INR Lab Assignment
RIPng

C. Dumitru Jeroen van der Ham Mick Pouw Arno Bakker
(Arno.Bakker@os3.nl)

Feedback deadline:
December 1, 2017 10:00 CET

Abstract

This week we will initially be working with a simple network and then move on to a slightly more complex network. First we will get hands on experience with the routing daemons that are shipped with the Quagga package. Next, we will be looking at the network traffic between the various nodes to understand the behavior of RIPng.

Figure 1: RIPng network 1

Preparation

Task 1. Take a look at the documentation available at `http://www.nongnu.org/quagga/docs/quagga.html` and explain what `zebra` is.

A simple RIPng network

Task 2. In the `examples` folder of *pogo* you will find a config file for the simple RIPng network of Figure 1. Edit the `ripng.cfg` so the `pass_quaggadir` variables contain the absolute paths on the host of the **router**$_i$ directories. Start the network. The RIPng protocol should start automatically.

Task 3. Test the connectivity between **router1** and **router3** using the `traceroute6` utility.

Task 4. Examine the config files for both the network script and quagga and then on one of the routers connect to (via `vtysh`) and explore the `ripng` and `zebra` daemons. Very briefly explain what the following commands do: `show ?, show run, config terminal, router ?, network, exit, write memory` . Hint: some commands only exist in certain contexts.

Task 5. Connect to the RIPng process on **router2** and display the RIPng routes **and** protocol status.

Task 6. Start a `ping6` from **router1** to **router3**. Remove the global IPv6 address configured on **router2**'s `eth1`. What do you observe after max. 30 seconds? Why is the ping still working?

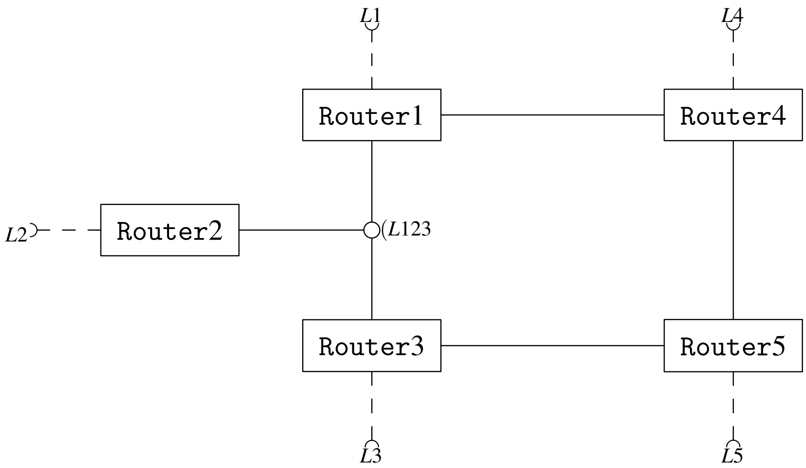

Figure 2: RIPng network 2

A more complex RIPng network

Task 7. Create the config file required to start the network depicted in Figure 2. Each **router**$_i$ has two or three network interfaces, one connected to a simple host via LAN L$_i$ (host not shown in the figure) and the rest connected to other routers (L$_{ij...}$). You will use the IPv6 blocks defined in the previous labs. The addressing is as follows:

- For L1 : The : y01 : `/64` block
- For L2 : The : y02 : `/64` block
- For L3 : The : y03 : `/64` block
- For L4 : The : y04 : `/64` block
- For L5 : The : y05 : `/64` block
- For L14 : The 8th `/64` block
- For L35 : The 16th `/64` block
- For L45 : The 32nd `/64` block
- For L123 : The 42nd `/64` block

As a convention, each router **router**$_i$ will use the ::i address on the network segment. (e.g., assuming that the IP block for L35 is `2001:0db8::/64`, **router5** will use `2001:0db8::5`). *Hint: you can arbitrarily number eth devices!*.

The end hosts do not need to be created, but you do need to create the network segments. Show the config file on your log.

Task 8. Draw the diagram of the network including the IP address of each router interface.

Task 9. For **router1** display the following:

- the routing table
- the RIPng table
- the RIPng status

Are there any differences between the RIPng table and the routing table? (*Hints: ip -6, vtysh, show*)

Task 10. Configure **router2** as a default gateway for all the other routers. Just adding static routes on each router is not allowed. Describe how you did this.

Task 11. Start a ping from **router3** to **router1**'s address on the L1 segment. Turn off the interface that connects **router1** to L123. How much time does it take for **router3** to reach **router1** again? Why? What can you say about the efficiency of RIPng in a network with unstable network paths?

Task 12. Using the packet dump for the L123 segment explain very briefly the startup and convergence of the RIPng protocol and the payload of the packets (look for interesting packets, when are they sent and why). Only mention the relevant packets. Upload the raw dump file (as generated by the sniffer) to your wiki and provide a link on your log.

Task 13. Configure **router1** such that it will accept updates only from **router2** and **router3**. Show the configuration changes. Warning: some solutions only work via `ripngd`'s telnet interface, not `vtysh`.

Optional

Bird is an alternative routing daemon. It is not installed on the UML image that you have built in the previous lab.

Task 14. Stop the `quagga` daemon on all the routers and configure `bird` to achieve similar results.