Based on
Mastering Networks - An Internet Lab Manual
by Jörg Liebeherr and Magda Al Zarki

Adapted for
'Labo Computernetwerken'
by Johan Bergs, Nicolas Letor, Michael Voorhaen and Kurt Smolderen

Completed by
Josse Coen Armin Halilovic Jonas Vanden Branden Group 2

March 22, 2016

Lab 4

Dynamic Routing Protocols (RIP and OSPF)

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.

4.1 Prelab 4

Routing protocols

- *Distance Vector and Link State Routing Protocols*: Go to the website http://docwiki.cisco.com/wiki/Internetworking_Technology_Handbook and read the article about dynamic routing protocols. Review your knowledge of interdomain and intradomain routing, distance vector routing, and link state routing.
- *Zebra*: Go to the website of the Zebra fork Quagga at http://www.nongnu.org/quagga/ and study the information on the Quagga routing protocol software for Linux systems. Also find and read the man pages on zebra, ripd, ospfd and bgpd. Note: Quagga is a fork of the GNU Zebra project.
- *RIP*: Read the overview of the Routing Information Protocol (RIP) and study the commands to configure RIP on a Cisco router at http://www.routeralley.com/guides/rip.pdf.
- *OSPF*: Read the overview of Open Shortest Path First (OSPF) routing protocol and study the commands to configure OSPF on a Cisco router at http://www.routeralley.com/guides/ospf.pdf.

Prelab Questions

Question 1)
Provide the command that configures a Linux PC as an IP router (see Lab 3).
sysctl net.ipv4.ip_forward=1

Question 2)
What are the main differences between a distance vector routing protocol and a link state routing protocol? Give examples for each type of protocol.
** dit nakijken, ik heb de dingen wat gecopy pasted **

RIP is a distance vector routing protocol. OSPF is a link state routing protocol.
Distance vector routing will get the shortest path, while link state routing goes for the fastest path.

Distance vector: routing information is only exchanged between directly connected neighbors; a router cannot see beyond its own neighbors.
Link state: information is flooded throughout the link-state domain (an area in OSPF) to ensure all routers posses a synchronized copy of the area's link-state database.

Question 3)
What are the differences between an intradomain routing protocol (also called interior gateway protocol or IGP) and an interdomain routing protocol (also called exterior gateway protocol or EGP)? Give examples for each type of protocol.
Intradomain routing protocols are used for routing in one network. Examples are RIP and OSPF.
Interdomain routing protocols are used for routing between different networks. For example: BGP.

Question 4)
Which routing protocols are supported by the software package Zebra?
OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP-4, IS-IS for IPv4, OLSR

Question 5)
In the Zebra software package, the processes ripd, ospfd, and bgpd deal, respectively, with the routing protocols RIP, OSPF, and BGP. Which role does the process zebra play?
** gewoon ge copy pasted van de documentation, mag dat? **

zebra is an IP routing manager. It provides kernel routing table updates, interface lookups, and redistribution of routes between different routing protocols.

Question 6)
Describe how a Linux user accesses the processes of Zebra (zebra, ripd, ospfd, bgpd) processes to configure routing algorithm parameters?
To enable the processes, the user can edit the entries in /etc/quagga/daemons.
The processes can then be started/stopped with /etc/init.d/quagga command. For each process, a configuration file in /etc/quagga (e.g. ripd.conf) can be edited (or created if there is no file) to configure routing algorithm parameters.

Question 7)
What is the main difference between RIP version 1 (RIPv1) and RIP version 2 (RIPv2)?
RIPv1 is a classful routing protocol, whereas RIPv2 is a classless one.
This means that RIPv2 can send subnet masks in RIP updates to other routers but RIPv1 cannot.

Question 8)
Explain what it means to "run RIP in passive mode".

In passive mode, incoming RIP packets are processed, but does no RIP packets are transmitted.

Question 9)

Explain the meaning of “triggered updates” in RIP.

In RIP, a triggered update means that a router sends a RIP packet with a routing update, whenever one of its routing table entries changes.

Question 10)

Explain the concept of split-horizon in RIP?

The split-horizon method in RIP is used to prevent routing loops by forbidding a router from advertising a route back to the interface form which it was learned.

Question 11)

What is an autonomous system (AS)? Which roles do autonomous systems play in the Internet?

An autonomous system is a group of IP networks under the authority of a single administration. The entire Internet is carved up into a large number of autonomous systems.

Question 12)

What is the AS number of your institution? Which autonomous system has AS number 1?

Numbers 1-6: Assigned by ARIN

AS1: LVLT-1 - Level 3 Communications, Inc.,US

** uantwerpen nummer? **

Question 13)

Explain the terms: Stub AS, Multi-homed AS and Transit AS?

****

4.2 Lab 4

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, neighbouring 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 the two most common intradomain protocols, namely, the Routing Information Protocol (RIP) and the Open Shortest Path First (OSPF) protocol. Parts 1-3 of this lab deal with RIP, and Parts 4-5 are about OSPF.

This lab uses two different network configurations. The first network configuration, shown in Figure 4.1, is used in Parts 1-2, and is modified in Part 3 (Figure 4.3). The network configuration in Parts 4 and 5 is shown in Figure 4.4.

Part 1. Configuring RIP on a Cisco router

This lab starts with the same network topology as used in Part 5 of Lab 3. Different from Lab 3, where the routing tables were configured manually, here you run the routing protocol RIP to perform the same task. In Part 1, you configure RIP on the Cisco routers. In Part 2, you configure RIP on the Linux PCs.

RIP is one of the oldest dynamic routing protocols on the Internet that is still in use. This lab uses the latest revision of RIP, RIPv2 (RIP version 2). 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 each point-to-point link or LAN constitutes a hop.

Each RIP enabled router periodically sends the content of its routing table to all its neighbouring 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 neighbouring router, it updates its own routing table.

Figure 4.1 and Table 4.1 describe the network configuration for this part of the lab.

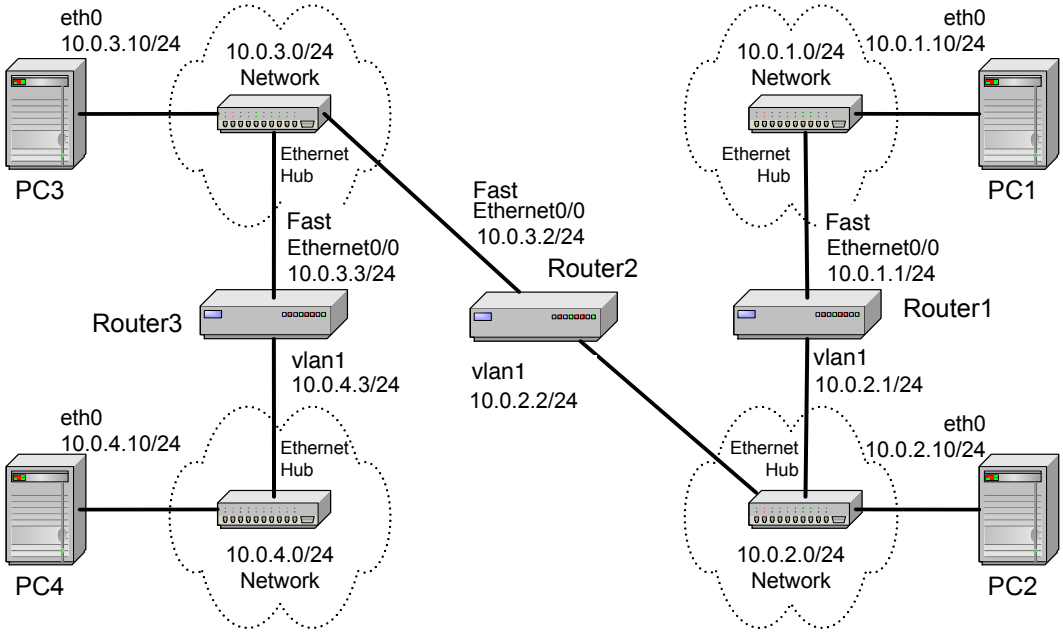

Figure 4.1: Network configuration for Parts 1 and 2.

Exercise 1. Configuring RIP on Cisco routers

Configure all three Cisco routers to run the routing protocol RIP. Once the configuration is completed, all Cisco routers 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.

The following can can be executed in the Global Configuration mode.

Linux PC	eth0	eth1
PC1	10.0.1.10/24	Disabled
PC2	10.0.2.10/24	Disabled
PC3	10.0.3.10/24	Disabled
PC4	10.0.4.10/24	Disabled
Cisco Router	**FastEthernet0/0**	**vlan1**
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: IP addresses

- Enable the routing protocol RIP on the local router, and enters the router configuration mode from the following prompt:

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

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

  ```
  router rip
  ```

- Disable RIP on the local router.

  ```
  no router rip
  ```

The following can can be executed in the Privileged EXEC mode.

- Enable a debugging mode where the router displays a message for each received RIP packet.

  ```
  debug ip rip
  ```

- Disable the debugging feature

  ```
  o debug ip rip
  ```

The following can can be executed in the Router Configuration mode.

- Associate the network IP address *Netaddr* with RIP. RIP sends updates only on interfaces where the network address has been associated with RIP.

  ```
  etwork Netaddr
  ```

- Disable RIP for the specified network address.

  ```
  no network Netaddr
  ```

- Set the interface *Iface* in RIP passive mode. On an interface in passive mode, the router processes incoming RIP packets, but does not transmit RIP packets.

  ```
  passive-interface Iface
  ```

- Enable active mode on interface *Iface*. This means that RIP packets are transmitted on this interface.

```
no passive-interface Iface
```

- Increase the metric (hop count) of incoming RIP packets that arrive on interface *Iface* by *value*, where *value* is a number.

```
offset-list 0 in value Iface
```

- Increase the metric of outgoing RIP packets that are sent on interface *Iface* by *value*.

```
offset-list 0 out value Iface
        \end{verbatim}
\item Disable the specified offset-list command for incoming RIP packets.
        \begin{cmdblock}
no offset-list 0 in value Iface
```

- Disable the specified offset-list command for outgoing RIP packets.

```
no offset-list 0 out value Iface
```

- Set the RIP version to RIPv2.

```
version 2
```

- Set the values of the timers in the RIP protocol. The timers are measured in seconds.

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

`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 packets with routing updates after a link failure (Default: 180 sec).

`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).

Example:

```
Router1(config-router)# timers basic 30 180 180 240
```

- Set the router to not perform triggered updates, when the next transmission of routing updates is due in time. If time is set to the same value as the update timer, then triggered updates are disabled. In RIP, a triggered update means that a router sends a RIP packet with a routing update, whenever one of its routing table entries changes.

```
flash-update-threshold time
```

1. Connect the the Linux PCs and the Cisco routers as shown in Figure 4.1. The PCs and routers are connected with Ethernet hubs.
2. Verify that the serial interfaces of the PCs are connected to the console port of the routers. PC1 should be connected to Router1, PC2 to Router2, and so on. Once the serial cables are connected, establish a minicom session from each PC to the connected router.
3. On Router1, Router2, and Router3, configure the IP addresses as shown in Table 4.1, and enable the routing protocol RIP. The commands to set up Router1 are as follows:

```
Router1> enable Password: <enable secret>
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-router)# interface FastEthernet0/0
Router1(config-if)# no shutdown
Router1(config-if)# ip address 10.0.1.1 255.255.255.0
Router1(config-if)# interface FastEthernet0/1
Router1(config-if)# no shutdown
Router1(config-if)# interface vlan1
Router1(config-if)# no shutdown
Router1(config-if)# ip address 10.0.2.1 255.255.255.0
Router1(config-if)# end
Router1# clear ip route *
```

The command `no ip routing` is used to reset all previous configurations related to routing (RIP, OSPF, etc). The command `clear ip route *` deletes all entries in the routing table. Make sure that all static routing entries are removed, since, in IOS, RIP does not overwrite static routing entries.

4. After you have configured the routers, check the routing table at each router by typing

   ```
   Router1# show ip route
   ```

   Each router should have four entries in the routing table: two entries for directly connected networks, and two other entries for remote networks that were added by RIP.

5. From each router, issue a ping command to the IP addresses of interfaces *FastEthernet0/0* and *vlan1* on all remote routers. For example, to issue a ping from Router1 to interface *FastEthernet0/0* on Router2, type

   ```
   Router1# ping 10.0.3.2
   ```

Once you can successfully contact the IP addresses of all routers, proceed to the next exercise.

Part 2. Configuring RIP on a Linux PC

In this part of the lab, you continue with the network configuration in Figure 4.1 and Table 4.1, and configure RIP on the Linux PCs.

In Figure 4.1, all Linux PCs are set up as hosts. Since hosts do not perform IP forwarding, they need not send routing messages. Therefore, when a routing protocol is configured on a host, the protocol is set to run in passive mode, where a host receives and processes incoming routing messages, but does not transmit routing messages. (We note that, normally, routing protocols are not enabled on hosts. Instead, one generally configures a static routing table entry for the default gateway. Obviously, when a routing protocol is enabled, there is no need to configure a default gateway.)

The configuration of routing protocols on Linux PCs in Lab 4 is done with the routing software package Quagga. Before starting the exercise, we give a brief tutorial on the Quagga software package. The tutorial focuses on the features used in the lab exercises and omits many interesting features of Zebra.

An Introduction to Quagga

Quagga is a software package that manages the routing tables of a Linux system, and that provides the ability to execute a variety of routing protocols. For this course we make use of Quagga, which is a fork of the GNU Zebra project and while the project has a new name, many of the references to Zebra still remain, e.g. there is still a `zebra` control process.

The Quagga architecture, shown in Figure 4.2, consists of a set of processes. The process `zebra` updates the routing tables and exchanges routes between different routing protocols. Each routing protocol has a separate process, and each routing process can be started, stopped, configured, and upgraded independently of the other routing processes. The process `zebra` must be invoked prior to starting and configuring any of the routing protocols. The routing processes used in this lab and the routing protocols they manage are shown in table 4.2.

Routing Process	Routing Protocol
ripd	RIPv1 and RIPv2
ospfd	OSPFv2 (Version 2)

Table 4.2: Quagga routing processes used for this lab.

(a) Adding the directory with Quagga commands to the search path
On Ubuntu systems, the script to start, stop and control the `zebra` process and its routing processes is located in directory `/etc/init.d`.

```
PC1% /etc/init.d/quagga start
```

(b) Starting and stopping Quagga processes

`/etc/init.d/quagga start`
Start the Quagga processes.

`/etc/init.d/quagga stop`
Terminate the Quagga processes.

`/etc/init.d/quagga restart`
Stop and restart the Quagga processes.

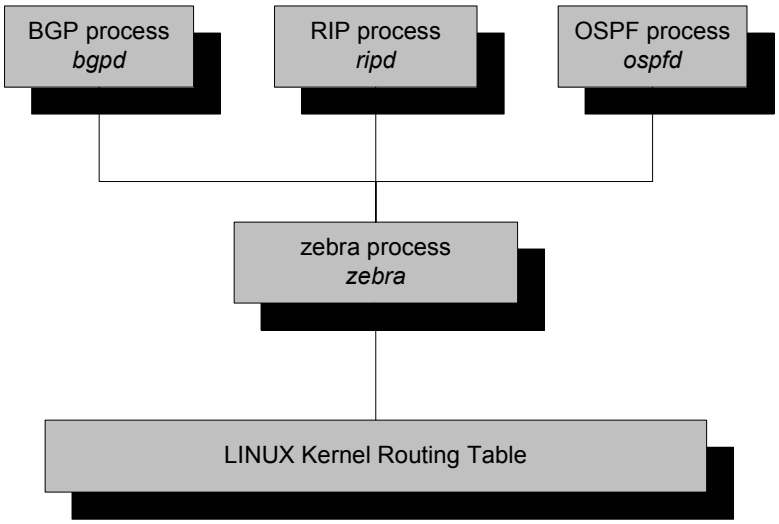

Figure 4.2: Quagga processes

To set up a routing process, you must enable the routing daemon in the Quagga configuration file `/etc/quagga/daemons` and than start the `quagga` service. For example to start the RIP routing protocol daemon, your `daemons` file should look as shown below. Afterwards, you can start the `zebra` and the `ripd` daemons by running `/etc/init.d/quagga start` or `/etc/init.d/quagga restart` in case Quagga was already running.

```
zebra=yes
bgpd=no
ospfd=no
ospf6d=no
ripd=yes
ripngd=no
isisd=no
```

Make sure the `zebra` daemon is always enabled as the other routing daemons depend on this process. When you type `/etc/init.d/quagga 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 `/usr/local/etc` or `/etc/quagga`, and have names `zebra.conf`, `ripd.conf`, etc. The configuration files look similar to the configuration files of IOS, and contain commands that are executed when the process is started.

(c) Configuring the `zebra` process and the routing protocol processes
After starting the `zebra` process or any of the routing protocol processes, you can configure each process by establishing a Telnet session to that process. Each process listens on a specific port for incoming requests to establish a Telnet session. The port numbers of the processes are as follows:

- 2601 - Zebra
- 2602 - ripd
- 2604 - ospfd

If you establish a Telnet session to a routing process, you are asked for a password. If the password is correct, a command prompt is displayed. For example, to access the ripd process on the local host you type:

```
PC1% telnet localhost 2602
```

This results in the following output.

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.20.1).
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
```

(d) 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 IOS, and the syntax of commands is generally the same as the corresponding commands in 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
```

The password and enable password for all Quagga deamons (ripd and ospfd) is set to ‘mvkbj1n‘.

After this brief tutorial, you can now complete the configuration of RIP on the Linux PCs.

Exercise 2. Configuring RIP on Linux PCs with Quagga

Enable RIP on all Linux PCs. Since all Linux PCs are running as hosts, RIP is set to passive mode, where the PCs receive and process incoming RIP packets, but do not transmit RIP packets. The following guidelines describe the configuration of PC1. Repeat the steps on each PC.

1. On PC1, start the `zebra` and the`ripd` daemons by typing

```
PC1% /etc/init.d/quagga start
```

Make sure your `daemons` configuration file is correctly configured.

2. To configure the RIP routing process on PC1, connect to the `ripd` process via Telnet.

```
PC1% telnet localhost 2602
```

The system will prompt you for a login password. The password should be the same password as the login password on the Cisco routers.

3. The Linux PCs, which are configured as hosts, will be set to run RIP in passive mode. The commands to enable RIP in passive mode are as follows:

```
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
```

The `show ip rip` displays the routing database of the RIP protocol. This command does not exist in IOS. It may take a few minutes until RIP has built up its routing database. When the routing table has stabilized, that is, the results of the command `show ip rip` do not change after subsequent rounds of update messages, save the output of the command, and exit the Telnet session with the command.

```
ripd# exit
```

4. On PC1, view the routing table with the command

```
PC1% netstat -rn
```

and save the output to a file. Compare the output of `netstat -rn` to the output of `show ip rip`. Note the cost metric for each entry.

5. Repeat Steps 1-5 for the other three Linux PCs.

6. Once you can successfully issue a ping from each Linux PCs to every other Linux PC, display the route from PC1 to PC4 (10.0.4.10) with the `traceroute` command and save the result to a file:

```
PC1% traceroute 10.0.4.10
```

7. Start to capture traffic with Wireshark on all four Linux PCs. Set a capture filter or display filter to display only RIP packets.

8. Stop the traffic Wireshark capture on the PCs and save the traces for your report to a pcap file. Save the content of those RIP messages, needed to answer the questions in Part 8 (Select the Print details option).

Question 2.1)

Use the captured data of a single RIP packet and explain the fields in a RIP message.
From frame 4 in /Lab 4/traces/2-8.PC2.out:

```
Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    IP Address: 10.0.3.0, Metric: 1
        Address Family: IP (2)
        Route Tag: 0
        IP Address: 10.0.3.0 (10.0.3.0)
        Netmask: 255.255.255.0 (255.255.255.0)
        Next Hop: 0.0.0.0 (0.0.0.0)
        Metric: 1
    IP Address: 10.0.4.0, Metric: 2
        Address Family: IP (2)
        Route Tag: 0
        IP Address: 10.0.4.0 (10.0.4.0)
        Netmask: 255.255.255.0 (255.255.255.0)
        Next Hop: 0.0.0.0 (0.0.0.0)
        Metric: 2
```

We notice here that in this packet there is information for two entries in the routing table.

Explanations:

```
Routing Information Protocol
    Command: RIP command (request or response)
    Version: RIP version
    IP Address: IP address of a host to update the routing table, Metric: amount of
        internetwork hops to go through to get to the destination
        Address Family: the used address family. RIP can carry routing information
            for several different protocols (IP in this case).
        Route Tag: used to distinguish between internal and external routes
        IP Address: the IP address for the table entry
        Netmask: subnet mask for the table entry
        Next Hop: IP address of the next hop to which packets for the entry should
            be forwarded
        Metric: amount of internetwork hops to go through to get to the destination
```

Question 2.2)

For PC1, include the output of the commands `show ip rip` and `netstat -rn` from Steps 4 and 5. Discuss the differences in the output of the commands.

Step 4. Compare the output of "netstat -rn" to the output of "show ip rip".

PC1:

show ip rip:

```
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag Time
C(i) 10.0.1.0/24        0.0.0.0               1 self              0
R(n) 10.0.2.0/24        10.0.1.1              2 10.0.1.1          0 02:46
R(n) 10.0.3.0/24        10.0.1.1              3 10.0.1.1          0 02:46
R(n) 10.0.4.0/24        10.0.1.1              4 10.0.1.1          0 02:46
```

netstat -rn:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
```

```
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.2.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.3.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
```

We see that "show ip rip" shows more information related to RIP, while "netstat -rn" provides a general overview of the routing table. Information specific to RIP, such as Metric, From, Tag, Time is not shown in "netstat -rn". ** nog wat ?? **

Step 5. Repeat Steps 1-5 for the other three Linux PCs.

PC2:

show ip rip:

```
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network         Next Hop         Metric From            Tag Time
R(n) 10.0.1.0/24     10.0.2.1              2 10.0.2.1         0 02:37
C(i) 10.0.2.0/24     0.0.0.0               1 self             0
R(n) 10.0.3.0/24     10.0.2.2              2 10.0.2.2         0 03:00
R(n) 10.0.4.0/24     10.0.2.2              3 10.0.2.2         0 03:00
```

netstat -rn:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        10.0.2.1        255.255.255.0   UG        0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.3.0        10.0.2.2        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        10.0.2.2        255.255.255.0   UG        0 0          0 eth0
```

PC3:

show ip rip:

```
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network         Next Hop         Metric From            Tag Time
R(n) 10.0.1.0/24     10.0.3.2              3 10.0.3.2         0 02:41
R(n) 10.0.2.0/24     10.0.3.2              2 10.0.3.2         0 02:41
C(i) 10.0.3.0/24     0.0.0.0               1 self             0
R(n) 10.0.4.0/24     10.0.3.3              2 10.0.3.3         0 02:40
```

netstat -rn:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        10.0.3.2        255.255.255.0   UG        0 0          0 eth0
10.0.2.0        10.0.3.2        255.255.255.0   UG        0 0          0 eth0
10.0.3.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.4.0        10.0.3.3        255.255.255.0   UG        0 0          0 eth0
```

PC4:
show ip rip:

```
** missing **
```

netstat -rn:

```
** missing **
```

Question 2.3)
Include the output of `traceroute` from Step 7.

```
student@lab2pc1:~$ traceroute 10.0.4.10
traceroute to 10.0.4.10 (10.0.4.10), 30 hops max, 60 byte packets
 1  10.0.1.1 (10.0.1.1)  2.078 ms  2.563 ms  2.944 ms
 2  10.0.2.2 (10.0.2.2)  2.941 ms  4.512 ms  5.326 ms
 3  10.0.3.3 (10.0.3.3)  3.990 ms  4.817 ms  5.311 ms
 4  10.0.4.10 (10.0.4.10)  3.432 ms  3.671 ms  3.678 ms
```

Question 2.4.a)
What is the destination IP address of RIP packets?
We notice that in each "/Lab 4/traces/2-8.PC*.pcap" file, there is a RIPv2 Request from a PC to destination 224.0.0.9, followed by a RIPv2 Response from each router directly connected to the same PC with destination the IP address of that PC.
Afterwards, all destinations for RIPv2 packets have destination 224.0.0.9

Question 2.4.b)
Do routers forward RIP packets? In other words, does PC1 receive RIP packets sent by Router3?
No, we see that in "/Lab 4/traces/2-8.PC*.pcap" each PC only gets RIP packets from routers they are directly connected to.

Question 2.4.c)
Which types of routing RIP messages do you observe? The type of a RIP message is indicated by the value of the field command. For each packet type that you observed, explain the role that this message type plays in the RIP protocol.
We observed Request and Response messages.
Request: A RIP request packet is used to find routing information. A Request message sent by a node will trigger the sending of Response messages to the node by its neighbours. This message is usually sent on initialization of such a node.

Response: A RIP response packet contains routing information. Nodes which receive these packets can update their routing table if possible. Response messages can carry a whole routing table, or only entries queried for by a preceding Request message.

After a request message has been sent, response messages will be gratuitously sent then an update timer expires.

Question 2.4.d)
A RIP message may contain multiple routing table entries. How many bytes are consumed in a RIP message for each routing table entry? Which information is transmitted for each message?
20 bytes are consumed per routing table entry.

The information that is sent: IP Address, Metric, Address Family, Route Tag, IP Address, Netmask, Next Hop, Metric

Part 3. Reconfiguring the topology in RIP

In Part 3, you add Router4 to the network topology of Figure 4.1. The configuration of the network with Router4 is illustrated in Figure 4.3. The IP configuration of Router4 is given in Table 4.3. The purpose of this exercise is to explore how RIP detects changes to the network topology, and how long it takes until RIP updates the routing tables.

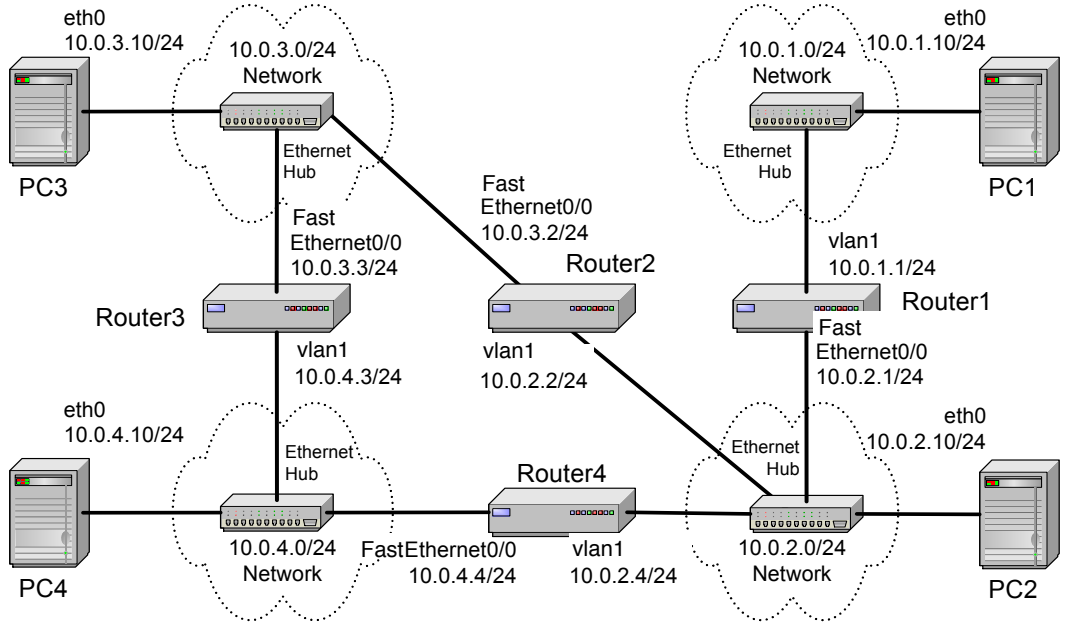

Figure 4.3: Network configuration for Part 3.

Cisco Router	FastEthernet0/0	vlan1
Router4	10.0.4.4/24	10.0.2.4/24

Table 4.3: IP addresses of Router4

Exercise 3-A. Updating the routing tables

Add Router4 to the network and observe the routing table updates made by RIP to reflect the new topology.

1. Continue with the network configuration of Part 2. RIP must be enabled on all Routers shown in Figure 4.1, and a RIP process must be running (in passive mode) on all Linux PCs.
2. Before attaching Router4, save the routing tables on all four Linux PCs with the command `netstat -rn`.
3. Connect Router4 as shown in Figure 4.3 and assign the IP addresses to the interfaces as shown in Table 4.3.
4. Configure Router4 to run RIP, following the same steps as in Part 1.

5. Use the command `netstat -rn` on the Linux PCs to observe how the routing tables are updated. Once the routing tables on the PCs have converged, save the routing tables on all four Linux PCs.

Question 3.A)

Include the routing tables of the Linux PCs before the topology was changed (Step 2) and after Router4 has been added and the routing tables have been updated (Step 5). Discuss the time it took to update the routing tables.

It took a couple of seconds for the routing tables to update. This happened quickly because Router4 sends a Request message when it connects and RIP is enabled.

PC1:

before:

```
student@lab2pc1:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.2.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.3.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
```

after:

```
student@lab2pc1:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.2.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.3.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        10.0.1.1        255.255.255.0   UG        0 0          0 eth0
```

PC2:

before:

```
student@lab2pc1:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        10.0.2.1        255.255.255.0   UG        0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.3.0        10.0.2.2        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        10.0.2.2        255.255.255.0   UG        0 0          0 eth0
```

after:

```
student@lab2pc1:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        10.0.2.1        255.255.255.0   UG        0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.3.0        10.0.2.2        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        10.0.2.4        255.255.255.0   UG        0 0          0 eth0
```

PC3:

before:

```
** missing **
```

after:

```
** missing **
```

PC4:
before:

```
student@lab2pc1:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        10.0.4.3        255.255.255.0   UG        0 0          0 eth0
10.0.2.0        10.0.4.3        255.255.255.0   UG        0 0          0 eth0
10.0.3.0        10.0.4.3        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

after:

```
student@lab2pc1:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.1.0        10.0.4.4        255.255.255.0   UG        0 0          0 eth0
10.0.2.0        10.0.4.4        255.255.255.0   UG        0 0          0 eth0
10.0.3.0        10.0.4.3        255.255.255.0   UG        0 0          0 eth0
10.0.4.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

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

Next you disconnect the Ethernet cable of interface Ethernet0/0 on Router4 and observe how much time RIP takes to update the routing table of the Linux PCs to reflect the new topology.

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

   ```
   PC4% ping 10.0.1.10
   ```

2. Disconnect the Ethernet cable connected to interface *FastEthernet0/0* on Router4. Now, the output of ping on PC4 should show that the destination network is unreachable.

3. Wait until the ping command is successful again, that is, ICMP Echo Reply messages arrive at 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.

4. Stop the ping command with `Ctrl-c` and save the ping statistics output (i.e. the data that appears at the bottom of the terminal screen when you stop the ping process).

5. Count the number of lost packets and calculate the time it took RIP to update the routing tables. (The `ping` command issues an ICMP Echo Request message approximately once every second.)

Question 3.B)

Include your answer on the convergence time from Step 4. Count the number of lost packets and calculate the time it took RIP to update the routing tables. (The ping command issues an ICMP Echo Request message approximately once every second.)

```
1 --- 10.0.1.10 ping statistics ---
  189 packets transmitted, 23 received, +126 errors, 87% packet loss, time 188036ms
3 rtt min/avg/max/mdev = 1.707/2.136/3.152/0.396 ms, pipe 4
```

With an 'inter-ping' time of 1 second, and 126 errors, we come to an update time of approximately 2 minutes.

Part 4. 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 in messages that are called link state advertisements (LSAs). LSAs are disseminated using flooding, that is, a router sends its LSAs to all its neighbours, which, in turn, forward the LSAs to their neighbours, and so on. However, each LSA is forwarded only once. Each router maintains a link state database of all received LSAs, which provides the router with complete information about the topology of the network. Routers use their link state databases to run a shortest path algorithm that computes the shortest paths in the network.

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

OSPF is the most important 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). The network configuration is shown in Figure 4.4 and Table 4.4. Note that some Linux PCs and routers are connected with crossover cables.

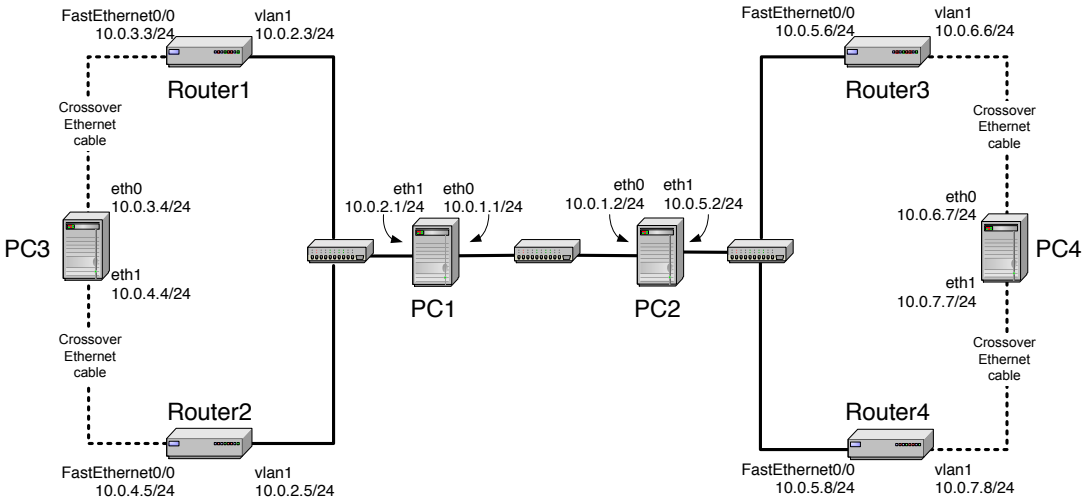

Figure 4.4: Network configuration for Part 4.

Linux PC	eth0	eth1
PC1	10.0.1.1/24	10.0.2.1/24
PC2	10.0.1.2/24	10.0.5.2/24
PC3	10.0.3.4/24	10.0.4.4/24
PC4	10.0.6.7/24	10.0.7.7/24
Cisco Router	**FastEthernet0/0**	**vlan1**
Router1	10.0.3.3/24	10.0.2.3/24
Router2	10.0.4.5/24	10.0.2.5/24
Router3	10.0.5.6/24	10.0.6.6/24
Router4	10.0.5.8/24	10.0.7.8/24

Table 4.4: IP addresses for Part 5

Exercise 4-A. 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.

1. Connect the routers as shown in Figure 4.4. Some of the interfaces are connected with crossover cables or with hubs in between them.
2. Configure the Cisco routers to run OSPF. The following set of commands are used to configure Router1.

```
Router1> enable
Password: <enable secret>
Router1# configure terminal
Router1(config)# no ip routing
Router1(config)# ip routing
Router1(config)# no router rip
Router1(config)# router ospf 1
Router1(config-router)# network 10.0.0.0 0.255.255.255 area 1
Router1(config-router)# interface FastEthernet0/0
Router1(config-if)# ip address 10.0.3.3 255.255.255.0
Router1(config-if)# interface vlan1
Router1(config-if)# ip address 10.0.2.3 255.255.255.0
Router1(config-if)# end
Router1# clear ip route *
```

   The above commands disable RIP, enable OSPF for Area 1 and network 10.0.0.0/8, and configure the IP addresses of the routers. Since no router-id is specified, the highest IP address of Router1, 10.0.3.3, is used as the router-id. The router-id can be verified by issuing the command `show ip ospf`.
3. Repeat the configuration on the other routers. Refer to Figure 4.4 for the connections, and to Table 4.4 for the IP addresses.

Exercise 4-B. Configuring OSPF on Linux PCs

On the Linux PCs, OSPF is configured using the Quagga package. The syntax of the Quagga commands is essentially identical to the corresponding IOS commands. All PCs are set up as IP routers. The following describes the configuration of PC1.

1. Connect PC1 as shown in Figure 4.4.
2. Enable IP forwarding on PC1 by typing

```
PC1% echo "1" > /proc/sys/net/ipv4/ip_forward
```

3. Terminate the existing ripd process and disable the `ripd` daemon in the `daemons` configuration file:

```
PC1% /etc/init.d/quagga stop
```

4. Disabel the `ripd` and enable the `ospfd` daemon in the `daemons` configuration file:

```
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no
```

5. Restart Quagga

```
PC1% /etc/init.d/quagga start
```

6. Set the OSPF configuration on PC1. Note that the commands for configuring OSPF in Quagga are very similar to the IOS commands:

```
PC1% telnet localhost 2604 Password: <login password>
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 10.0.1.1
ospfd(config-router)# no passive-interface eth0
ospfd(config-router)# no passive-interface eth1
ospfd(config-router)# end
ospfd# exit
```

   Note that 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 latter is necessary since Quagga does not assign a default value for the router-id. In Quagga, the router-id must be explicitly set. In this exercise we use the IP address of the Ethernet interface *eth0* as the router-id for the Linux PCs.

7. Repeat the OSPF configuration in Steps 1-6 for all other Linux PCs.

8. When the OSPF configuration is complete, all hosts and routers should be able to communicate with each other. You can test the network configuration by running `traceroute` and `ping` commands on a Linux PC (or trace and ping commands on a Cisco router). When you have verified that the network connection is correct, proceed with the next step.

Exercise 4-C. 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.

1. On PC1, start to capture traffic with Wireshark on interface *eth0*. Set a filter to only display OSPF packets.

2. From PC3, run a `traceroute` command to PC4

```
PC3% traceroute 10.0.7.7
```

   Confirm from the output and Figure 4.4, whether the path from PC3 to PC4 includes Router 3 or Router4.

3. Issue a `ping` command from PC3 to PC4 (10.0.7.7). Do not terminate the `ping` command until this exercise is completed.

```
PC3% ping 10.0.7.7
```

4. If the path from PC3 to IP address 10.0.7.7 from Step 2 included Router3, then disconnect the Ethernet cable of the *Ethernet0/1* interface of Router3. Otherwise, disconnect the Ethernet cable of the *Ethernet0/1* interface of Router4. When the Ethernet cable is disconnected, the `ping` command on PC3 will show that IP address 10.0.7.7 is not reachable.

5. Now, OSPF updates the routing tables. Use the Wireshark window on PC1 to observe the transmitted OSPF messages:

Question 4.C.1.a)
How quickly are OSPF messages sent after the cable is disconnected?

Question 4.C.1.b)
How many OSPF messages are sent?

Question 4.C.1.c)
Which type of OSPF packet is used for flooding link state information?

Question 4.C.1.d)
Describe the flooding of LSAs to all routers.

Question 4.C.1.e)
Which type of encapsulation is used for OSPF packets (TCP, UDP or other)?

Question 4.C.1.f)
What is the destination address of OSPF packets?

6. Wait until the `ping` command is successful again, that is, ICMP Echo Reply messages arrive at PC3. This happens when the routing tables have been updated.

7. Stop the ping command with `Ctrl-c` and save the ping statistics output (i.e. the data that appears at the bottom of the terminal screen when you stop the ping process).

Question 4.C.2)
Include your answer on the convergence time from Step 7. Count the number of lost packets and calculate the time it took OSPF to update the routing tables. (The ping command issues an ICMP Echo Request message approximately once every second.)

8. Issue another `traceroute` command from PC3 to IP address 10.0.7.7. By now, the output should show the new route to PC4.

9. Save the link state database on all Cisco routers and on all Linux PCs, and verify that all routers indeed have the same link state database. On the Linux PCs, open a Telnet session to the ospfd process, and then type

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

On the Cisco routers, simply type

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

Save the output of the link state databases to a file.

Question 4.C.3)
Can you confirm that the link state databases are identical? Compare the output of the command `show ip ospf database` from the Cisco routers and the Linux PCs.

10. Stop Wireshark on PC1, and save the different types of OSPF packets captured by Wireshark. Save one copy of each type of OSPF packet that you observed (Selecting the Print Detail option).

Question 4.C.4)
From your saved Wireshark output, include one packet from each of the different OSPF packet types that you have observed. (Include only one packet from each type!)

Question 4.C.5)
Include the output of the link state database of PC2.

Question 4.C.6)
Pick a single link state advertisement packet captured by Wireshark, and describe how to interpret the information contained in the link state advertisement.

Part 5. Hierarchical Routing in OSPF

The concept of areas in OSPF can be used to construct a hierarchical routing scheme. When the network is partitioned into multiple areas, then routers must have complete topology information only about routers in the same area, and only limited information about other areas. All areas must be connected to Area 0, which is a special area, called the backbone area. This builds a two-level hierarchy: The backbone area is at the top of the hierarchy and the other areas are at the bottom of the hierarchy. Traffic between two areas is routed through the backbone area. Routers that connect to two areas are called area border routers.

The configuration in this part is shown in Figure 4.5. Here, the network from Part 4 is partitioned into three areas. The area in the middle is the backbone area (Area 0). The IP addresses are the same as in Part 4, and need not be modified. PC1 and PC2 are area border routers.

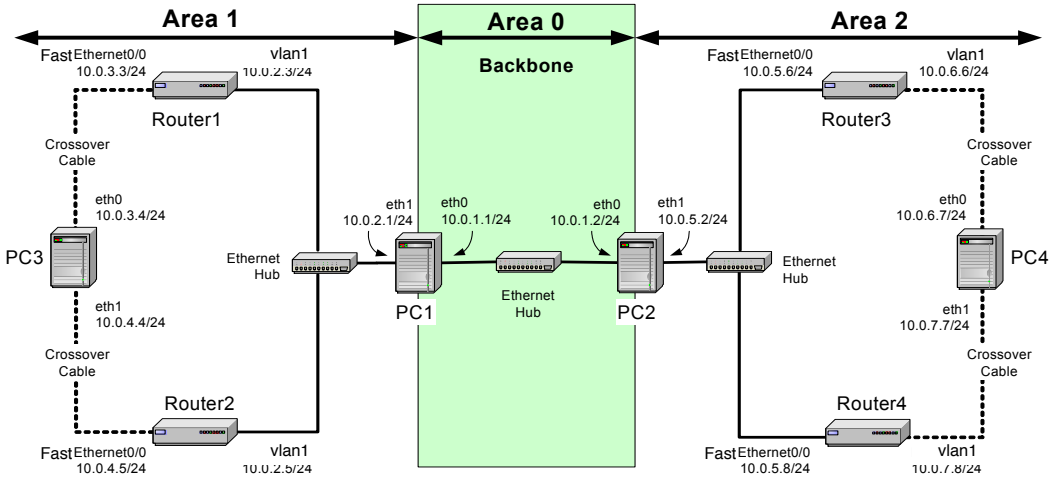

Figure 4.5: Network configuration for Part 5

In the following exercises you define the areas, and then observe how the link state databases are built.

Exercise 5. Defining multiple areas in OSPF

1. Restart the `zebra` and `ospfd` processes on all four Linux PCs. Use the same `daemons` configuration file as used in the previous exercise.

   ```
   PC1% /etc/init.d/quagga restart
   ```

2. Start Wireshark on PC1 and capture traffic on interface *eth0*.

3. Change the Area IDs of the Cisco routers and the PCs. On each system, the directly connected networks are assigned to an area with a 24-bit prefix. Here are the configurations for PC3, PC1, and Router 1. The other configurations are similar. PC3, which belongs to only one area, is configured as follows:

   ```
   PC3% telnet localhost 2604 Password: <login password>
   ospfd> enable
   ```

```
ospfd# configure terminal
ospfd(config)# router ospf
ospfd(config-router)# router-id 10.0.3.4
ospfd(config-router)# network 10.0.3.0/24 area 1
ospfd(config-router)# network 10.0.4.0/24 area 1
ospfd(config-router)# end
ospfd# exit
```

PC1, belongs to two areas, and is configured as follows:

```
PC1% telnet localhost 2604 Password: <login password>
ospfd> enable
ospfd# configure terminal
ospfd(config)# router ospf
ospfd(config-router)# router-id 10.0.1.1
ospfd(config-router)# network 10.0.2.0/24 area 1
ospfd(config-router)# network 10.0.1.0/24 area 0
ospfd(config-router)# end
ospfd# exit
```

The configuration of Router 1 is as follows:

```
Router1# configure terminal
Router1(config)# no router ospf 1
Router1(config)# router ospf 1
Router1(config-router)# network 10.0.3.0 0.0.0.255 area 1
Router1(config-router)# network 10.0.2.0 0.0.0.255 area 1
Router1(config-router)# end
Router1# clear ip ospf 1 process
```

4. Once the routing tables have converged, test the network configuration with the commands `traceroute` and `ping` on the Linux PCs, and the commands trace and ping on the Cisco routers. All hosts and routers should be able to communicate with each other.

5. Save the link state database on all Cisco routers and on all Linux PCs. On the Linux PCs, open a Telnet session to the ospfd process, and then type

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

   On the Cisco routers, type

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

   Save the output of the link state databases to a file.

Question 5.1.a)
Refer to the saved link state databases in your answers. Compare the link state databases to those saved in Part 4. Which differences do you note?

Question 5.1.b)
Which information do routers in Area 1 have about Area 2? Which information do they have about the backbone area (Area 0)?

Question 5.1.c)
How much information do the routers in the backbone area (Area 0) have about the topology of Area 1 and Area 2?

Question 5.1.d)
How do the IP routers in Area 1 know how to forward traffic to Area 2?

6. Display the area routers known to Router 1 from Area 1, with the command

   ```
   Router1# show ip ospf border-routers
   ```

   Save the output to a file.

7. Save the Wireshark output of OSPF packet types (selecting the Print Detail option) that you did not observe in Part 4. Only include one packet of each type.

Question 5.2)
Include the Wireshark output in your report showing, if any, the different types of OSPF packets that you did not observe in Part 5.

Question 5.3)
Include the output of the link state databases saved in Step 5.

Question 5.4)
Explain the output of the command “show ip ospf border-routers” in Step 6.