Lab
4

Dynamic Routing Protocols (RIP, OSPF and BGP)

What you will learn in this lab:

- How to configure the routing protocols RIP, OSPF, and BGP on a Linux PC and a Cisco router.
- How those routing protocols update the routing tables after a change in the network topology.
- How the count-to-infinity problem in RIP can be avoided.
- How OSPF achieves a hierarchical routing scheme through the use of multiple areas.
- How to set up and route traffic between autonomous systems with BGP and how to configure a routing policy.

Updated: October 2021

Table of Content

Study Material for Lab 4

1. **Distance Vector and Link State Routing Protocols**: The following article (in PDF) https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwj5ofvRl4TrAhUVlHIEHZ87BPQQFjALegQIAxAB&url=https%3A%2F%2Fptgmedia.pearsoncmg.com%2Fimages%2F9781587132063%2Fsamplechapter%2F1587132060_03.pdf&usg=AOvVaw2x7ZuazBLseG6JSLySw46_ provides an introduction to dynamic routing protocols. Review your knowledge of interdomain and intradomain routing, distance vector routing, and link state routing.
2. **Quagga**: Information on the *Quagga* routing software package is found at https://www.nongnu.org/quagga/docs/quagga.html
   Read the documents on *zebra*, *ripd*, *ospfd* and *bgpd*.
3. **RIP**: An overview of the Routing Information Protocol (RIP) is https://en.wikipedia.org/wiki/Routing_Information_Protocol
4. Commands to configure RIP on a Cisco router are explained at https://www.cisco.com/c/en/us/td/docs/ios/iproute_rip/command/reference/irr_book/irr_rip.html
5. **OSPF**: Here is an overview of Open Shortest Path First (OSPF) routing protocol: https://en.wikipedia.org/wiki/Open_Shortest_Path_First
6. Commands to configure OSPF on a Cisco router are explained at https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/configuration/xe-16/iro-xe-16-book/iro-cfg.html
7. **BGP**: This is an overview of the Border Gateway Protocol (BGP): https://en.wikipedia.org/wiki/Border_Gateway_Protocol
8. For an explanation of commands to configure BGP on a Cisco router go to https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bgp/configuration/xe-16/irg-xe-16-book/configuring-a-basic-bgp-network.html

Here are additional pointers to online course material on RIP, OSPF, and BGP:

- RIP: https://networklessons.com/rip
- OSPF: https://networklessons.com/ospf
- BGP: https://networklessons.com/bgp

Prelab 4

1. List one example each of a currently used distance vector routing protocol and a link state routing protocol for IPv4. For each protocol, list the full name, the acronym of the protocol, and version number (if applicable).
2. List one example each of a currently used distance vector routing protocol and a link state routing protocol for IPv6. For each protocol, list the full name, the acronym of the protocol, and version number (if applicable).
3. Provide a list of all routing protocols supported by the software package Quagga?
4. Which role does the *zebra* process play in Quagga?
5. Describe how a Linux user accesses the processes of Quagga (zebra, ripd, ospfd, bgpd) to configure routing parameters?
6. Explain what it means to "run RIP in passive mode".
7. Explain the meaning and the purpose of "triggered updates" in RIP.
8. What is the role of a designated router in OSPF? How is a designated router determined?
9. Provide the Cisco IOS and Quagga commands to display the link state database of OSPF.
10. If multiple routing protocols run simultaneously on a Cisco IOS router, how can you tell which routing table entries were created by the different routing protocols.

Lab 4 - Dynamic Routing Protocols

In the previous lab, you learned how to configure routing table entries manually. This was referred to as *static routing*. The topic of Lab 4 is *dynamic routing,* where *dynamic routing protocols* (from now on, called *routing protocols*) set the routing tables automatically without human intervention. Routers and hosts that run a routing protocol, exchange routing protocol messages related to network paths and node conditions, and use these messages to compute paths between routers and hosts.

Most routing protocols implement a shortest-path algorithm, which, for a given set of routers, determines the shortest paths between the routers. Some routing protocols allow that each network interface be assigned a *cost metric*. In this case, routing protocols compute paths with least cost. Based on the method used to compute the shortest or least-cost paths, one distinguishes *distance vector* and *link state* routing protocols. In a distance vector routing protocol, *neighboring* routers send the content of their routing tables to each other and update the routing tables based on the received routing tables. In a link state routing protocol, each router advertises the cost of each of its interfaces to *all* routers in the network. Thus, all routers have complete knowledge of the network topology, and can locally run a shortest-path (or least-cost) algorithm to determine their own routing tables.

The notion of an *autonomous system* (AS) is central to the understanding of routing protocols on the Internet. An autonomous system is a group of IP networks under the authority of a single administration, and the entire Internet is carved up into a large number of autonomous systems. Examples of autonomous systems are the campus network of a university and the backbone network of a global network service provider. Each autonomous system is assigned a globally unique identifier, called the *AS number*. On the Internet, dynamic routing within an autonomous system and between autonomous systems is handled by different types of routing protocols. A routing protocol that is concerned with routing within an autonomous system is called an *intradomain routing protocol* or *interior gateway protocol (IGP).* A routing protocol that determines routes between autonomous systems is called an *interdomain routing protocol* or *exterior gateway protocol (EGP).*

In this lab, you study two intradomain protocols, namely, the Routing Information Protocol (RIP) and the Open Shortest Path First (OSPF) protocol. Parts 1–4 of this lab deal with RIP, and Parts 5–6 are about OSPF. In Part 7, you are exposed to a few features of the Border Gateway Protocol (BGP), which is the interdomain routing protocol of the Internet.

For the network configurations, we start with the topology in Figure 4.1, to which we add additional PCs and routers in Part 2 (Figure 4.2) and Part 4 (Figure 4.3). For Part 7, we need a completely new topology (Figure 4.6).

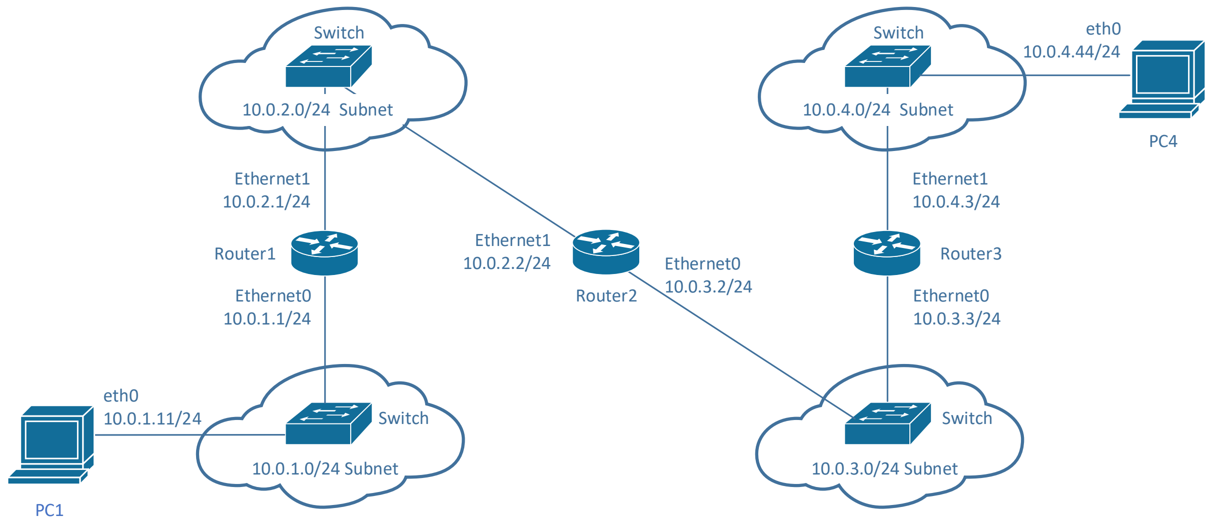


Figure 4.1. Network topology for Part 1.

Cisco Router	Interface Ethernet0	Interface Ethernet1
Router1	10.0.1.1/24	10.0.2.1/24
Router2	10.0.3.2/24	10.0.2.2/24
Router3	10.0.3.3/24	10.0.4.3/24

Table 4.1. IPv4 addresses of Cisco routers.

Linux PC	Interface *eth0*	Interface *eth1*	Default gateway
PC1	10.0.1.11/24	Disabled	10.0.1.1
PC4	10.0.4.44/24	Disabled	10.0.4.3

Table 4.2. IPv4 addresses of PCs.

Part 1. Configuring RIP on Cisco Routers

In this part of the lab, you configure the routing protocol RIP on the routers in Figure 4.1.

RIP is one of the oldest dynamic routing protocols on the Internet that is still in use. RIP is an intradomain routing protocol that uses a distance vector approach to determine the paths between routers. RIP minimizes the number of hops of each path, where the traversal of each subnet counts as a *hop*.

Two versions of RIP are in use today *RIPv2* (RIP Version 2) for IPv4 routers and *RIPng* (RIP next generation) for IPv6. In this lab, we only work with RIPv2. RIP is not useful for large networks since the convergence of routing tables after a change in the network is quite slow. On the other hand, RIP is very easy to configure.

Every RIP-enabled router periodically sends routing table entries to each of its neighboring routers in an update message. For each routing table entry, the router sends the destination (host IP address or network IP address) and the distance to that destination measured in hops. When a router receives an update message from a neighboring router, it updates its own routing table.

Exercise 1-a. Network setup

Step 1: Connect the Ethernet interfaces of the Linux PCs and the Cisco routers as shown in Figure 4.1.

Step 2: Configure the IP addresses and default gateways of *PC1* and *PC4* as shown in Table 4.2. Refer to Lab 3 for commands to configure a default gateway. Check the configuration by displaying the IP addresses (`ip addr`) and the routing table (`netstat -rn`).

Exercise 1-b. Configuring RIP on Cisco routers

Configure all the Cisco routers to run the routing protocol RIP. Once the configuration is completed, the Cisco routers and the PCs can issue *ping* commands to each other. Below, we give a brief overview of the basic commands used to configure RIP on a Cisco router.

IOS mode: privileged EXEC

`show ip protocols`

Displays parameters of the currently configured routing protocol.

`debug ip rip`

Enables a debugging mode where the router displays a message for each received RIP message.

`no debug ip rip`

Disables the debugging feature.

IOS mode: global configuration

`router rip`

Enables the routing protocol RIP and enters the router configuration mode with the following prompt:

```
Router1(config-router)#
```

You return from the router configuration mode to the global configuration mode by typing the command exit.

`no router rip`

Disables RIP.

<u>**IOS mode: RIP router configuration**</u>

`version 2`

Sets the RIP version to RIPv2.

`network` ***`10.0.2.0`***

Associates the subnet 10.0.2.0/24 with RIP. RIP sends updates for a network only if the network address has been associated with RIP. Note that the prefix length is not entered.

`no network` ***`10.0.2.0`***

Disables RIP for the specified network address.

`passive-interface` ***`Ethernet0`***

Sets interface *`Ethernet0`* in RIP passive mode. If an interface is in passive mode, the router processes incoming RIP messages, but does not transmit RIP messages on that interface.

`no passive-interface` ***`Ethernet0`***

Disables RIP passive mode on interface *`Ethernet0`*. This means that RIP messages are transmitted on this interface.

Step 1: Configure the Cisco Routers with IPv4 addresses as shown in Table 4.1 and enable the routing protocol RIP. The commands to set up *Router1* are as follows:

```
Router1# configure terminal
Router1(config)# no ip routing
Router1(config)# ip routing
Router1(config)# router rip
Router1(config-router)# version 2
Router1(config-router)# network 10.0.0.0
Router1(config)# interface Ethernet0
Router1(config-if)# ip address 10.0.1.1 255.255.255.0
Router1(config-if)# no shutdown
Router1(config)# interface Ethernet1
Router1(config-if)# ip address 10.0.2.1 255.255.255.0
Router1(config-if)# no shutdown
Router1(config-if)# end
Router1# clear ip route *
```

The command `'no ip routing'` is used to reset all previous configurations related to IP forwarding and routing protocols (RIP, OSPF, etc.). The command `'clear ip route *'` deletes all entries in the routing table (except the networks of the configured interfaces).

You can check the configured IP addresses and routing protocols with the commands

```
Router1# show ip interface
Router1# show ip interface brief
Router1# show ip protocols
```

Step 2: After the routers are configured, you can issue `ping` commands between any pair of routers or PCs. Try to ping *PC4* from *PC1* with the command

```
PC1$ ping 10.0.4.44
```

You can view the routing path by typing

```
PC1$ traceroute 10.0.4.44
```

Step 3: Check the routing table at the router by typing, e.g., on *Router1*,

```
Router1# show ip route
```

Every router should have one routing table entry for each subnet. Two entries are created by the configuration of IP addresses on the two interfaces. The other entries are created by RIP. Take a snapshot of the routing table of one of the routers.

Step 4: On *Router2*, the router in the middle, display the RIP updates with the command

```
Router2# debug ip rip
```

From the output you can infer how frequently RIP routers send distance vector information. Record how frequently the following events occur:

- *Router2* transmits a RIP update on the same interface;
- *Router2* receives a RIP update from *Router1*;
- *Router2* receives a RIP update from *Router3*.

Take a snapshot of the output of the debug ip rip command which contains two of each of the update events.

Step 5: Once you can successfully *ping* the IP addresses of all routers, proceed to Part 2.

Lab Question/Report

1. Include the snapshot of the routing table taken in Step 3. Indicate which routing table entries were created by RIP.

2. Answer the questions on the frequency of RIP updates from Step 4. Use the snapshot taken in Step 4 to support your answer.

Part 2. Configuring RIP on a Linux PC

In this part of the lab, you configure RIP on Linux PCs. The topology for this part, shown in Figure 4.2, adds *PC3* to the network topology. You will set up *PC3* as a router that runs the RIP protocol. You will also learn that running RIP on hosts (*PC1* and *PC4*), can replace the default route.

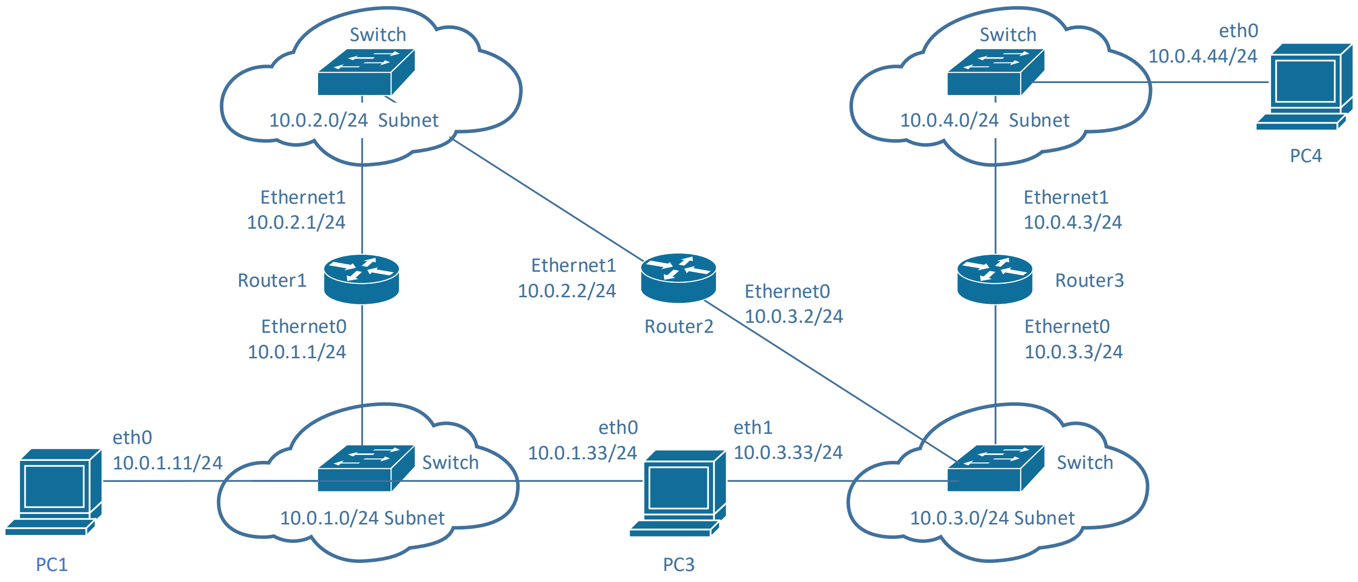


Figure 4.2. Network topology for Part 2.

The configuration of routing protocols on Linux PCs is done with the routing software package *Quagga*. Before starting the exercise, read the tutorial on the Quagga software package in the appendix. The tutorial focuses on the features used in the lab exercises and omits many features that are available in *Quagga*.

Below is a list of the most relevant commands

<u>Zebra Process</u>
The zebra process must be running if you want to configure routing protocols with Quagga.

```
sudo service zebra start
```
Starts the zebra process. The zebra process must be running when you use Quagga.

```
sudo service zebra status
```
Shows the status of the zebra process.

```
sudo service zebra stop
```
Terminates the zebra process.

```
sudo service zebra restart
```

Stops the zebra process, if it is running, and then starts the process.

Ripd Process

The `ripd` process runs the RIP routing protocol. The commands to control the process are analogous to those of the `zebra` process.

```
sudo service ripd start
sudo service ripd status
sudo service ripd stop
sudo service ripd restart
```

```
telnet localhost 2602
```

Access the command line interface of the `ripd` process. This requires a password (The default password is '`zebra`'. The password is found in `/etc/quagga/ripd.conf`). The command line interface shows the prompt

ripd>

Once the Telnet session is established you can type configuration commands. The commands and command modes are similar to those of a Cisco router. The commands are discussed in the exercises.

Typing

```
ripd> exit
```

closes the Telnet session.

```
show ip rip
```

Displays the RIP routing table, referred to as RIP table. These are the routes that are learned through RIP.

```
show ip rip status
```

Displays the status of the RIP configuration.

Exercise 2-a. Configuring a Linux PC as a RIP router

Here, you configure *PC3* as a RIP router, using the *Quagga* software package. The steps to configure RIP on *PC3* are similar to the configuration of RIP on a Cisco router.

Step 1: The network topology is shown in Figure 4.2. Attach the interfaces of *PC3* as shown in the Figure 4.2. The *eth0* interface of *PC3* connects to the switch, which is connected to *PC1* and *Router1*. The *eth1* of *PC3* interface is connected to the switch, where *Router2* and *Router3* connect to.

Step 2: The configuration of *PC1, PC4, Router1, Router2*, and *Router3* is as in Part 1. In particular, RIP is running on the Cisco routers.

Step 3: Configure the IP addresses of *PC3* for interfaces eth0 and eth1 as shown in Table 4.2.

Linux PC	Interface *eth0*	Interface *eth1*	Default Gateway
PC3	10.0.1.33/24	10.0.3.33/24	–

Table 4.2. IPv4 addresses of PCs.

Step 4: On *PC4*, issue a `traceroute` to *PC1*

```
PC4$ traceroute 10.0.1.11
```

and take a snapshot. Once *PC3* is configured as a RIP router, we expect that the route from *PC4* to *PC1* changes.

Step 5: Enable IPv4 forwarding and disable reverse path filtering on *PC3* by typing

```
PC3$ sudo sysctl -w net.ipv4.ip_forward=1
PC3$ sudo sysctl -w net.ipv4.conf.all.rp_filter=0
```

Step 6: Before configuring RIP on *PC3*, start two *Wireshark* sessions to capture the traffic on **both** interfaces of *PC3*.

Multiple Wireshark sessions on a PC (Ubuntu desktop only)

From the Ubuntu desktop, you can start the *Wireshark* application by selecting the *Wireshark* icon from the *Activities* menu. However, you can only start one instant of Wireshark in this fashion. If you need more than one Wireshark application (as in Step 6), open a terminal window and issue the command
`sudo wireshark &`

Step 7: Restart the `zebra` and `ripd` processes on *PC3* with

```
PC3$ sudo service zebra restart
PC3$ sudo service ripd restart
```

Establish a *Telnet* session to the *ripd* process using

```
PC3$ telnet localhost 2602
```

As password, use the default password `'zebra'`. If this does not work, lookup the password in the RIP configuration file by typing

```
PC3$ sudo more /etc/quagga/ripd.conf
```

Once the session is established, issue the following set of commands, which enable RIP on both interfaces of *PC3*.

```
ripd> enable
ripd# configure terminal
ripd(config)# router rip
ripd(config-router)# version 2
ripd(config-router)# network 10.0.0.0/8
ripd(config-router)# no passive-interface eth0
ripd(config-router)# no passive-interface eth1
ripd(config-router)# redistribute connected
ripd(config-router)# end
ripd# show ip rip status
```

The commands as well as command prompts are similar to what you have seen in Cisco IOS. Here are a few comments on commands that differ from Cisco commands:

- `network 10.0.0.0/8`: Associates the subnet 10.0.0.0/8 with RIP. Different from the corresponding command in Cisco IOS, the prefix length is required.
- `no passive-interface eth0`: Sets *eth0* in active mode for the RIP protocol. RIP messages are sent out only on interfaces in active mode.
- `redistribute connected`: Ensures that the *ripd* process sends updates on its directly connected interfaces. This command does not exist in Cisco IOS.
- `Show ip rip status:` Displays the status of the RIP configuration.
- `show ip rip`: Displays the routing table of RIP

Step 8: On *PC4*, repeat issuing `traceroute` commands to *PC1* as in Step 5. Since RIP performs shortest path routing, you will eventually see that the route from *PC1* to *PC4* switches to include *PC3*.

Take a screen snapshot of the traceroute output that shows the new route.

Also, perform traceroute from *PC1* to *PC4*. Compare the output with that of the traceroute from *PC4* to *PC1*.

Step 9: Go back to *PC3*. In the *Telnet* session to the *ripd* process, observe the RIP routing table by typing the command

```
ripd# show ip rip
```

and take a snapshot. Then terminate the Telnet session with the commands

```
ripd# exit
```

Display the IPv4 routing table of *PC3* with the command

```
PC3$ netstat -nr
```

and take a snapshot.

Observe the commonalities and differences in the RIP routing table and the IPv4 routing table.

Leave the *Wireshark* instances running on the two interfaces of *PC3*.

Lab Question/Report

1. Include the screen captures from Steps 4 and 8, and provide a brief explanation of the change of the output.
2. Include the screen capture from Step 9, and discuss commonalities and differences of the tables.

Exercise 2-b. Exploring RIP messages

Here, you will look at the RIP messages captured by Wireshark in the previous exercise.

Step 1: In the Wireshark applications started in the previous exercise, set a display filter to "rip" so that only RIPv2 messages are displayed.

Step 2: Explore the messages and answer the following questions:

a) What is the destination IP address of RIPv2 messages? What is the significance of this address? Who are the receivers of these messages?
b) How are RIP messages encapsulated? What are the port numbers?
c) Most RIPv2 messages that are captured are "RIP Response" messages, but you should see at least one RIP Request message. Who sent this message and when was it sent?
d) How frequently does *PC3* send a RIP Response message?
e) Explore the network prefixes and the metric that are contained in the RIPv2 messages sent and received by *PC3,* after *PC3* is configured as a RIPv2 router. Consider the packets captured on both interfaces of *PC3*. Relate the network prefixes and metric values to the routing RIP routing table from the previous exercise.

Step 3: If you have not recorded the answers to the questions in the previous step, save the captured RIP messages. Terminate the Wireshark traffic captures.

Lab Question/Report

1. Provide answers to the questions in Step 2.

Exercise 2-c. Configuring RIP on a Linux host

In the previous exercise, *PC1* and *PC4* are configured as hosts, that is, IP forwarding is not enabled. Hosts generally do not execute a routing protocol, and, instead, base their routing on a default route. As an alternative to a default route, we can configure Linux hosts so that they listen to RIPv2 messages, but do not transmit them. This is achieved by running the RIP protocol in *passive mode*.

In the following, you configure RIP on *PC1*. Note that this will improve the routing configuration of *PC1*. With the configured default route, traffic from *PC1* to *PC4* was first sent to *Router1*, which was followed by a route redirect. This is avoided when we run RIP on *PC1*.

Since the *PC1* is running as a host, the RIP configuration on *PC1* is set to passive mode, where the PC receives and processes incoming RIP messages, but does not transmit them.

Step 1: Continue with the configuration from the previous exercise. On *PC1*, display the IPv4 routing table with

```
PC1$ netstat -rn
```

and take screen capture of the output.

Step 2: On *PC1*, run a traceroute to *PC4* and to *Router2* with

```
PC1$ traceroute 10.0.4.44
PC1$ traceroute 10.0.2.2
```

Again, take a screen capture of the output.

Step 3: On *PC1*, remove the default route by typing

```
PC1$ sudo ip route flush default
```

You may want to check that the default route in the routing table is removed (`netstat -rn`)

Start *quagga* and the *ripd* by typing

```
PC1$ sudo service zebra restart
PC1$ sudo service ripd restart
```

Connect to the `ripd` process via `Telnet`.

```
PC1$ telnet localhost 2602
```

The system will prompt you for a password. Enter the default password '`zebra`', or the password is in `/etc/quagga/rpid.conf`.

Issue the following commands to enable RIP in passive mode on interface *eth0*:

```
ripd> enable
ripd# configure terminal
ripd(config)# router rip
ripd(config-router)# version 2
ripd(config-router)# network 10.0.0.0/8
ripd(config-router)# passive-interface eth0
ripd(config-router)# end
ripd# show ip rip
```

Here, the RIP configuration sets interface eth0 into passive mode, meaning that *PC1* listens to RIPv2 messages, but does not transmit them.

The '`show ip rip`' command displays the routing database of the RIP protocol. Pay attention to the Metric values in the table and match them with the length of the routes. Repeat the '`show ip` `rip'` command a few times until you see an entry for each subnet in Figure 4.2. Then, exit the *Telnet* session with the command.

```
ripd# exit
```

Step 4: Continue on *PC1*. View the *IPv4* routing table with the command

```
PC1$ netstat -rn
```

and take a screen snapshot of the output.

- Compare the output with the IPv4 routing table from Step 1.

Step 5: Perform a traceroute from *PC1* to *PC4*, and *Router2*

```
PC1$ traceroute 10.0.4.44
PC1$ traceroute 10.0.2.2
```

and take a screen snapshot of the output.

Step 6: On *PC1*, terminate the *zebra* and *ripd* processes and add the old default route from Table 2.2, by typing

```
PC1$ sudo service zebra stop
PC1$ sudo service ripd stop
PC1$ sudo ip route add default via 10.0.1.1
```

Step 7: Leave the configuration in place for the next part of the lab.

Lab Questions/Report

1. Include the screen snapshot of the IPv4 routing tables from Step 1 and Step 4, and compare.
2. Include the screenshots of the traceroutes from Step 2 and Step 5, and compare.

Part 3. Convergence of RIP

This part continues with the configuration of Part 2. You will explore how RIP detects changes to the network topology, and how long it takes until RIP updates the routing tables.

Exercise 3-a. Convergence of RIP after a link failure

Here, you disconnect the Ethernet connection of interface *eth0* on *PC3* and observe how much time RIP takes to update the routing tables to reflect the new topology.

Step 1: On *PC4*, run a traceroute to *PC1* (10.0.1.11). Verify that the route goes through *PC3*. If this is not the case, retrace the configuration of PCs and Cisco routers from the previous parts.

Step 2: Issue a `ping` command from *PC4* to *PC1*. Do not terminate the `ping` command until this exercise is completed in Step 4.

```
PC4$ ping 10.0.1.11
```

Step 3: On *PC3*, disable the *eth0* interface with the command

```
PC3$ sudo ip link set dev eth0 down; date
```

The `date` command displays the time you disconnect. Disconnect the Ethernet connection between the *eth0* interface on *PC3* to the switch. Note the time when you disconnect.

Now, all routes that pass through *PC3* will fail. The output of *ping* on *PC4* should show that the destination network is unreachable. RIP must find a new route between *PC4* and *PC1*.

Step 4: Wait until the `ping` command on *PC4* is successful again, that is, *ICMP Echo Reply* messages are returned to *PC4*. This occurs once an alternate path has been found between *PC4* and *PC1*, and the routing tables have been updated accordingly. This may take several minutes.

Record the time it takes until the *ping* is successful again (using the `date` command on one of the PCs).

Step 5: Stop the `ping` command on *PC4*. To show the new route between *PC4* and *PC1* run a traceroute from *PC4* to IP address 10.0.1.11.

Step 6: On *PC3*, re-enable the *eth0* interface on *PC3* with the command

```
PC3$ sudo ip link set dev eth0 up
```

Check whether disabling the interface in Step 3 has deleted the IP address (with ip addr show eth0). If the eth0 interface of *PC3* has no IPv4 address, reset the address to the original value given in Table 4.2.

Step 7: Leave the configuration in place for the next part of the lab.

Lab Questions/Report

1. Include your answer on the convergence time from Step 3.

Part 4. Count-to-infinity in RIP

Distance vector routing protocols, such as RIP, are susceptible to a convergence problem which is known as *count-to-infinity*. The problem is a consequence of the fact that distance vector routing protocols exchange routing information only with their neighbors. Here, it may happen that, after the failure of a link, information about routes that use the failed link are propagated a long time after the failure has occurred. This results in a slow convergence of the routing tables. Each time the routers exchange RIP messages, the cost of a path that uses the failed link increases, but it takes a long time until all routers realize that routes through the failed link are unavailable.

The goal of this part of the lab is to observe count-to-infinity. RIP has a number of protocol features that try to prevent count-to-infinity from occurring. To make count-to-infinity visible, these features will be disabled. Still, since count-to-infinity requires that routing updates occur in a certain order, it may take a large number of tries to observe it.

The network configuration is shown in Figure 4.3. Compared to the network in Part 3, *Router4* is added. After the routing tables have converged, you disable the *Ethernet1* interface on *Router4*, which may trigger the occurrence of the count-to-infinity problem.

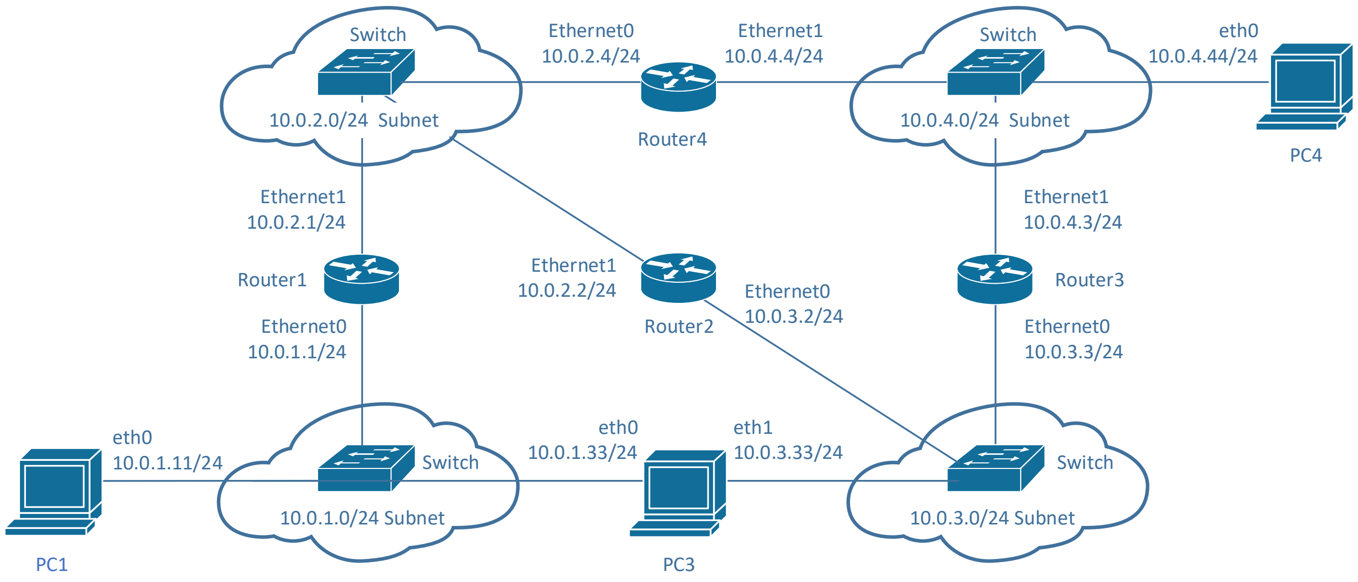


Figure 4.3. Network topology for Part 4.

The IP configuration of *Router4* is shown in Table 4.3. The configuration of the other devices is unchanged from the previous parts.

Cisco Router	Ethernet Interface *Ethernet0*	Ethernet Interface *Ethernet1*
Router4	10.0.2.4/24	10.0.4.4/24

Table 4.3. IP configuraiton of Router 4.

Exercise 4-a. Adding Router4 to the network configuration

Step 1: *PC1, PC3, PC4, Router1, Router2*, and *Router3* are configured as in Part 3. Verify that all Cisco routers and PCs can send messages to each other.
Also verify that the *eth0* interface on *PC3* has been re-enabled (Step 6 in *Exercise 3-b*).

Step 2: Configure *Router4* with the IPv4 addresses in Table 4.3, enable IPv4 forwarding, and set up RIP following with

```
Router4# configure terminal
Router4(config)# no ip routing
Router4(config)# ip routing
Router4(config)# router rip
Router4(config-router)# version 2
Router4(config-router)# network 10.0.0.0
Router4(config)# interface Ethernet0
Router4(config-if)# ip address 10.0.2.4 255.255.255.0
Router4(config-if)# no shutdown
Router4(config)# interface Ethernet1
Router4(config-if)# ip address 10.0.4.4 255.255.255.0
Router4(config-if)# no shutdown
Router4(config-if)# end
Router4# clear ip route *
```

Step 3: Check the routing table on *Router4* with the command

```
Router4# show ip route
```

Wait until you see a routing table entry for each subnet.

Step 4: Test the routing setup with a traceroute from *PC4* to IP address 10.0.1.11. On *PC4*, type

```
PC4$ traceroute 10.0.1.11
```

The output should include *Router4*, more specifically, it should show IP address 10.0.4.4.

Exercise 4-b. The count-to-infinity problem

The goal of this exercise is to observe the effects of the count-to-infinity problem for routes to network 10.0.1.0/24. Note in Figure 4.3, that traffic to network 10.0.1.0/24 passes either through *PC3* or *Router1*. First, you ensure that traffic to network 10.0.1.0/24 always prefers the path through *PC3*. This is done by setting the link metric of *Router1*'s interface on network 10.0.1.0/24 to a large value. Then you disable the *eth0* interface of *PC3*. When the interface is disabled, the next update packet from *PC3* will list the cost metric for network 10.0.1.0/24 as 16, which, in RIP, is interpreted as *infinity*. If this information is not distributed quickly, then the other routers in the network send RIP messages, which assume that *PC3* is still connected to network 10.0.1.0/24. For example, *Router3* may still believe that it can reach network 10.0.1.0/24 in two hops. When such an update arrives at *Router4*, then *Router4* wrongly assumes that it can reach network 10.0.1.0/24 via *Router3* in three hops. In this situation, it may take the routers a very long time, before they realize that the best path to network 10.0.1.0/24 is through *Router1*. This slow convergence of the routing tables is called the *count-to-infinity problem*.

RIP has several protocol features that try to avoid the count-to-infinity problem. One of them, called *triggered update*, forces a router to immediately issue a RIP response packet, whenever the cost metric of a routing table entry changes. In the above scenario, the link failure triggers an update message, which ensures that information about the link failure of *PC3* is quickly propagated to all systems in the network. This prevents routers from continuing to advertise routes that are based on incorrect information about the network topology. Another feature that avoids count-to-infinity is the *hold-down* mechanisms. When a router receives information that a route is unavailable (i.e., has cost metric 16), it puts the route in a hold-down state. In this state, the router ignores routing updates that advertise a better cost metric for a certain period of time (which is given by the hold-down timer).

To increase the likelihood of the count-to-infinity problem occurring, you will disable triggered update and set the hold-down timer to zero. Still, it may happen that the count-to-infinity problem does not manifest itself, since the problem is dependent on the sequence of RIP messages that routers send to each other. Repeating the exercise several times will eventually exhibit the count-to-infinity problem.

The exercises in this part uses additional configuration commands for RIP.

IOS mode: RIP router configuration

Offset commands

In RIP, by default each router increases the cost of a path by one. The offset command can be used to artificially increase the cost by a given value.

`offset-list` *`0`* `in` *`10 Ethernet0`*

Increases the metric (hop count) of incoming RIP messages that arrive on interface *Ethernet0* by value *10*.

`offset-list` *`0`* `out` *`10 Ethernet0`*

Increases the metric of outgoing RIP messages that are sent on interface *Ethernet0* by value *10*.

`no offset-list` *`0`* `in` *`10 Ethernet0`*

Disables the specified offset-list command for incoming RIP messages for interface *Ethernet0*.

`no offset-list` *`0`* `out` *`10 Ethernet`*

Disables the specified offset-list command for outgoing RIP messages.

Hold-down timer

The timers command can be used to manipulate four timers of the RIP protocol.

`timers basic <update> <invalid> <hold-down> <flush>`

Sets the values of the timers in the RIP protocol. The timers are measured in seconds:

<update>	–	The time interval between transmissions of RIP update messages (Default: 30 sec).
<invalid>	–	The time interval after which a route, which has not been updated, is declared invalid (Default: 180 sec).
<hold-down>	–	Determines how long after a route has been updated as unavailable, a router will wait before accepting a new route with a lower metric. This introduces a delay for processing incoming RIP messages with routing updates after a link failure (Default: 180 sec). Setting the hold-down timer to 0 means that the RIP process immediately accepts updates to a route.
<flush>	–	The amount of time that must pass before a route that has not been updated is removed from the routing table (Default: 240 sec).

Triggered update

In RIP, a triggered update means that a router sends a RIP message with a routing update, whenever one of its routing table entries changes. The triggered update feature is controlled by setting the value of the flash-update-threshold timer. Triggered updates are disabled by setting the flash-update-threshold timer to the same value as the update timer. Assuming that the update timer is set to the default value of 30 seconds, the command to disable triggered updates is

`flash-update-threshold <time>`

Whenever the metric of a routing table changes, the router sends a RIP Response packet only if the next regularly scheduled Response messages is more than <time> seconds away. Hence, if *<time>* is set to the same value as the update timer, then triggered updates are disabled.

Step 1: On ***all*** Cisco routers, set the *hold-down timer* to zero and disable *triggered updates*. Below is the configuration for *Router1*.

```
Router1# configure terminal
Router1(config)# ip routing
Router1(config)# router rip
Router1(config-router)# version 2
Router1(config-router)# network 10.0.0.0
Router1(config-router)# timers basic 30 180 0 240
Router1(config-router)# flash-update-threshold 30
Router1(config-if)# end
Router1# clear ip route *
Router1# show ip protocols
```

The last command shows the parameters setting of RIP. Check the parameters to make sure the entries are correct.

Step 2: Next, you make routes to and from network 10.0.1.0/24 through *Router1* unattractive by increasing the cost metric of interfaces *Ethernet0/0* and *Ethernet0/1* at *Router1* to 10. This simulates a situation where the hop count from *Router1* to network 10.0.1.0/24 is 10 hops.

```
Router1# configure terminal
Router1(config)# router rip
Router1(config-router)# offset-list 0 out 10 Ethernet0
Router1(config-router)# offset-list 0 out 10 Ethernet1
Router1(config- router)# end
```

Step 3: Wait until the routing tables on all routers have converged. Since the cost of the interface on *Router1* is set high, you should observe that the traffic from all hosts and routers to network 10.0.1.0/24 passes through *PC3*. This can be verified by issuing a `traceroute from` *PC4* to IP address 10.0.1.11.

Step 4: Start Wireshark to capture traffic on the *eth1* interface of *PC3*. Set a display filter to `‘rip’` to display only RIP messages.

Step 5: Issue a `ping` command from *PC4* to *PC1*, and leave the *ping* running.

```
PC4$ ping 10.0.1.11
```

Step 6: On *Router3*, enable the debugging mode of RIP. In this mode, the router displays all received RIP messages. The mode is enabled by typing

```
Router3# debug ip rip
```

If you want to leave the debugging mode, type `` `no debug ip rip’``.

Step 7: On *PC3*, disable interface *eth0* to break the connection between *PC3* and network 10.0.1.0. *PC3* will send out a RIP message indicating that the cost metric to network 10.0.1.0/24 is *infinity* (numeric 16).

```
PC3$ sudo ip link set eth0 down
```

When the interface is disabled, `ping commands` from *PC4* to *PC1* will fail. The `ping` commands to *PC1* are again successful, once the routing tables have converged.

Step 8: Now you may be able to observe the slow convergence of the routing tables due to the count-to-infinity problem, by observing the content of the RIP messages that are shown by *Wireshark* on *PC3*, and by observing the debugging output on *Router3*.

You can also see the arrival of RIP message in the console of *Router3*, where RIP debugging is turned on.

Look for RIP messages sent by *Router2* or *Router3*, which advertise the network 10.0.1.0/24. Before the interface at *PC3* is disabled, the value of Metric to network 10.0.1.0/24 that is advertised by *PC3* value 1. After the interface at *PC3* was disabled, the shortest path to network 10.0.1.0/24 has a metric 12 (since *Router1* adds 10 units to the metric). If you observe that a metric less than 12 is advertised (e.g., 4, 7, 10), you know that a count-to-infinity is in progress, since a path to network 10.0.1.0/24 with this length does not exist.

Count-to-infinity is not reliably observed
When Wireshark shows that the path to network 10.0.1.0/24 has metric 12 (this is the path across *Router1*), RIP has converged. If convergence occurs quickly, that is, right after you disable the *eth0* interface on *PC3*, then count-to-infinity has not manifested itself. In this case repeat the experiment, by re-enabling *eth0* on *PC3*

```
PC3$ sudo ip link set dev eth0 up
```

then waiting for a few seconds and, finally, disabling the interface again with

```
PC3$ sudo ip link set dev eth0 down
```

You may have to repeat this several times, since count-to-infinity depends on events at different routers occurring in a certain sequence. Eventually, you will observe count-to-infinity in the above configuration. Feel free to give up after 10 attempts.

Step 9: Save the debug messages on *Router3*. Then disable RIP debugging on *Router3*:

```
Router3# no debug ip rip
```

Step 10: On *PC3*, stop Wireshark and save the RIP messages that were captured by *Wireshark*. You only need to save the packets needed to explain the count-to-infinity problem.
Also, stop the *ping* command on *PC4*.

Lab Questions/Report

1. Use the saved RIP messages from *Router3* and *PC3* to describe the count-to-infinity problem in the above exercise. Include relevant fields from the saved RIP messages to illustrate your description.

Part 5. Configuring Open Shortest Path First (OSPF)

Next, you explore the routing protocol Open Shortest Path First (OSPF). OSPF is a link state routing protocol, where each router sends information on the cost metric of its network interfaces to all other routers in the network. The information about the interfaces is sent using Link State Advertisements (LSAs). LSAs are disseminated using flooding, that is, a router sends its LSAs to all its neighbors, which, in turn, forward the LSAs to their neighbors, and so on. Each router maintains a link state database (LSDB) of all received LSAs, which provides the router with complete information about the topology of the network. A router uses its LSDB to run a shortest path algorithm that computes the best paths in the network.

Unlike distance vector routing protocols, link state routing protocols do not have convergence problems, such as count-to-infinity. This is seen as a significant advantage of link state protocols over distance vector protocols.

OSPF is the most widely used link state routing protocol on the Internet. The functionality of OSPF is rich, and the lab exercises highlight only a small portion of the OSPF protocol. The Internet Lab uses OSPF version 2 (OSPFv2). OSPF version 3 (version 3) is a more recent version, which is mostly used for routing of IPv6 traffic.

This part of the lab uses the network topology from Part 4 (Figure 4.3). The IPv4 addresses of the interfaces are the same as before. They are summarized in Table 4.4. In this part, the Cisco routers and *PC3* are set up as OSPF routers. At a later time, an additional network is configured.

Point-to-point links vs. stub networks vs. transit networks

OSPF distinguishes between point-to-point links between routers and routers that are connected by a multi-access network (e.g., Ethernet). For multi-access networks, OSPF classifies those with a single router as *stub networks* and all others as *transit networks*. (This may be counterintuitive: OSPF labels all point-to-point links as stub networks.)

Avoiding a combinatorial problem
Consider the subnet 10.0.2.0/24 in Figure 4.3, which is a transit network with three routers. If OSPF advertises the links in this subnet there should be a total of 3 LSAs, one link for each pair of routers. However, this does not scale. For a transit network with *N* routers, OSPF would require *N(N-1)/2* LSAs. To prevent this, for each transit network, OSPF elects one router as the Designated Router (DR), e.g., the router with the largest router ID. This router sends out a single *Network LSA* for the multiaccess network, which contains the list of all routers connected to this network. In addition, each router sends out a *Router LSA.* So, with *N* routers on a subnet there are at most *N+1* LSAs.

Cisco Router	Interface Ethernet0	Interface Ethernet1
Router1	10.0.1.1/24	10.0.2.1/24
Router2	10.0.3.2/24	10.0.2.2/24
Router3	10.0.3.3/24	10.0.4.3/24
Router4	10.0.2.4/24	10.0.4.4/24

Table 4.4. IPv4 addresses of Cisco routers.

Linux PC	Interface *eth0*	Interface *eth1*	Default gateway
PC1	10.0.1.11/24	Disabled	10.0.1.1
PC3	10.0.1.33/24	10.0.3.33/24	–
PC4	10.0.4.44/24	Disabled	10.0.4.3

Table 4.5. IPv4 addresses of PCs.

Exercise 5-a. Network setup

The following instructions consider that you may have skipped Part 4. However, it is assumed that you completed the configuration in Parts 1–3.

Step 1: If you continue from Part 4, skip to Step 4.

Step 2: Connect the Ethernet interfaces of *Router4* as shown in Figure 4.3.

Step 3: Configure *Router4* with the following commands

```
Router4# configure terminal
Router4(config)# no ip routing
Router4(config)# ip routing
Router4(config)# interface Ethernet0
Router4(config-if)# ip address 10.0.2.4 255.255.255.0
Router4(config-if)# no shutdown
Router4(config)# interface Ethernet1
Router4(config-if)# ip address 10.0.4.4 255.255.255.0
Router4(config-if)# no shutdown
Router4(config-if)# end
Router4# clear ip route *
```

Step 4: You may want to verify that the IPv4 configuration of the Cisco Routers and PCs is as shown in Tables 4.4 and 4.5.

Step 5: On *PC3*, disable RIP routing by stopping the *ripd* process with the command

```
PC3$ sudo service ripd stop
```

Once the *ripd* process is stopped, the entries in the IPv4 routing table that are configured by RIP will disappear.

Also, double check that the *eth0* interface of *PC3* is up and has IPv4 address. To make sure you may want to run the commands

```
PC3$ sudo ip link set eth0 up
PC3$ ip addr show eth0
```

If necessary, configure the IPv4 address as given in Table 4.2.

Exercise 5-b. Configuring OSPF on Cisco routers

Here, you configure OSPF on the Cisco routers. Below we give a brief description of the basic IOS commands used to configure OSPF on a Cisco router. As usual, each command must be issued in a particular IOS command mode.

IOS mode: Global configuration

`router ospf <process-id>`

Enables an OSPF routing process. Each router can execute multiple OSPF processes. The `<process-id>` is a number that identifies the process. In this lab, only one OSPF process is started per router, where the <process-id> value is always set to one. The command enters the router configuration mode, which has the following command prompt:

```
Router1(config-router)#
```

`no router ospf <process-id>`

Disables the specified OSPF process.

IOS mode: privileged EXEC

`show ip ospf`

Displays general information about the OSPF configuration.

`show ip route`

Displays the routing table

`show ip route ospf`

Displays the routing table entries related to OSPF.

`show ip protocols`

Displays IP protocols running and active in the router. If OSPF is running, it displays details information about the process. This is a good way to determine if the commands that you have entered are correct and being effective or not.

`show ip ospf database`

Displays a summary of entries in the LSDB listing the router and network link states.

`show ip ospf database network`

Displays detailed information of the network link states in the LSDB. For each network link state, all OSPF routers on this network are listed.

`show ip ospf database router`

Displays detailed information of the router link states in the LSDB. For each router link state, both connected transit and stub networks are shown.

<u>IOS mode: Router configuration</u>

`network` ***`<Netaddr> <InvNetmask>`*** **`area`** ***`<AreaID>`***

Associates a network prefix with OSPF and associates an OSPF area to the network address. The prefix is specified with an IP address *`<Netaddr>`* and an inverse netmask *`<InvNetmask>`*. For example, `Netaddr=10.0.0.0` and `InvNetmask=0.255.255.255` specify the network prefix 10.0.0.0/8. The `AreaID` is a number that associates an area with the address range. Area 0 is reserved to specify the backbone area.

Example: To run OSPF on *Router1* for the address range 10.0.0.0/8 and assign it to Area 1, type

```
Router1(config-router)# network 10.0.0.0 0.255.255.255 area 1
```

`no network` ***`<Netaddr> <InvNetmask>`*** **`area`** ***`<AreaID>`***

Disables OSPF for the specified network area.

`passive-interface` ***`<Iface>`***

Sets interface *`<Iface>`* into passive mode. In passive mode, the router only receives and processes OSPF messages, but does not transmit OSPF messages.

`no passive-interface` ***`<Iface>`***

Sets interface *`<Iface>`* into active mode. In active mode, the router receives and transmits OSPF messages.

`router-id` ***`<IPaddress>`***

Assigns the IP address *<IPaddress>* as the router identifier (router-id) of the local OSPF router. In OSPF, the router-id is used in LSA messages to identify a router. In IOS, by default, a router selects the highest IP address as the router-id. The above command can be used to set the value explicitly

Step 1: Configure *Router1* to run OSPF by typing

```
Router1# configure terminal
Router1(config)# no ip routing
Router1(config)# ip routing
Router1(config)# router ospf 1
Router1(config-router)# network 10.0.0.0 0.255.255.255 area 1
Router1(config-router)# end
```

The above commands (1) reset IP forwarding by disabling and then enabling it, (2) enable the OSPF routing protocol, and (3) link subnet 10.0.0.0/8 with OSPF Area 1. Since no *router-id* is specified, the highest IP address of *Router1*, 10.0.2.1, is used as the *router-id*. In (1), disabling IP forwarding also terminates the RIP routing process that was configured in earlier parts of the lab.

Display the status of the OSPF configuration with the commands:

```
Router1# show ip protocols
Router1# show ip ospf
```

Step 2: Next, configure the other Cisco routers. The commands for configuring OSPF are the same as in Step 2. Each time you set up a new OSPF router, the LDBS at each router will grow. To follow the evolution of the LSDB, display the LSDB entries at *Router1* each time **before** you configure one of the other routers with the command

```
Router1# show ip ospf database
```

The command gives an overview of the entries in the LSDB, but does not show the details of the entries. Observe that there are two types of link states: router link states and network link states.

Observe that the link state database at *Router1* grows, as you add OSPF routers.

Take a screenshot of the output after all Cisco routers have been configured as OSPF routers.

Step 3: When all four Cisco routers have been configured, display the details of the LSDB at *Router1* by typing

```
Router1# show ip ospf database
Router1# show ip ospf database network
Router1# show ip ospf database router
```

The second command displays details of network link states and the third command displays details of router link states.

Save the output. (Taking screen snapshots is an alternative, but the output is quite long.)

Step 4: Display the routing table entries at *Router1* that were created by OSPF with

```
Router1# show ip route ospf
```

and take a screen snapshot.

Step 5: In a network with OSPF routing, all routers have the same link state database. Confirm this by performing Step 4 on one of the other Cisco routers and comparing the output. There is no need to save the output.

Lab Questions/Report

1. Provide the saved LSDB summary from Step 2 and the screen snapshot from Step 4.

2. Show how the information in the LSDB of *Router1* (from Step 3) can be used to construct parts of the network topology from Figure 4.3, including the Cisco routers, networks that connect two or more Cisco routers, and networks that connect to only one Cisco router.

3. Explain how *Router1* can use the information of its LSDB to create routing table entries displayed in Step 5.

Exercise 5-c. Configuring OSPF on Linux PCs

Next you configure OSPF on *PC3* with Quagga. The syntax of the Quagga commands is very similar to the corresponding IOS commands. In addition to setting up another OSPF router, you will observe OSPF messages.

Step 1: On *PC3,* makes sure that IPv4 forwarding is enabled and reverse path filtering is disabled with

```
PC3$ sudo sysctl -w net.ipv4.ip_forward=1
PC3$ sudo sysctl -w net.ipv4.conf.all.rp_filter=0
```

Step 2: Start two instances of Wireshark to capture the traffic on the *eth0* and *eth1* interfaces of *PC3.* Set a display filter to "ospf".

Step 3: Restart the zebra and OSPF processes on *PC3* with

```
PC3$ sudo service zebra restart
PC3$ sudo service ospfd restart
```

Establish a *Telnet* session to the *ospfd* process using

```
PC3$ telnet localhost 2604
```

As password, use the default password "zebra". If this does not work, lookup the password in the OSPF configuration file by typing

```
PC3$ sudo more /etc/quagga/ospfd.conf
```

Step 4: On *PC3*, proceed with the configuration of OSPF. The commands are

```
ospfd> enable
ospfd# configure terminal
ospfd(config)# router ospf
ospfd(config-router)# network 10.0.0.0/8 area 1
ospfd(config-router)# router-id 10.0.3.33
ospfd(config-router)# no passive-interface eth0
ospfd(config-router)# no passive-interface eth1
ospfd(config-router)# end
```

Different from Cisco IOS, the command to enable OSPF (`router ospf') does not use a *process-id*. Also, there is an explicit command to set the *router-id*. The command is necessary since Q*uagga* does not assign a default value for the *router-id*.

Step 5: Once OSPF is started, the LSDB at *PC3* (as well as all other routers) should get updated. The commands to view the LSDB are identical to those on the Cisco routers. Display the LSDB with the commands

```
ospfd# show ip ospf database
ospfd# show ip ospf database network
ospfd# show ip ospf database router
```

and save the output.

Then, terminate the *Telnet* session with the command

```
ospfd# exit
```

Step 6: When you set up OSPF on *PC3* in Step 4, *PC3* and the other OSPF routers started to exchange OSPF messages. These messages have been captured by *Wireshark*. Explore the different types of OSPF messages, and try to infer their meanings. The following are the message types:

- Hello Packet,
- DB Description,
- LS Request,
- LS Update,
- LS Acknowledge.

For each message type answer the following questions:

- Which type of encapsulation is used for OSPF messages (TCP, UDP or other)?
- Who are the senders and receivers of the messages? Which messages use multicast addresses?
- Are these messages sent in regular time intervals or only when certain events happen? In the former case, what is the time interval? In the latter case, what is the event?
- What is the purpose of the message?

Step 7: The LS Update messages contain one or more LSA, which are used to populate the LSDB. Select one captured LS Update and relate it to a router link state or a network link state in the LSDB of *PC3*.

Step 8: Save the OSPF messages captured by one of the two Wireshark applications and then terminate the two Wireshark traffic captures. (Save the details of the captured traffic as a text file.)

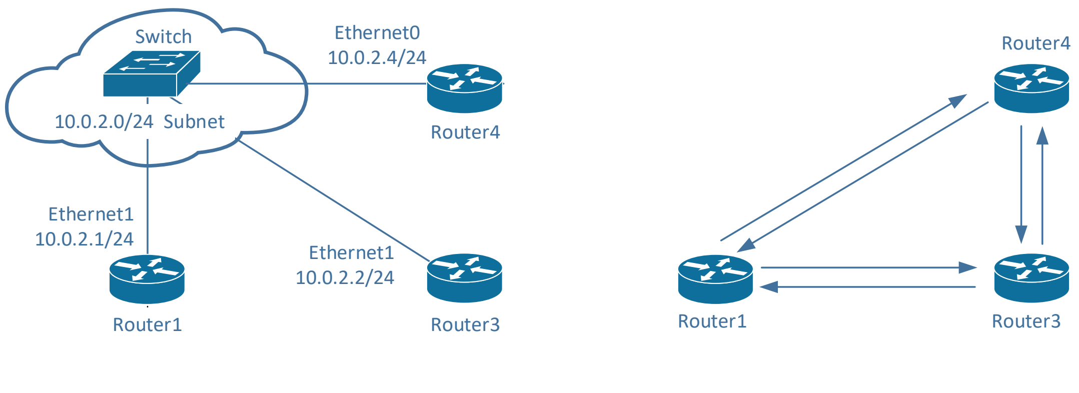
Lab Questions/Report

1. Provide the answers to the questions in Step 6.
2. For your answer in Step 7, provide the details of the LS Update message and show the relevant LSDB entry of the router link state or network link state. Justify why the link state and the LS Update message relate to each other.

Part 6. OSPF Metrics and Traffic Engineering

An OSPF router uses the content of the LSDB to generate a map of the network topology in the form of a directed graph. Each directed edge in the graph is assigned a cost, which is generally referred to as *metric*. The router then runs Dijkstra's shortest path algorithm to all known destinations and sets the routing table based on the result.

Figure 4.5 illustrates the construction of the directed graph. For the network in Figure 4.5(a) where three routers are connected by a switch, the resulting directed graph is shown in Figure 4.5(b). The metric of a directed edge is configured at the interface of the router where the edge originates. For example, the metric of the edges from *Router1* to *Router2* and from *Router1* to *Router3* is configured at the *Ethernet1* interface of *Router1*.



(a) Three routers connected to a switch. (b) Directed graph.
Figure 4.5. Construction of the directed graph.

The interfaces of OSPF routers are configured with a default metric. By changing the values of the metrics, network operators can control the path of network traffic. In practice, network operators periodically adjust the metrics of routers to achieve given objectives, e.g., load balancing or minimizing delays. This is referred to as *traffic engineering*.

Default metric of Cisco routers
On Cisco routers the default metric of an interface in OSPF is obtained by rounding down the ratio of 100 Mbps and the link rate of the interface. If the ratio is less than 1, the metric is set to 1.

In the following, we assume that Ethernet interfaces of the Cisco routers have a rate of 100 Mbps or more, which results in a default metric of 1. (This is a bit inconsistent since Cisco uses the interface names Ethernet0, Ethernet1, etc. for 10 Mbps Ethernet). However, 10 Mbps Ethernet is rarely encountered these days.

Exercise 6-a. Changing routing metrics

Step 1: Continue with the configuration from Part 5 (Exercise 5-c).

Step 2: On *Router4,* disable IP forwarding with the commands

```
Router4# configure terminal
Router4(config)# no ip routing
Router4(config-router)# end
```

With this, *Router4* is no longer acting as an IPv4 router. The commands above also terminate the OSPF routing process on *Router4*.

Step 3: Run a traceroute from *PC1* to *PC4* with

```
PC1$ traceroute 10.0.4.44
```

and a traceroute from *PC4* to *PC1* with

```
PC4$ traceroute 10.0.1.11
```

Take screen captures of the output of both commands.

You observe that the routes shown by the *traceroute* commands do not include *PC3*, even though the path *PC1* ↔ *PC3* ↔ *Router3* ↔ *PC4* traverses fewer routers than the alternative path *PC1* ↔ *Router1* ↔ *Router2* ↔ *Router3* ↔ *PC4*. This is different from what you observed with RIP (Exercise 2-b)!

Step 4: To explain the routes observed in Step 3, investigate the link metrics.

Go back to the saved output of the LSDB of *PC3* from Exercise 5-c (Step 5). In the output of '`show ip ospf database router`', search for the field "Metrics", which is displayed for each link.

- Record the metrics displayed for the links of the Cisco routers and the metrics of the links of *PC3*.

Other methods to view the link metrics

Link metrics are also contained in LS Update messages. Yet, another method to view the link metrics is to use the command `show ip ospf interface'. Here, the link metric is listed as a "cost". The command exists on both Cisco routers and the *ospfd* process of Quagga. (Recall that you have open a Telnet session to the *ospfd* process with `telnet localhost 2604'.)

Step 5: Use the output of the LSDB from Step 4 to construct a graph of the network. Label the interfaces of the OSPF routers (*Router1*, *Router2*, *Router3*, *PC3*) with their metric.

Use the graph to explain the results of the *traceroute* commands from Step 3.

Step 6: Next, change the metric of the eth0 interface at *PC3*. Establish a *Telnet* session to the *ospfd* process with

```
PC3$ telnet localhost 2604
```

and issue the commands

```
ospfd> enable
ospfd# configure terminal
ospfd(config)# interface eth0
ospfd(config-if)# ip ospf cost 1
ospfd(config-if)# end
ospfd# show ip ospf interface
ospfd# exit
```

The command `` `show ip ospf interface' ``, now shows that the metric/cost of the *eth0* interface of *PC3* is set to 1. (The metric/cost of interface *eth1* remains unchanged.)

After some time (10-30 seconds), all other OSPF routers should have received the change of the metric. Confirm the change of the metric by repeating the command `` `ip ospf database router' ``, and parse the output.

Step 7: Repeat Step 3 and take screen captures of the output of both commands.

- Explain the output of the two traceroute commands and relate it to the change of the metric in Step 6.
- Which route passes through *PC3*, and why?

Lab Questions/Report

1. Include the screenshots of the *traceroute* commands from Step 4 (before the metric was changed) and from Step7 (after the change).
2. Include the graph from Step 5, and use it to explain the outcomes of the traceroute command in Step 4.
3. Modify the graph due to the change of the link metric in Step 6, and use it to explain the outcomes of the traceroute command in Step 7.

Exercise 6-b. Observing convergence of OSPF

In comparison to the distance vector protocol RIP, the link state routing protocol OSPF quickly adapts to changes in the network topology. In this exercise you observe the interactions of OSPF after a change to the network topology. As in the RIP example in Exercise 3, you will create a link failure by disabling the *eth0* interface of *PC3*.

Step 1: Re-enable *Router4* as an OSPF router by issuing the configuration commands from Step 1 of Exercise 5-b.

Wait about a minute to make sure that the LSDB of all routers are updated. Verify this with the commands from the previous exercises for displaying the LSDB content.

Step 2: Run a traceroute from *PC4* to *PC1* to verify that the routing path from *PC4* to *PC1* passes through *PC3* with

```
PC4$ traceroute 10.0.1.11
```

Take a screenshot of the output.

Then, issue a *ping* command from *PC4* to *PC1*, with

```
PC4$ ping 10.0.1.11
```

Do not terminate the *ping* command until this exercise is completed in Step 4.

Step 3: On *PC3*, disable the *eth0* interface on *PC3* with the command

```
PC3$ sudo ip link set dev eth0 down; date
```

This disconnects the Ethernet connection between the *eth0* interface on *PC3* to the switch. When the *ping* is no longer successful.

The `date` command in the above command line takes a timestamp, when the interface is disabled.

Step 4: On PC4, wait until the *ping* command is successful again, that is, ICMP Echo Reply messages are returned to *PC4*.
Repeat the *traceroute* from Step 2 and take a screenshot of the output.

a) How long did it take to repair the path?
b) How does it compare to what you observed for RIP in Part 3?

Step 5: On PC5, re-enable enable the *eth0* interface with the command

```
PC3$ sudo ip link set dev eth0 up; date
```

Step 6: Go back to *PC4* and run a *traceroute* to *PC1*. When the traceroute displays IP address 10.0.3.33, the route has changed back to what you saw in Step 1.

- How long did it take to change back to the old route?

Step 7: Next, you repeat the disabling/enabling of the eth0 interface (Step 3 and Step 5), but now focus on the OSPF messages. Start *Wireshark* traffic captures for the traffic of *PC3* on both interfaces and set a display filter to “ospf”.

Step 8: Repeat Step 2 and observe the OSPF messages sent and received by *PC3*.

a) How quickly are OSPF messages sent after the cable is disconnected? Who sends messages to whom?

a) How many OSPF messages are sent (within a few seconds after the interface on *PC3* is disabled)?

b) Which type of OSPF message do you observe? Which information do the messages convey?

Step 9: Repeat Step 5 to re-enable the *eth0* interface on *PC3*. Save the traffic capture of *Wireshark* and terminate the traffic captures.

Step 10: On *PC3*, disable OSPF by stopping the *ospfd* process. The command is

```
PC3$ sudo service ospfd stop
```

Step 11: On all Cisco routers, clean up the routing tables and disable OSPF. The commands for *Router1* are

```
Router1# configure terminal
Router1(config)# no ip routing
Router1(config)# ip routing
Router1(config)# end
```

Repeat the commands on *Router2*, *Router3*, and *Router4*.

Lab Questions/Report

1. Include the screenshots from Step 2 and Step 4, which show the routes from *PC1* to *PC4* before and after the interface at *PC3* is disabled.
2. Provide answers to the questions from Step 4, Step 6, and Step 8.

Part 7. Configuring the Border Gateway Protocol (BGP)

The last part of this lab provides some exposure to the interdomain Border Gateway Protocol (BGP), which determines paths between autonomous systems on the Internet. The exercises in this lab only cover the basics of BGP. Essentially, you learn how to set up an autonomous system and observe BGP traffic between autonomous systems. BGP is a distance vector protocol that uses a path vector algorithm, where routers exchange full path information of a route. An important feature of BGP is that it permits to define *routing policies*, which can be used by a network to specify which type of traffic it is willing to allow to pass through it. The version of BGP used in the following exercise, is BGP version 4 (BGP-4).

The network configuration for this part is shown in Figure 4.6, and the IP configuration information is given in Tables 4.6 and 4.7. The network has four autonomous systems with AS numbers 100, 200, 300, and 400.

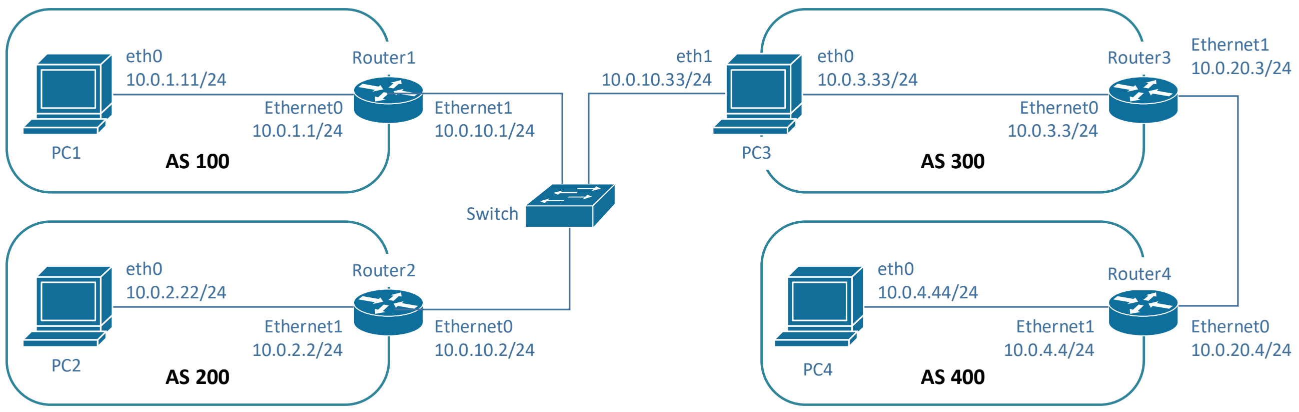


Figure 4.6. Network topology for Part 7.

BGP routers exchange routing information over a TCP connection. BGP routers that have a TCP connection established are called *BGP peers*, or simply *peers*, and the connection is called a *BGP session*. A BGP session between peers in different autonomous systems is said to run *external BGP (eBGP*). A BGP session between peers are in the same autonomous system is said to run *internal BGP (iBGP).*

Cisco Router	Interface Ethernet0	Interface Ethernet1
Router1	10.0.1.1/24	**10.0.10.1/24**
Router2	**10.0.10.2/24**	10.0.2.2/24
Router3	10.0.3.3/24	**10.0.20.3/24**
Router4	**10.0.20.4/24**	10.0.4.4/24

Table 4.6. IPv4 addresses of Cisco routers.

Linux PC	Interface *eth0*	Interface *eth1*	Default gateway
PC1	10.0.1.11/24	Disabled	10.0.1.1
PC2	10.0.2.22/24	Disabled	10.0.2.2
PC3	10.0.3.33/24	**10.0.10.33/24**	–
PC4	10.0.4.44/24	Disabled	**10.0.4.4**

Table 4.7. IPv4 addresses of PCs.

Exercise 7-a. Network setup and IPv4 configuration

The network topology is quite different from earlier parts of the Labs. Make sure that if you continue from Part 6, you only need to change the IP addresses shown in bold in Tables 4.6 and 4.7. However, the Ethernet connections have to be rewired. All routing configurations, such as RIP routing and default routes have to be removed.

Step 1: Connect the Ethernet interfaces of the Linux PCs and the Cisco router as shown in Figure 4.6. The network topology is quite different from earlier parts of this lab. Many PCs and Cisco routers are directly connected without a switch. Using a switch is an alternative setup which does not impact the ability to communicate.

Step 2: Configure *PC1, P2*, and *PC4* with the IP addresses and default gateways listed in Table 4.6. These PCs run as hosts. If you continue from Part 6, you only need to change the addresses that are shown in bold.

Step 3: Next, configure the IP addresses of *PC3*, which is again configured as an IP router.

Changing IP addresses of Linux interfaces
When you change the IP address of a network interface, do not forget to delete the old IP address. You can change the IP address of *PC3 (eth1)* from Part 6 to Part 7 with the commands

```
PC3$ sudo ip addr del 10.0.1.33/24 dev eth1
PC3$ sudo ip addr add 10.0.10.33/24 dev eth1
```

On Cisco routers, adding a new IP address deletes the old address.

On *PC3*, Check that the *ripd* and *ospfd* processes are not running by

```
PC3$ sudo service ripd status
PC3$ sudo service ospfd status
```

Terminate (stop) the processes, if they are running.

Step 4: Next, set up the IPv4 addresses of the Cisco routers according to Table 4.7. If you continue from Part 6, you only need to change the addresses that are shown in bold. Make sure that the RIP and OSPF protocols are disabled.

The commands for *Router1* are given below. The configuration of the other Cisco routers is done accordingly.

```
Router1# configure terminal
Router1(config)# no ip routing
Router1(config)# ip routing
Router1(config-router)# interface Ethernet0
Router1(config-if)# ip address 10.0.1.1 255.255.255.0
Router1(config-if)# no shutdown
Router1(config-if)# interface Ethernet1
Router1(config-if)# ip address 10.0.10.1 255.255.255.0
Router1(config-if)# no shutdown
Router1(config-if)# end
```

Step 5: At this time, PCs and Cisco routers that are directly connected or are connected to the same switch should be able to *ping* each other. Verify that this is the case and correct the configuration if a *ping* fails.

Exercise 7-b. eBGP configuration of Cisco routers

Here, you configure the Cisco routers and *PC3* as BGP routers. You assign routers to autonomous systems and establish eBGP sessions.

Below we summarize the Cisco IOS commands that are used to enable BGP.

IOS mode: global configuration

`router bgp <ASnumber>`

Enables the BGP routing protocol, and sets the autonomous system number to *<ASnumber>*. The command enters the router configuration mode with the following prompt:

```
Router1(config-router)#
```

`no router bgp <ASnumber>`

Disables the BGP routing process.

IOS mode: privileged EXEC

`show ip bgp`

Displays the BGP routing table.

`show ip bgp neighbors`

Displays the neighbors, also called peers, of this BGP router.

`show ip bgp paths`

Displays the BGP path information in the local database.

`clear ip bgp *`

Deletes BGP routing information.

<u>IOS mode: router configuration</u>

`network` ***`10.0.1.0`*** **`mask`** ***`255.255.255.0`***

Specifies that subnet *10.0.1.0/24* will be advertised by the *local* BGP process.

`neighbor` ***`10.0.10.2`*** **`remote-as`** ***`2`***

Adds the BGP router with IP address *10.0.10.2* of AS 2 as a neighbor to the BGP neighbor table.

`timers bpg` ***`<keepalive> <holdtime>`***

Sets the values of the *`<keepalive>`* and *`<holdtime>`* timers of the BGP process. BGP routers exchange periodic messages to confirm that the connection between the routers is maintained. The interval between these messages is *`<keepalive>`* seconds (default: 60 seconds). The number of seconds that a BGP router waits for any BGP message before it decides that a connection is down (default: 180 seconds).

Step 1: Configure the Cisco routers to run BGP with the autonomous system numbers shown in Figure 4.6**Error! Reference source not found.**. The routers must know the AS number of their neighbors. Below is the configuration for *Router1. Router1* is in AS 200, and has neighbors in AS 100 and in AS 300.

```
Router1# configure terminal
Router1(config)# router bgp 100
Router1(config-router)# neighbor 10.0.10.2 remote-as 200
Router1(config-router)# neighbor 10.0.10.33 remote-as 300
Router1(config-router)# network 10.0.1.0 mask 255.255.255.0
Router1(config-router)# end
Router1# clear ip bgp *
```

The first command starts BGP and assigns *Router1* to AS 100. The next two commands configure *Router2* in AS 200 and *PC3* in AS 300 as neighbors. The next command configures the router to advertise the prefix 10.0.1.0/24. The last command cleans up the routing table.

Configure *Router2*, *Router3*, and *Router4* as BGP routers following the instructions above. Set up the following neighbor relationships

Router	Neighbors
Router2	*Router1* (AS 100), *PC3* (AS 300)
PC3	*Router1* (AS 100), *Router2* (AS 200)
Router3	*Router4* (AS 400)
Router4	*Router3* (AS 300)

Do not set up neighbor relationships between *Router3* and *PC3*! This will be later configured as an iBGP session.

Step 2: Once the Cisco routers have been configured, you should be able to ping *PC2* from *PC1*. On *PC1*, issue a `ping` command to *PC2*. Issue the command.

```
PC1$ ping -c3 10.0.2.22
```

You should also be able to do a ping from *PC4* to IP address 10.0.3.3.

```
PC4$ ping -c3 10.0.3.3
```

- The ping from *PC4* to 10.0.3.33 is not successful. Why?

Step 3: Once the BGP peers have exchanged routing information, you can display the routing table and BGP information. BGP routing table. On each Cisco router, save the output of the following commands:

```
Router1# show ip route
Router1# show ip bgp
Router1# show ip bgp summary
```

The first command is the usual IP routing table display. The second command shows the table of the BGP protocol, which includes information about the routing paths. The last command displays the BGP sessions to peers.

Take a screen capture of the output of these commands at *Router1*.

Lab Questions/Report

1. Provide the screen capture from Step 3.

Exercise 7-c. eBGP configuration of Linux PC

Next you configure *PC3* as a BGP peer. The syntax of the Quagga commands is close to that of Cisco IOS commands. In this exercise, you will also observe BGP traffic between *PC3* and its peers.

Step 1: Make sure that IPv4 forwarding is enabled by checking

```
PC3$ sudo sysctl  -w net.ipv4.ip_forward=1
```

If necessary, enable IPv4 forwarding. Different from Part 6, for this network topology we do not have to worry about reverse path filtering.

Step 2: Start two instances of *Wireshark* to capture the traffic on the eth0 and eth1 interfaces of *PC3*. Set a display filter to "*bgp*".

Step 3: Restart the *zebra* and *bgpd* processes on *PC3* with

```
PC3$ sudo service zebra restart
PC3$ sudo service bgpd restart
```

Establish a Telnet session to the *bgpd* process using

```
PC3$ telnet localhost 2605
```

As password, use the default password `'zebra'`. If this does not work, lookup the password in the BGP configuration file by typing

```
PC3$ sudo more /etc/quagga/bgpd.conf
```

Step 4: On *PC3*, configure BGP using the same commands as used in Cisco IOS. The commands are:

```
bgpd> enable
bgpd# conf term
bgpd(config)# router bgp 300
bgpd(config-router)# neighbor 10.0.10.2 remote-as 200
bgpd(config-router)# neighbor 10.0.10.1 remote-as 100
bgpd(config-router)# network 10.0.3.0 mask 255.255.255.0
bgpd(config-router)# end
```

Step 5: Now, *PC3* establishes BGP sessions to *Router1* and *Router2*, and exchange routing information. You can display the BGP information at *PC3* with the commands

```
bpgd# show ip bgp
bpgd# show ip bgp summary
```

Take a screen capture of the output of the commands.

In the output of `'show ip bgp'` there are multiple lines for networks 10.0.1.0/24 and 10.0.2.0/24. Explain the significance of this.

Then, terminate the `Telnet` session with

```
bpgd # exit
```

Step 6: Verify the BGP configuration on *PC3* by issuing a *ping* to *PC1* and *PC2* with

```
PC3$ ping -c3 10.0.1.11
PC3$ ping -c3 10.0.2.22
```

Step 7: Next take a look at the BGP protocol messages that were captured by *Wireshark*. You only see BGP messages on the eth1 interface of *PC3*. Explore the different types of BGP messages, and try to infer their meanings. The following are the message types that you should observe:

- OPEN message,
- KEEP ALIVE message,
- UPDATE message,

Answer the following questions:

a) Which type of encapsulation is used for BGP messages (TCP, UDP or other)?

b) Are these messages sent in regular time intervals or only when certain events happen? In the former case, what is the time interval? In the latter case, what is the event?
Inspect the “Path attributes” and the “Network Layer Reachability Information (NLRI)” information in the UPDATE messages that are sent and received by *PC3*.

Step 8: Save all UPDATE messages that are captured by Wireshark. The display filter to show only captured UPDATE messages is “*bgp.type==2*”. Save the details of the captured traffic. Do not terminate the *Wireshark* applications.

At this time, AS 100, AS 200, and AS 300 can exchange traffic, as can AS 300 and AS 400, but the routing across AS 300 is not yet configured. This will be done in the remaining exercises.

Lab Questions/Report

1. Include the screen snapshot from Step 4. Explain why the BGP routing table has multiple entries for networks 10.0.1.0/24 and 10.0.2.0/24.

2. Provide the answers to the questions in Step 7. (You can use the details of one of the UPDATE messages saved in Step 7.

3. Create a table for the captured UPDATE messages (from Step 8), with one row for each UPDATE message. The columns of the table are: source IP address, destination IP address, advertised network prefix (NLRI), and AS PATH.

Exercise 7-d. iBGP configuration

You next establish a iBGP session between *PC3* and *Router3* so that prefixes that are advertised to PC3 are sent to *Router3*, and vice versa. As we will see, advertising the routes is not sufficient and another step must be taken. To observe the need for this additional step, the iBGP configuration is done in two phases.

Phase 1: Enabling iBGP

Step 1: Make sure the Wireshark sessions for the traffic on *PC3 (eth0)* and *PC3 (eth1)* are active.

Step 2: Take a look at the routing tables at *Router3* and *PC3*.
On *PC3* type

```
PC3$ netstat -rn
```

On *Router3*, the command is

```
Router3# show ip route
```

Take screenshots of the output of the commands.

The routing tables do not have entries for some of the subnets in Figure 4.6. Note which ones are missing and explain why they are missing.

Step 3: On *Router3*, configure an iBGP session to *PC3* with the commands

```
Router3# show ip bgp
Router3# configure terminal
Router3(config)# router bgp 300
Router3(config-router)# neighbor 10.0.3.33 remote-as 300
Router3(config-router)# end
```

Step 4: On *PC3*, set up a *Telnet* session to the *bgpd* process with

```
PC3$ telnet localhost 2605
```

and then issue the commands

```
bgpd> enable
bgpd# conf term
bgpd(config)# router bgp 300
bgpd(config-router)# neighbor 10.0.3.3 remote-as 300
bgpd(config)# end
```

Step 5: There are now three things to observe:
First, observe the BGP UPDATE messages that are sent between *Router3* and *PC3*.These messages contain updates where the two BGP peers send each other all of their known routes.

Step 6: The second observation concerns the BGP routing table. On *PC3*, issue the command

```
bgpd# show ip bgp
```

Take a screen snapshot of the output.

On *Router3*, type

```
Router3# show ip bgp
```

The BGP routing tables now have entries for subnets 10.0.1.0/24, 10.0.2.0/24, 10.0.3.0/24, and 10.0.4.0/24. The entries that have been added through the iBGP session are marked with an "*i*". Take note of the *Next Hop* value of these entries.

Step 7: The third observation is made when looking at the IPv4 routing table at *PC3* and *Router3*. On *PC3*, you have to terminate the Telnet session and then list the kernel routing table. The commands are

```
bgpd# exit
PC3$ netstat -rn
```

Take a screen snapshot of the output at *PC3*.

On *Router3*, type

```
Router3# show ip route
```

You can observe that the routing tables have not changed from Step 2.

Why have the routing tables of *PC3* and *Router3* not changed?
The reason is found in the BGP routing tables. In the output of the `‘show ip bgp’` command at *Router3* for subnet 10.0.1.0/24, the Next Hop entry shows *10.0.10.1.* But *Router3* does not know how to reach 10.0.10.1 since it has no match for this address in its routing table. Without a valid *Next Hop* address, *Router3* cannot create an entry for the IPv4 routing table entry for network 10.0.1.0/24.

Lab Questions/Report

1. Include the captures of the IPv4 routing tables from Step 2 (before the configuration of the iBGP sessions).
2. Include the captures of the BGP routing tables from Step 6 (after the configuration of the iBGP sessions).
3. Include the captures of the IPv4 routing tables from Step 7 (after the configuration of the iBGP sessions).
4. Summarize your observations about the captured IPv4 and BGP routing tables.

Phase 2: Resolving the Next Hop address

This issue with iBGP, that is, that the routing tables could not be updated, highlights a difficulty of configuring BGP routing involving iBGP sessions. There are multiple solutions to this issue, all of which are found in practice. The following instructions pursue a solution that is the most straightforward for the topology in Figure 4.6.

The root cause of the missing routing table entries is as follows: *iBGP* peers forward BGP messages, which they received over an *eBGP* session to each other, without modifying them. In particular, the *NEXT_HOP* attribute is not updated. Then, when an *iBGP* peer receives such a forwarded BGP message it may not know how to reach the node listed in the *NEXT_HOP* attribute.

A solution is to force an iBGP peer to replace the NEXT_HOP attribute with its own IP address (where it uses the IP address that connects to the same subnet as the other iBGP peer). The replacement is done for all UPDATE messages that are received on an eBGP session before the messages are forwarded to an iBGP peer.

The above can be accomplished with a single command. Once the commands are issued there will be IPv4 routing table updates at all BGP routers in Figure 4.6.

Step 1: On *Router1*, configure an iBGP session to *PC3* with the commands

```
Router3# conf term
```

```
Router3(config)# router bgp 300
Router3(config-router)# neighbor 10.0.3.33 next-hop-self
Router3(config-router)# end
```

Step 2: On *PC3*, re-establish the *Telnet* session to the *bgpd* process with

```
PC3$ telnet localhost 2605
```

and then type

```
bgpd> enable
bgpd# conf term
bgpd(config)# router bgp 300
bgpd(config-router)# neighbor 10.0.3.3 next-hop-self
bgpd(config)# end
```

Step 3: Display the BGP routing table on both *Router3* and *PC3*. Also, display the IPv4 routing tables on both *Router3* and *PC3*.

Take snapshots of the BGP and IPv4 routing tables at *PC3*.

- Confirm that the Next Hop entries in the BGP routing table of *Router3* and *PC3* are all reachable.
- Confirm that *PC3* and *Router3* have IPv4 table entries for the subnets in AS 100, AS 200, and AS 400.
- Check the NEXT_HOP attribute in the UPDATE messages that are sent by *PC3* (on interface *eth0*) to *Router3*. Note the difference to the NEXT_HOP attribute in the UPDATE messages sent by *PC3* from Step 6.

Step 4: Observe that *PC3* has sent UPDATE messages to *Router1* and *Router2*, which contain advertisements for the prefix 10.0.4.0/24. With these messages *Router1* and *Router2* will create entries for network 10.0.4.0/24 in the BGP routing table and in the IPv4 routing table.

To confirm this, display both tables at *Router1* and take a screen snapshot.

Step 5: Now *PC4* can exchange messages with *PC1* and *PC4*. Confirm this by issuing ping commands from *PC4*

```
PC4$ ping -c3 10.0.1.11
PC4$ ping -c3 10.0.2.22
```

Step 6: On *Router4*, run a *ping* command from *Router4* to *PC1* and *PC4*

```
Router4# ping 10.0.1.11
Router4# ping 10.0.2.22
```

a) Why do the *ping* commands from *Router4* to 10.0.1.11 and 10.0.2.22 fail?
b) What needs to be done to fix this?

Step 7: Save the traffic capture of *Wireshark* but do not terminate the Wireshark sessions.

Lab Questions/Report

1. Include the BGP routing table and IPv4 routing table of *PC3* and *Router3* from Step 3. Use the tables to support the observations in the first two bullets in Step 3.

2. Include the screen captures of the BGP and IPv4 routing tables of *PC1* (Step 4) to confirm that *Router1* has received updates for AS 400 from *PC3*.

3. Provide your answers to the questions in Step 6.

Exercise 7-e. Routing policy (selective transit)

A major task of BGP is to realize routing policies between autonomous systems. Routing policies are implemented by either rejecting certain network prefixes in received UPDATE messages or by not including certain prefixes in transmitted UPDATE messages.

In this exercise you implement a simple routing policy. Suppose AS 300 is a service provider, which has AS 100 and AS 400 as its customers. Since AS 200 is not a customer, AS 300 does not want to accept traffic to and from AS 200. This is referred to as *selective transit*, i.e., AS 300 will transport traffic between AS 100 and AS 400, but not between AS 200 and AS 400,

With the configuration from the previous exercises, all autonomous systems accept traffic and routing updates from all other autonomous systems.

Step 1: Check that the *Wireshark* sessions for the traffic on the interfaces of *PC3* are still running.

Step 2: On *PC1* and *PC2*, run the following *ping* commands

```
PC1$ ping 10.0.4.44
PC2$ ping 10.0.4.44
```

Let the *ping* commands running until the end of this exercise.

Step 3: The remaining configuration is done exclusively on *PC3*. Establish a *Telnet* session to the *bgpd* process with

```
PC3$ telnet localhost 2605
```

and then type

```
bgpd> enable
bgpd# conf term
bgpd(config)# ip prefix-list MYLIST permit 10.0.1.0/24
```

The last command defines a prefix list that is assigned the name *"MYLIST"*. The argument '`permit 10.0.1.0/24`' states that all traffic from 10.0.1.0/24 is permitted. Everything that is not permitted is blocked. (Note: A prefix list can be extended to contain multiple prefixes and ranges of prefixes. For example, '`permit 0.0.0.0/0 le 32`') permits all IPv4 addresses.

Step 4: Next, apply the prefix list to the BGP configuration of *PC3*. The commands continue as follows:

```
bgpd(config)# router bgp 300
bgpd(config-router)# neighbor 10.0.10.1  prefix-list MYLIST in
bgpd(config-router)# neighbor 10.0.10.2  prefix-list MYLIST in
bgpd(config)# end
```

Here, the list is applied to both 10.0.10.1 and 10.0.10.2. This is necessary, since both routers advertise the 10.0.2.0/24 prefix.

Step 5: The last thing to do is to force a refresh of the BGP sessions with 10.0.10.1 and 10.0.10.2. This is done with the commands

```
bgpd# clear ip bgp 10.0.10.1 in prefix-filter
bgpd# clear ip bgp 10.0.10.2 in prefix-filter
```

Step 6: Now, check the output of the ping commands started in Step 2. The ping from *PC1* to *PC4* continues, but the ping form *PC2* to *PC4* has stopped.

Step 7: Analyze the impact of the configurations from Steps 3–5. Check the BGP routing table and the IPv4 routing table at *PC3* with the commands

```
bpgd# show ip bgp
bpgd# exit
PC3$ netstat -rn
```

You should see that the routing table entries for network 10.0.2.0/24 have disappeared.

Step 8: Now check the UPDATE messages that are sent between *PC3, PC2 and PC1*. The most recent UPDATE messages were exchanged after the commands in Step 5.

- Take a closer look at the UPDATE messages. Identify the UPDATE messages, where *Router3* withdraws the route to 10.0.2.0/24.

Step 9: Since *Router3* has no longer a routing table entry for 10.0.2.0/24 it does not send traffic to this network. How about the traffic that is sent by 10.0.2.0/24 into AS 300?

- First check the routing table at *Router2* (`show ip route`). Verify that there are still entries for networks 10.0.3.0/24 and 10.0.4.0/24.
- Next, change the display filters of the *Wireshark* applications to "icmp and ip.addr==10.0.2.22". This captures the traffic due to the ping command at *PC2*. The *Wireshark* on *PC3(eth1)* shows the *ICMP Echo Reply* messages, but the *Wireshark* on *PC3(eth0)* does not show any traffic from 10.0.2.22. This shows that *PC3* no longer forwards traffic from 10.0.2.0/24.

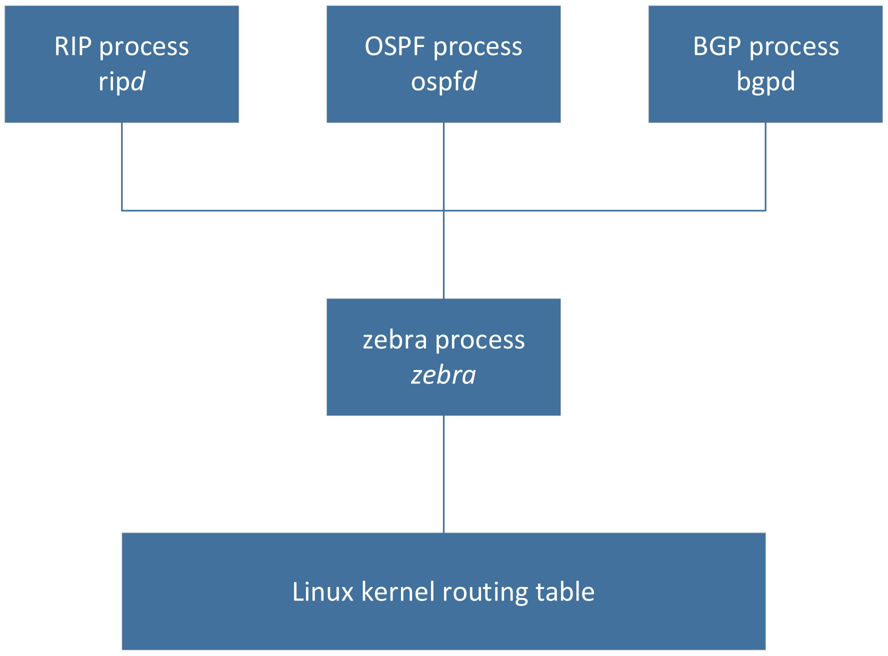
Appendix: An Introduction to Quagga

Quagga is a network routing software suite providing implementations of Open Shortest Path First (OSPF), Routing Information Protocol (RIP), Border Gateway Protocol (BGP) and IS-IS for Linux systems. The Quagga architecture consists of a routing manager (*zebra*) that manages the routing tables of various routing protocols and communicates with the Linux kernel.

The routing processes used in this lab and the routing protocols they manage are as follows:

Routing Process	Routing Protocol
`zebra`	Routing manager (not a routing protocol)
`bgpd`	BGP-4
`ripd`	RIPv2
`ospfd`	OSPFv2

To run Quagga, you must start the zebra service. You also need to start the service for each routing protocol that you configure.



1. Starting and stopping quagga processes

You start the *zebra* process by typing

```
PC1$ sudo service zebra start
```

You can verify if a *zebra* process is already running by typing

```
PC1$ sudo service zebra status
```

You terminate the *zebra* process with the command

```
PC1$ sudo service zebra stop
```

You can stop and restart the *zebra* process in a single command by typing

```
PC1$ sudo service zebra restart
```

To set up a routing process, you must first start the *zebra* process, and then start the routing protocol process. For example to start the process that runs the RIP routing protocol you type

```
PC1$ sudo service zebra start
PC1$ sudo service ripd start
```

As with the *zebra* process, you can query the status of the RIP process with the command `*ripd status*' and you can stop the process with the command `*ripd stop*'. When you type `*zebra stop*', then all routing protocol processes are stopped as well.

For the *zebra* process and all other routing processes, there is a configuration file which is read when the process is started. The configuration files are located in the directory `/etc/quagga`, and have names `zebra.conf`, `ripd.conf`, etc. The configuration files look similar to the configuration files of Cisco IOS, and contain commands that are executed when the process is started.

2. Configuring the routing protocol processes

After you start the services for zebra or a routing protocol (e.g., ripd) you can configure the services. To configure any of the routing processes, establish a *Telnet* session with the TCP port number of the service process. Each service listens on a specific port for incoming requests to establish a *Telnet* session. The port numbers s are as follows:

Routing Process	TCP Port number
zebra	2601
ripd	2602
ospfd	2604
bgpd	2605

If you establish a *Telnet* session to a routing process, you are asked for a password. The password is If the password is correct, a command prompt is displayed. You can find the password in the configuration file of the routing process, e.g., `/etc/quagga/ripd.conf`. Since the configuration files can be accessed only by users with sudo privileges, you can list the configuration file by typing

```
PC1$ sudo more /etc/quagga/zebra.conf
```

To access the `ripd` process, that is running on *PC1*, from *PC1* you type

```
PC1$ sudo telnet localhost 2602
```

This results in the following output:

```
labuser@PC1$ sudo telnet localhost 2602
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.24.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password: <enter password>
ripd>
```

At the prompt, you may type configuration commands. The *Telnet* session is terminated with the command

```
ripd> exit
```

3. Typing configuration commands

Once you have established a *Telnet* session to a routing process, you can configure the routing protocol of that process. The command line interface of the routing processes emulates the IOS command line interface, that is, the processes have similar command modes as Cisco IOS, and the syntax of commands is generally the same as the corresponding commands in Cisco IOS.

For example, the following commands configure the RIP routing protocol for network 10.0.0.0/8 on a Linux PC.

```
ripd> enable
ripd# configure terminal
ripd(config)# router rip
ripd(config-router)# version 2
ripd(config-router)# network 10.0.0.0/8
ripd(config-router)# end
ripd# exit
```

Here, in the default configuration of the *Quagga* processes, an enable password is not required.