Build a network router on Linux

Zebra offers a competent substitute for dedicated Cisco routers

Level: Intermediate

Dominique Cimafranca (dom-at-sketches.kom.ph), IT Specialist, IBM
Rex Young (youngrv-at-c-cubeservices.com), Network Manager, C-Cube

08 Oct 2003

Zebra is open source TCP/IP routing software that is similar to Cisco's Internetworking Operating System (IOS). Flexible and powerful, it can handle routing protocols such as Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), and all of their various flavors. This article shows how our authors set up Zebra and used it to manage routes dynamically in conjunction with real Cisco hardware.

Related content:
Setting up a Local Area Network
Building a wireless access point on Linux

Subscriptions:
dW newsletters
dW Subscription (CDs and downloads)

Dynamic and robust routing is so essential to the workings of the Internet that any fledgling internetworking engineer must not only understand its concepts but also be able put it into practice in real situations. At the same time, this part of the routing scene is dominated by products from high-end network equipment providers such as Cisco, well outside the means of most individuals: learning would otherwise be limited to school or laboratory environments, with time and availability of the resources as perennial adversaries.

We were faced with such a dilemma when organizing a class on TCP/IP routing. Working with a small test network, we wanted to demonstrate various load-balancing scenarios with Routing Information Protocol (RIP) and Open Shortest Path First (OSPF). However, we were limited by the number of Cisco routers on hand. We did have some PCs available, so we began looking for alternative means to simulate Cisco routers using Linux.

Initially, we attempted to set up our test network using the traditional routed and gated daemons, but we quickly realized that their awkward configuration and limited capabilities created more hindrance than value. We decided to look for more modern means of completing our network and fortunately came upon Zebra.

What is Zebra?

Zebra is TCP/IP routing software that supports BGP-4, BGP-4+, OSPFv2, OSPFv3, RIPv1, RIPv2, and RIPng. It is distributed under the GNU General Public License and runs on Linux as well as other UNIX variants. Zebra is included in most modern distributions as routing software. The latest version, along with documentation, is available at the GNU Zebra Web site (see Resources for a link).

The original Zebra package was written by Kunihiro Ishiguro and Yoshinari Yoshikawa back in 1996. Today, the package is maintained primarily by IP Infusion -- of which Mr. Ishiguro is the CTO -- with the assistance of networking engineers and open source volunteers.

Zebra is unique in its design because it takes a modular approach to the protocols that it manages. Protocols can be enabled or disabled as network requirements dictate.

By far the most useful feature we found with Zebra was its close similarity to the Cisco IOS configuration formats. While there are some differences from IOS, the feel is close enough that network engineers already familiar with IOS will feel very comfortable in the environment.

While Zebra has not yet hit version 1.0 -- the version at the time of this writing is 0.93b -- the quality of the product is adequate for small networks that need a core router. *[Please note that this article was written using version 0.93b. Later versions may need to be installed and configured differently. -Ed.]*

Installing Zebra

Our test platform for Zebra was an old but sturdy ThinkPad X20 running Red Hat Linux 9. The ThinkPad has a built-in Ethernet port, and we added another Ethernet PCMCIA card so it could act as a router. Before proceeding with the Zebra installation, we made sure that both network cards were recognized by Linux and were confirmed to be working.

An RPM for Zebra-0.93b already ships with Red Hat 9. As this was the same version available on the Zebra Web site, we decided to use this instead of downloading and compiling our own. The Zebra RPM installs binaries, scripts, and configuration files as well as the requisite manuals, examples, and documentation files.

Basic Zebra configuration

The zebra daemon is the actual routing manager that controls the other modules; it also provides the primary point of interaction with the user. This was the first thing we needed to configure, and we did it through the /etc/zebra/zebra.conf file.

The Zebra RPM package includes a complete sample configuration file. However, at a minimum, we really only needed to create a /etc/zebra/zebra.conf file containing the following lines:

Listing 1. A minimal Zebra configuration file

```
hostname speedmetal
password zebra
enable password zebra
```

The `hostname` directive specifies the name of the router whenever you enter interactive configuration mode. It can be any label and does not necessarily have to correspond to the hostname of the machine.

The `password` directive specifies the password for logging into the interactive Zebra terminal.

The `enable password` directive specifies the password for a higher level of access to Zebra, when you want to make configuration changes.

Having created the /etc/zebra/zebra.conf file, we could now start the zebra daemon by executing:

```
# service zebra start
```

We were now able to enter the Zebra interactive session by telnetting into port 2601 of our machine.

Listing 2. A sample Zebra session

```
[root@speedmetal zebra]# telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.

Hello, this is zebra (version 0.93b).
Copyright 1996-2002 Kunihiro Ishiguro.

User Access Verification

Password: zebra
speedmetal> enable
Password: zebra
speedmetal# ?
  configure  Configuration from vty interface
  copy       Copy configuration
  debug      Debugging functions (see also 'undebug')
  disable    Turn off privileged mode command
  end        End current mode and change to enable mode.
  exit       Exit current mode and down to previous mode
  help       Description of the interactive help system
  list       Print command list
  no         Negate a command or set its defaults
  quit       Exit current mode and down to previous mode
  show       Show running system information
  terminal   Set terminal line parameters
  who        Display who is on vty
  write      Write running configuration to memory, network, or terminal
speedmetal#
```

Navigating within the interactive terminal is easy. For hints about the available commands, you can press `?` at any time and the options will show on screen. If you're setting up your own Zebra router, this configuration should be very familiar if you have Cisco experience.

At this point, only Zebra was configured and running; none of the other protocols were, as yet. Later, when we get to the meat of the configuration, we'll show you how we did that.

Configuring and using MRLG

The Multi-Router Looking Glass, or MRLG, written by John Frazier of EnterZone, is a Web-based utility that can be used to display the interfaces and routes recognized by Zebra. MRLG is really nothing more than a Web interface to the Zebra shell with a limited set of commands, but in the course of our testing, we found it to be a quick and useful way to display routes. So, before getting to the configuration of the Zebra protocols, we'll show you how we installed it.

MRLG requires the `Net::Telnet` Perl package so that it can communicate with the Zebra shell. Unfortunately, this package is not included as part of the stock Red Hat 9 distribution, so we had to download it (see Resources for a link).

Since MRLG runs as a CGI application, we also needed to install a Web server. If you're trying this yourself, you can use the stock httpd RPM that comes with Red Hat 9.

We copied the mrlg.cgi file from /usr/share/doc/zebra-0.93b/tools to /var/www/cgi-bin. Then, we modified line 36 of mrlg.cgi, changing it from:

```
$url="http://www.sample.com/mrlg.cgi";
```

to:

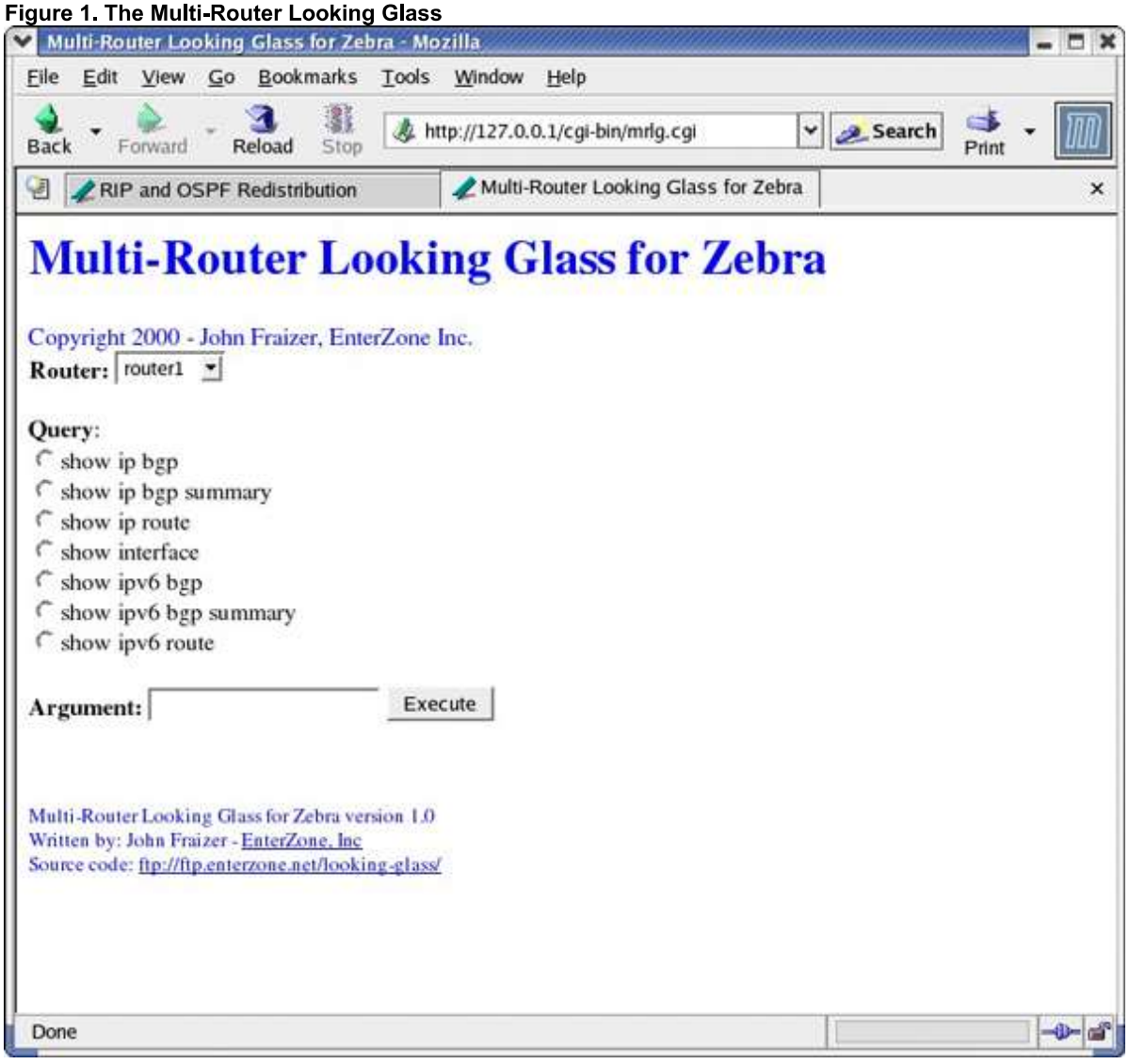
```
$url="http://127.0.0.1/cgi-bin/mrlg.cgi";
```

We also modified the block from lines 168 to 174 so that it read as follows:

```
if ($Form{'router'} eq 'router1')
        {
$server = '127.0.0.1';
$login_pass = 'zebra';
$bgpd = "2605";
$zebra = "2601";
$full_tables=1;
```

To access MRLG, point the browser to http://127.0.0.1/cgi-bin/mrlg.cgi.

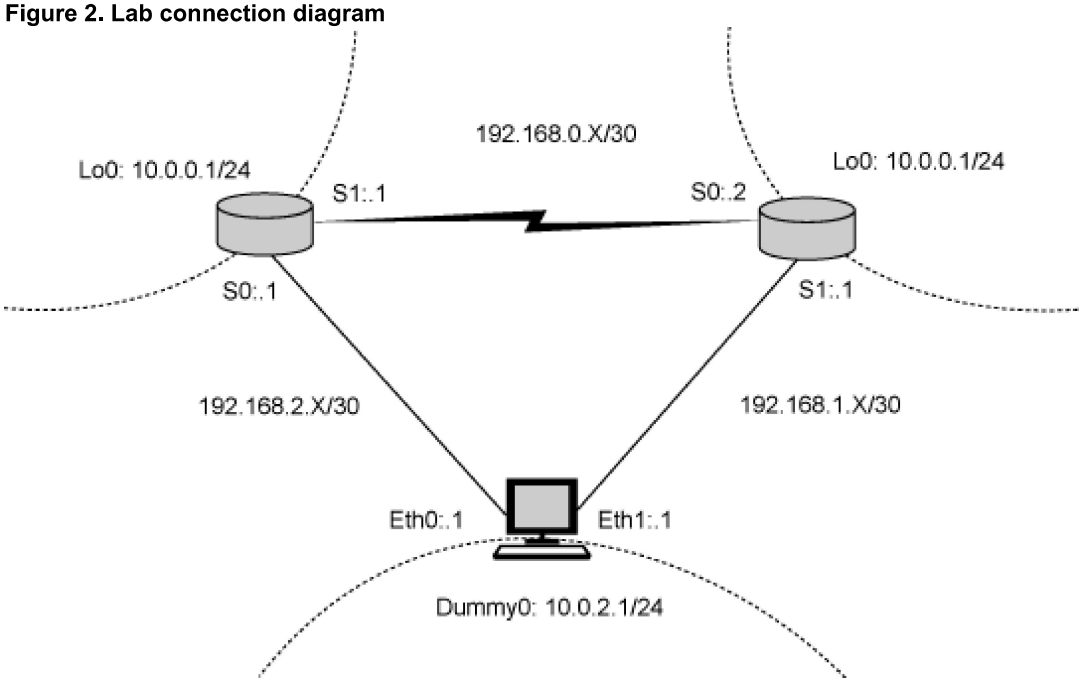
Figure 1. The Multi-Router Looking Glass



Basic lab configuration

Our lab configuration consisted of two Cisco 3620 routers and a ThinkPad X20 with a built-in Ethernet port and a Home-and-Away PCMCIA Ethernet card. The two routers communicated with each other via serial connection, and each router connected to the ThinkPad through Ethernet. This is shown in our connection diagram:

Figure 2. Lab connection diagram



Setting up the interfaces with Zebra

Our first foray into routing with Zebra started with RIP. We installed Zebra on the ThinkPad as described above. Since we needed another network interface on the ThinkPad, we installed a dummy network device like so:

```
# modprobe dummy
# ifconfig dummy0
```

We telnetted into the Zebra port to begin configuration. Our dialog with Zebra followed this sequence:

Listing 3. Configuring IP interfaces

```
User Access Verification

Password: zebra
speedmetal> enable
Password: zebra
speedmetal# configure terminal
speedmetal(config)# interface eth0
speedmetal(config-if)# ip address 192.168.2.1/30
speedmetal(config-if)# quit
speedmetal(config)# interface eth1
speedmetal(config-if)# ip address 192.168.1.1/30
speedmetal(config-if)# quit
speedmetal(config)# interface dummy0
speedmetal(config-if)# ip address 10.0.2.1/24
speedmetal(config-if)# write
Configuration saved to /etc/zebra/zebra.conf
speedmetal(config-if)# end
speedmetal# show run

Current configuration:
!
hostname speedmetal
password zebra
enable password zebra
!
interface lo
!
interface eth0
 ip address 192.168.2.1/30
!
interface dummy0
 ip address 10.0.2.1/24
!
interface eth1
 ip address 192.168.1.1/30
!
!
line vty
```

```
!
end
```

Take note that we did not set the IP addresses on the ThinkPad using the ordinary means; instead, we configured them through Zebra. The settings are kept in the configuration file, /etc/zebra/zebra.conf, so every time the Zebra service starts up, these settings will take effect.

The contents of our zebra.conf file, as modified by Zebra, are:

Listing 4. /etc/zebra/zebra.conf, as modified by Zebra

```
!
! Zebra configuration saved from vty
!   2003/08/20 00:07:51
!
hostname speedmetal
password zebra
enable password zebra
!
interface lo
!
interface eth0
 ip address 192.168.2.1/30
!
interface dummy0
 ip address 10.0.2.1/24
!
interface eth1
 ip address 192.168.1.1/30
!
!
line vty
!
```

We are also able to check the status of the interfaces using MRLG by selecting the default, "router1," choosing the radio button "show interface," and clicking "Execute."

Setting up RIP routing with Zebra

Now that we had set up the network interfaces on our ThinkPad/router, we configured it to work with RIP updates. As we've already mentioned, Zebra implements the routing protocols using separate daemons, so we first had to create a simple configuration file, ripd.conf, for the RIP daemon in /etc/zebra.

Listing 5. A basic /etc/zebra/ripd.conf file

```
hostname speedmetal-rip
password zebra
enable password zebra
```

Then we started the ripd daemon:

```
# service ripd start
```

That done, we were able to configure the RIP daemon by telnetting into port 2602 of our Zebra router.

Listing 6. Configuring RIP

```
User Access Verification

Password: zebra
speedmetal-rip> enable
Password: zebra
speedmetal-rip# configure terminal
speedmetal-rip(config)# router rip
speedmetal-rip(config-router)# network 10.0.0.0/8
speedmetal-rip(config-router)# network 192.168.0.0/16
speedmetal-rip(config-router)# end
speedmetal-rip# show run

Current configuration:
!
hostname speedmetal-rip
password zebra
enable password zebra
```

```
!
interface lo
!
interface eth0
!
interface dummy0
!
router rip
 network 0.0.0.0/0
 network 192.168.0.0/16
!
line vty
!
end
speedmetal-rip# write
Configuration saved to /etc/zebra/ripd.conf
speedmetal-rip#
```

The resulting ripd.conf configuration file is:

Listing 7. Resulting /etc/zebra/ripd.conf file

```
!
! Zebra configuration saved from vty
!   2003/08/19 13:50:30
!
hostname speedmetal-rip
password zebra
enable password zebra
!
interface lo
!
interface eth0
!
interface eth1
!
interface dummy0
!
router rip
 network 10.0.0.0/8
 network 192.168.0.0/16
!
line vty
!
```

Setting up RIP routing on the Cisco routers

To facilitate configuration of the two Cisco 3620 routers, which we named "A" and "B," we configured only the basic settings needed to make the routers run properly. This included setting up the interface IP addresses, the loopback address, and the serial clockrates for proper serial port communication.

Listing 8. Configuring router A

```
Router#config terminal
Router(config)#hostname RouterA
RouterA(config)#int s0/0
RouterA(config-if)#ip address 192.168.0.1 255.255.255.252
RouterA(config-if)#no shut
RouterA(config-if)# interface fastEthernet 0/0
RouterA(config-if)#ip address 192.168.2.2 255.255.255.252
RouterA(config-if)#no shut
RouterA(config-if)#int loopback 0
RouterA(config-if)#ip address 10.0.0.1 255.255.255.0
RouterA(config-if)#end
RouterA#write
```

We followed a similar procedure for router B.

Listing 9. Configuring router B

```
Router#configure terminal
Router(config)#hostname RouterB
RouterB(config)#int s0/0
RouterB(config-if)#ip address 192.168.0.2 255.255.255.252
```

```
RouterB(config-if)#no shut
RouterB(config-if)#int fastEthernet0/0
RouterB(config-if)#ip address 192.168.1.2 255.255.255.252
RouterB(config-if)#no shut
RouterB(config-if)#int loopback 0
RouterB(config-if)#ip address 10.0.1.1 255.255.255.0
RouterB(config-router)#end
RouterB#write
```

Setting up RIP on the 3620 routers is very similar to the commands in Zebra. We accessed both of the 3620s through the console cables and issued the following commands:

Listing 10. Configuring router A for RIP

```
RouterA#conf t
Enter configuration commands, one per line. End with CNTL/Z.
RouterA(config)#router rip
RouterA(config-router)#network 10.0.0.0
RouterA(config-router)#network 192.168.0.0
RouterA(config-router)#network 192.168.2.0
RouterA(config-router)#version 2
RouterA(config-router)#end
RouterA#write
```

And, on router B:

Listing 11. Configuring router B for RIP

```
RouterB#conf t
Enter configuration commands, one per line. End with CNTL/Z.
RouterB(config)#router rip
RouterB(config-router)#network 10.0.1.0
RouterB(config-router)#network 192.168.0.0
RouterB(config-router)#network 192.168.1.0
RouterB(config-router)#version 2
RouterB(config-router)#end
RouterB#write
```

The `router rip` command turns on the RIP process. The `network` command tells the router which networks will be propagated by RIP.

Propagated routes with RIP

Now that the Cisco routers and Zebra were all configured, we checked that the routes were being propagated. From MRLG, we selected "show ip route" and clicked "Execute." This generated the following report:

Listing 12. Zebra reflecting RIP routes

```
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, > - selected route, * - FIB route

R>* 10.0.0.0/24 [120/2] via 192.168.2.2, eth0, 00:11:05
R>* 10.0.1.0/24 [120/2] via 192.168.1.2, eth1, 00:02:08
C>* 10.0.2.0/24 is directly connected, dummy0
K * 127.0.0.0/8 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
R>* 192.168.0.0/30 [120/2] via 192.168.2.2, eth0, 00:11:05
C>* 192.168.1.0/30 is directly connected, eth1
C>* 192.168.2.0/30 is directly connected, eth0
```

The routes learned through RIP are marked with an R.

Note that Zebra now knew about the networks 10.0.0.0/24 and 10.0.1.0/24, as broadcast by router A and router B. We tested this by pinging 10.0.0.1 and 10.0.1.1 from the ThinkPad Zebra router, and by pinging 10.0.2.1 (the ThinkPad dummy interface) from either router.

To test for route failover, we disconnected the network connection from router A, which led to network 10.0.0.0/24. After a total timeout of about two minutes, Zebra learned about the alternate route to network 10.0.0.0/24, going through router B. Note that in the listing below, Zebra reached 10.0.0.0/24 through 192.168.1.2 instead of its previous path.

Listing 13. Zebra reflecting RIP routes

```
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
```

```
       B - BGP, > - selected route, * - FIB route

R>* 10.0.0.0/24 [120/3] via 192.168.1.2, eth0, 00:00:26
R>* 10.0.1.0/24 [120/2] via 192.168.1.2, eth1, 00:06:02
C>* 10.0.2.0/24 is directly connected, dummy0
K * 127.0.0.0/8 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
R>* 192.168.0.0/30 [120/2] via 192.168.1.2, eth1, 00:00:26
C>* 192.168.1.0/30 is directly connected, eth1
C>* 192.168.2.0/30 is directly connected, eth0
```

Why was the total timeout more than two minutes? The default timeout for RIP is 30 seconds, but the RIP protocol specifies three retries (total of 90 seconds) before it determines a route is invalid and another period for flushing the invalid route (another 240 seconds). RIP has generally been known to be slow to responding to connection failures, and this behavior is clearly demonstrated here.

Here is the output showing router A's routing table before failover occurred.

Listing 14. Router A's routing table before failover

```
RouterA#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

     10.0.0.0/24 is subnetted, 3 subnets
R       10.0.2.0 [120/1] via 192.168.2.1, 00:00:11, FastEthernet0/0
C       10.0.0.0 is directly connected, Loopback0
R       10.0.1.0 [120/1] via 192.168.0.2, 00:00:18, Serial0/0
     192.168.0.0/30 is subnetted, 1 subnets
C       192.168.0.0 is directly connected, Serial0/0
     192.168.1.0/30 is subnetted, 1 subnets
R       192.168.1.0 [120/1] via 192.168.0.2, 00:00:18, Serial0/0
                    [120/1] via 192.168.2.1, 00:00:11, FastEthernet0/0
     192.168.2.0/30 is subnetted, 1 subnets
C       192.168.2.0 is directly connected, FastEthernet0/0
```

And after failover:

Listing 15. Router A's routing table after failover

```
RouterA#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

     10.0.0.0/24 is subnetted, 3 subnets
R       10.0.2.0 [120/2] via 192.168.0.2, 00:00:09, Serial0/0
C       10.0.0.0 is directly connected, Loopback0
R       10.0.1.0 [120/1] via 192.168.0.2, 00:00:09, Serial0/0
     192.168.0.0/30 is subnetted, 1 subnets
C       192.168.0.0 is directly connected, Serial0/0
     192.168.1.0/30 is subnetted, 1 subnets
R       192.168.1.0 [120/1] via 192.168.0.2, 00:00:09, Serial0/0
     192.168.2.0/30 is subnetted, 1 subnets
R       192.168.2.0 [120/2] via 192.168.0.2, 00:00:10, Serial0/0
```

Setting up OSPF routing with Zebra

Having done RIP, we moved on to OSPF routing. Although OSPF and RIP can be used together, we preferred to work with a simpler configuration and used OSPF exclusively. To disable RIP from Zebra, we simply shut down the ripd service.

```
# service ripd stop
```

As before, we started with a basic configuration file for OSPF, in this case, ospfd.conf, still in /etc/zebra.

Listing 16. A basic /etc/zebra/ospfd.conf file

```
hostname speedmetal-ospf
password zebra
enable password zebra
```

Then, we started the OSPF service:

```
# service ospfd start
```

Configuring OSPF is actually simpler than configuring RIP: at the most basic, we only need to tell OSPF to broadcast all the routes it knows.

The port for OSPF configuration is 2604.

Here was our dialog for OSPF configuration.

Listing 17. OSPF configuration dialog

```
[root@speedmetal zebra]# telnet 127.0.0.1 2604
User Access Verification

Password: zebra
speedmetal-ospf> enable
Password: zebra
speedmetal-ospf# configure terminal
speedmetal-ospf(config)# router ospf
speedmetal-ospf(config-router)# network 0.0.0.0/0 area 0
speedmetal-ospf(config-router)# end
speedmetal-ospf# write
Configuration saved to /etc/zebra/ospfd.conf
speedmetal-ospf# show run

Current configuration:
!
hostname speedmetal-ospf
password zebra
enable password zebra
!
!
router ospf
 network 0.0.0.0/0 area 0
!
line vty
!
end
speedmetal-ospf#
```

The resulting configuration file, `ospfd.conf`, was:

Listing 18. /etc/zebra/ospfd.conf as modified by Zebra

```
!
! Zebra configuration saved from vty
!   2003/08/19 14:22:17
!
hostname speedmetal-ospf
password zebra
enable password zebra
!
!
!
interface lo
!
interface eth0
!
interface eth1
!
interface dummy0
!
```

```
router ospf
 network 0.0.0.0/0 area 0
!
line vty
!
```

Setting up OSPF on the Cisco routers

To remove RIP from the routers and add OSPF, we executed the following commands:

Listing 19. Removing RIP and adding OSPF

```
RouterA#conf term
RouterA(config)no router rip
RouterA(config)#router ospf 100
RouterA(config-router)#network 0.0.0.0 255.255.255.255 area 0
RouterA(config-router)end
```

We performed the same steps for both router A and router B.

Propagated routes with OSPF

Our MRLG report looked like this:

Listing 20. Zebra reflecting OSPF routes

```
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, > - selected route, * - FIB route

O>* 10.0.0.0/32 [110/11] via 192.168.2.2, eth0, 00:00:01
O>* 10.0.1.1/32 [110/11] via 192.168.1.2, eth1, 00:02:53
O   10.0.2.0/24 [110/10] is directly connected, dummy0, 00:03:31
C>* 10.0.2.0/24 is directly connected, dummy0
K * 127.0.0.0/8 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 192.168.0.0/30 [110/58] via 192.168.2.2, eth0, 00:00:01
                            via 192.168.1.2, eth1, 00:00:01
O   192.168.1.0/30 [110/10] is directly connected, eth1, 00:03:21
C>* 192.168.1.0/30 is directly connected, eth1
O   192.168.2.0/30 [110/10] is directly connected, eth0, 00:03:31
C>* 192.168.2.0/30 is directly connected, eth0
```

Note that the routes to 10.0.0.1/32 and 10.0.1.1/32 are marked with O, indicating that they were learned through OSPF.

When we disconnected the connection from the Zebra router to router A, the routes were automatically updated. MRLG reported the following:

Listing 21. Zebra reflecting OSPF routes after failover

```
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, > - selected route, * - FIB route

O>* 10.0.0.1/32 [110/59] via 192.168.2.2, eth0, 00:01:10
O>* 10.0.1.1/32 [110/11] via 192.168.1.2, eth1, 00:09:46
O   10.0.2.0/24 [110/10] is directly connected, dummy0, 00:10:24
C>* 10.0.2.0/24 is directly connected, dummy0
K * 127.0.0.0/8 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 192.168.0.0/30 [110/58] via 192.168.1.2, eth1, 00:01:10
O   192.168.1.0/30 [110/10[ is directly connected, eth1, 00:10:14
C>* 192.168.1.0/30 is directly connected, eth1
O   192.168.2.0/30 [110/10] is directly connected, eth0, 00:10:24
C>* 192.168.2.0/30 is directly connected, eth0
```

Due to the link-state nature of OSPF, failover was much faster: the new routes were propagated in under 30 seconds.

Summary

We started out with a simple need to find a substitute Cisco router for our equipment-strapped networking class. Zebra was an intriguing option, one that had received good recommendations from reviewers on the Internet. As our testing shows, Zebra makes an adequate replacement for Cisco routers in simple networking environments, and possibly for some more complex installations as well.

Admittedly, there was some degree of learning in going to Zebra. The use of separate routing daemons and separate configuration files was somewhat confusing at first, but after we had sorted that out, its general affinity to Cisco IOS immediately shone through.

Overall, Zebra makes dynamic routing in a Linux environment a much simpler task. If you're ever in a situation where you need to set up a router quickly on a limited budget, you should give Zebra a try.

Resources

- The GNU Zebra Web site is the ideal place to locate resources having to do with Zebra.
- The Multi-Router Looking Glass requires the Net::Telnet Perl package, available from CPAN.
- Iljitsch van Beijnum, author of the O'Reilly BGP book, gives an overview of Zebra in his article "Running Zebra on a Unix Machine".
- More details on RIP and OSPF can be found in this Cisco's UniverCD tutorial.
- If you need to learn the basics of building a Linux LAN, read "Setting up a Local Area Network" (*developerWorks*, February 2001).
- Going wireless? Read "Building a wireless access point on Linux" (*developerWorks*, July 2003).
- More on IBM's networking software can be found in our Networking listing.
- For information on all of IBM's networking hardware, visit the Networking Products page.
- Find more resources for Linux developers in the *developerWorks* Linux zone.

About the authors

Dominique Cimafranca is a Linux IT Specialist for IBM Philippines. He has implemented Linux in the whole range of hardware platforms from IBM for several enterprise customers. He has been writing about Linux and technology issues for the past three years. You can reach Dominique at dom-at-sketches.kom.ph.

Rex Young is network manager for C-Cube, a major call center in the Philippines. He has worked with large Cisco internetworking and VoIP implementations. He is a Certified Cisco Network Professional. You can reach Rex at youngrv-at-c-cubeservices.com.

PDF e-mail it!

Rate this article

This content was helpful to me:

Strongly disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

Comments?

Submit feedback