distribute-list 10 out serial 0
```

Assuming that Serial 0 is the link between Router D and Router E, this resolves the problem. RIP does not advertise the 172.16.20.0/24 route from Router D to Router E. If you have more than one connection back into the RIP side of the network, it can be difficult to manage the distribution lists that must be maintained.

Using Route Maps to Prevent Redistribution Routing Loops

An alternative to using a distribute list is to configure a route map on Router D, as demonstrated in Example 3-26.

Example 3-26 *Using a Route Map to Stop a Redistribution Routing Loop*

```
access-list 10 deny 172.16.20.0 0.0.0.255
access-list 10 permit any
!
route-map kill-loops permit 10
 match ip address 10
!
router rip
 redistribute eigrp 100 route-map kill-loops
```

This configuration allows only those networks that are permitted by access list 10 to be redistributed into RIP. This has the same effect as the distribute list used in the preceding solution, but it applies the filter in the redistribution rather than in the advertisement to Router D.

Another alternative is to match all external EIGRP routes in the route map, as demonstrated in Example 3-27.

Example 3-27 *Using a Route Map to Filter External Routes*

```
route-map kill-loops deny 10
 match route-type external
route-map kill-loops permit 20
```

However, this approach also destroys any external EIGRP routes that are learned from a protocol other than RIP. In other words, it prevents external destinations elsewhere in the EIGRP network from being reached by the hosts that are attached on the RIP side of the network.

Using Prefix Lists to Prevent Redistribution Routing Loops

In addition to using distribute lists and route maps to troubleshoot redistribution routing loops, you can use prefix lists. For example, you can configure Router D with the prefix lists in Example 3-28.

Example 3-28 *Using Prefix Lists to Prevent Redistribution Routing Loops*

```
ip prefix-list loop-list 10 deny 172.16.20.0/24
ip prefix-list loop-list 20 permit 0.0.0.0/0 le 32
!
route-map kill-loops permit 10
 match prefix-list loop-list
!
router rip
 redistribute eigrp 100 route-map kill-loops
```

Prefix lists allow you to match based on prefix length (the subnet mask) and the actual prefix (destination network). Many possibilities for filtering exist when this application is considered, but they are not covered here.

Setting the Administrative Distance to Troubleshoot Redistribution Routing Loops

Whereas all the previous mechanisms rely on the configuration (and maintenance) of an access list to prevent a redistribution routing loop, setting the administrative distance of all external routes learned by Router D from Router A does not rely on access lists. You can configure this technique using the **distance** command. On Router D, you would configure the following:

```
router eigrp 100
 distance 255 172.16.21.1 0.0.0.0
```

If the Router A address is 172.16.21.1, Router D assigns an administrative distance of 255 to any routes that it receives from Router A. A route that has an administrative distance of 255 is

never inserted into the routing table; therefore, it is not redistributed into RIP from EIGRP. (Redistribution always occurs from the routing table rather than any private databases that the various routing protocols use.)

The only problem with this approach is that Router D refuses all routes learned from Router A, including legitimate ones. You can remedy this by adding the access list back into the equation, as demonstrated in Example 3-29.

Example 3-29 *Using the* **distance** *Command with an Access List to Block Redistribution Loops*

```
access-list 10 permit 172.16.20.0 0.0.0.255
!
router eigrp 100
 distance 255 172.16.21.1 0.0.0.0 10
```

By providing an access list that identifies a particular range of addresses and blocks all others from this neighbor, you can accomplish slightly more selective filtering.

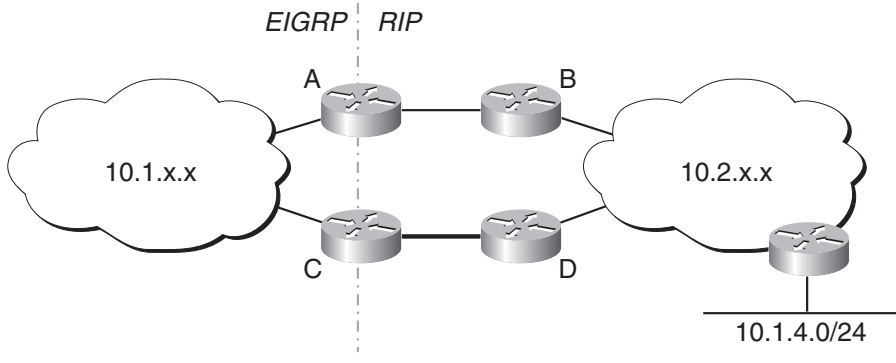
One additional limitation of this approach is that the **distance** command is applied to both internal and external routes. Therefore, if you are trying to limit the filtering to stop the receipt of external routes, you cannot use the **distance** command to accomplish it.

Using External Flags to Prevent Redistribution Routing Loops

All of the previously mentioned troubleshooting methods work, but they require either configuring a list of networks or removing the alternative route through the other protocol as a possible backdoor route in the case of failure. Tagging EIGRP externals to block routing loops resolves these two problems and is fairly straightforward to configure.

Connecting Router A to Router B and Router C to Router D has recently merged the two networks in Figure 3-24. At some point in the future, the network administrators intend to replace RIP with EIGRP; for now, they are redistributing between RIP and EIGRP on Routers A and C.

Figure 3-24 *Complex Redistribution Routing Loop*



This setup produces a classic redistribution routing loop:

1. Router B learns about some destination, such as 10.1.4.0/24, through RIP. Then it advertises this route to Router A.
2. Router A redistributes this route into EIGRP and advertises it to Router C.
3. Router C redistributes this route back into RIP and advertises it to Router D.
4. Router D advertises the route back to Router B (possibly with a better metric than Router B learned in the original advertisement).

Almost all of the EIGRP network in this figure uses addresses from the 10.1.0.0/16 address space, and almost all of the RIP network uses addresses from the 10.2.0.0/16 address space. However, some exceptions exist, such as the 10.1.4.0/24 network.

If it were not for the exceptions, this redistribution routing loop would be easy to resolve. You would simply prevent Router A and Router C from advertising routes in the 10.2.0.0/16 address range to Router B and Router D and prevent Router B and Router D from advertising routes in the 10.1.0.0/16 address range to Router A and Router C. Distribution lists combined with summarization would make this configuration easy.

Because of the exceptions, though, preventing this redistribution routing loop is more difficult. You could build distribution lists around the subnets present on each side and apply them on Router A, Router B, Router C, and Router D, but this adds some serious administrative overhead if many exceptions exist. Specific distribution lists would also require modification for each new exception added.

It is easier to use an automatic method to flag the routes learned through RIP on Router A and Router C. Then you can prevent any route that is flagged from being redistributed back into RIP. For example, Router A still learns about the 10.1.100.0/24 network through EIGRP and advertises this destination to Router B through RIP.

Router B still advertises 10.1.4.0/24 to Router A, which redistributes it into EIGRP and advertises it to Router C. However, Router A flags this route as coming from the RIP domain so that Router C does not advertise it back into RIP. Using some sort of tag like this means that adding a new network in the RIP AS should not require reconfiguration on the routers that are doing the redistribution. This type of routing loop is a good use for EIGRP administrator tags.

Administrator tags are applied and matched using route maps. On Router A and Router C, you create the route maps and then apply them to the redistribution between EIGRP and RIP by issuing the commands in Example 3-30.

Example 3-30 *Setting Administrative Tags on Redistribution*

```
route-map setflag permit 10
  set tag 1
route-map denyflag deny 10
  match tag 1
route-map denyflag permit 20
```

The **setflag** route map sets the administrator tag on any route to 1, whereas the **denyflag** route map denies routes with a flag of 1 and permits all others. On Router A and Router C, you apply these route maps to the redistribution between EIGRP and RIP by issuing the commands in Example 3-31.

Example 3-31 *Applying Tag Filtering on Redistribution*

```
router eigrp 4000
  redistribute rip route-map setflag
router rip
  redistribute eigrp 4000 route-map denyflag
```

As routes are redistributed from RIP to EIGRP, the **setflag** route map is applied, setting the EIGRP administrative tag to 1. As the routes are redistributed from EIGRP to RIP, the administrative tag is checked; if it is 1, the route is denied so that it is not redistributed.

Case Study: Retransmissions and SIA

Two timers that can interact in EIGRP to cause an SIA route in EIGRP are the SIA timer and the hold timer between two peers. How do these two relate? This section examines the two timers independently and then looks at how they interact.

The Hold Timer

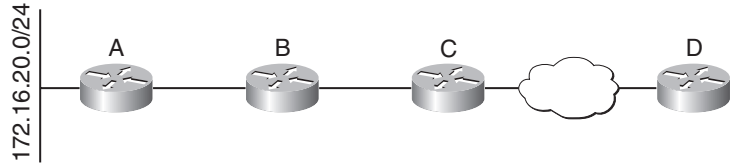
The obvious use for the hold timer is to determine how long to hold up a neighbor relationship without hearing EIGRP hellos. Each time a router receives a hello packet from a neighbor, it resets the hold timer to the hold time contained in the hello packet and decrements it once for each second that passes.

After the hold timer reaches zero, the neighbor is assumed dead. All paths through that neighbor are marked unusable (DUAL is run over these destinations to determine if the route needs to go active), and the neighbor is marked down.

However, the hold timer is also used by the EIGRP reliable transport mechanism as an outer bound on how long to wait for a neighbor to acknowledge the receipt of a packet. As mentioned in Appendix A, EIGRP attempts to retransmit 16 times or until retransmission has been occurring for as long as the hold timer, whichever is longer.

In the network depicted in Figure 3-25, assume that the Router D hold timer is 240 seconds. (Ignore the Hello timer because these are separate timers.)

Figure 3-25 *Interactions Between Hold Timers and SIA Timers*



If Router C sends a packet to Router D, and Router D does not acknowledge the packet, Router C continues retransmitting until it has retransmitted 16 times. Then Router C checks to see if it has been retransmitting for 240 seconds. If it has not, Router C continues sending the packet until it has been retransmitting for 240 seconds. After Router C has attempted retransmission for 240 seconds, it assumes that Router D is never going to answer and clear its neighbor relationship.

SIA Timer

The other timer that you need to concern yourself with is the SIA timer because it determines how long a query can be outstanding before the route is declared SIA and the neighbor relationship with the router that has not answered is torn down and restarted.

Prior to the SIA enhancements explained in the section "Enhanced EIGRP Active Process," the active timer is, by default, 3 minutes (although there has been talk of changing it). This means that a router waits 3 minutes after it has declared a route active until it decides that any neighbor that has not replied for this active route has a problem and restarts the neighbor.

Going back to Figure 3-25, this means that if Router A loses its connection to 172.16.20.0/24, it sends a query to Router B. If it does not receive a reply to that query within 3 minutes, it restarts its neighbor relationship with Router B. Note that two completely different things are being discussed here:

- How long to wait before getting an acknowledgement for a packet
- How long to wait for a reply to a query

Interaction Between the Hold Timer and the SIA Timer

You can work through an example of how these two timers interact. Assume that Router A in Figure 3-25 loses its connection to 172.16.20.0/24. Because it has no other paths to this destination, it marks the route as active and sends Router B a query.

Router B acknowledges the query and sends a query to Router C; Router C, in turn, acknowledges the query and sends a query to Router D. Router D, for some reason, never acknowledges the query. Router C begins retransmitting the query to Router D. It attempts to do so until it has retransmitted for the length of the hold timer.

For the entire time that Router C is trying to get an acknowledgement from Router D, the Router A SIA timer is running. Because the SIA timer is 3 minutes, and the Router D hold timer is 4 minutes, it is safe to assume that the Router A SIA timer will go off before Router C gives up retransmitting the query to Router D and clears the neighbor relationship.

Therefore, Router A registers an SIA and clears its neighbor relationship with Router B. It is important to remember when designing your network that the hold timer for any given link should never be more than or equal to the SIA timer for the entire network.