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
Stan Draulans Sam Mylle Federico Quin Group 3

March 11, 2017

Lab 3

Static Routing

What you will learn in this lab:

- How to turn a computer with multiple interfaces into a router
- How to set up static routing on Linux PC-routers and Cisco commercial routers
- How ICMP messages update routing table entries
- How Proxy ARP helps to connect different networks without reconfiguring the hosts
- How to work with different network prefixes

Prelab 3

Network Commands in Linux

Read the manual pages of the following commands at http://manpages.ubuntu.com/ for the operating system version “trusty 14.04 LTS”:

- `route`
- `traceroute`
- `minicom`: This lab uses the `minicom` utility program to establish a serial connection between a Linux PC and a Cisco router.

Proxy ARP

Go to the website of Cisco at http://goo.gl/ixuktT and read about Proxy ARP.

Cisco IOS

The Cisco routers in the Lab are running a recent version of the Cisco Internet Operating System (IOS). Read about the IOS at http://goo.gl/UD23vX. Note that this is reference material that you can use. You are not expected to go through all of the manuals listed here!


The most useful manuals for this course are the “IP Application Services Configuration Guide” and “Cisco IOS IP Switching Configuration Guide”.

Prelab Questions

Question 1)

What is the IOS command to change the MTU (maximum transmission unit) for an interface on a Cisco router?

In the interface config mode (config-if) reached by using the command interface *<interface> <...>*, *mtu <mtu_size>* configures the size of the maximum transmission unit for that interface.

Question 2)

How does a router determine whether a datagrams to particular host can be directly delivered through one of its interfaces?

The datagrams come in the form of IP-packets. These IP-packets contain the destination address, which the router will compare to its routing table. The longest match will be used to determine on which interface it can place the datagram.

Question 3)

Which systems generate ICMP Route Redirect messages? Routers, hosts, or both?

ICMP redirect messages are sent by routers to notify hosts on the data link a better route is available for a destination.

Question 4)

What is the default maximum TTL value used by traceroute when sending UDP datagrams?

Traceroute has a default maximum TTL of 30. This should be enough to reach most hosts, but if necessary this can be increased by the *-m <max TTL>* flag.

Question 5)

Describe the role of a default gateway in a routing table?

If no match is found in the routing table, the packets are routed to the default gateway as a "last resort": maybe another router will know how to route said packet.

Question 6)

What is the network prefix of IP address 192.110.50.3/24?

$192.110.50.0/24$

Question 7)

Explain the difference between an IP address and a network prefix.

An IP address is an address which should identify a host. A network prefix is the part of the address which identifies the network, on which multiple hosts can exist.

Question 8)

An organization has been assigned the network number 140.25.0.0/16 and it needs to create networks that support up to 60 hosts on each IP network. What is the maximum number of networks that can be set up? Explain your answer.

In order to support 60 hosts, one would need a network prefix small enough so the host part of the address can identify up to 60 hosts. As 64 is the next power of 2, using up $log_2(64) = 6$ bits, we have $32 - 16 - 6 = 10$ bits left to create networks. As $2^{10} = 1024$, we can therefore create 1024 individual networks, each with a maximum of 64 hosts.

Lab 3

In this lab you work with four different network topologies. The topology for Parts 1-4 is shown in Figure 3.1. These parts address router configuration on a Linux PC and a Cisco Router. The topology for Part 5 is shown in Figure 3.4. This topology is used to study the role of ICMP route redirect message. For Part 6 we add one more router to the topology of Part 5 and examine the effect of routing loops as displayed in Figure 3.5. The topology for Part 7 is shown in Figure 3.6. There, you explore the relationship between network prefixes and IP forwarding.

Configuring a Linux PC as a Router

Any Linux PC with at least two network interfaces can be set up as an IP router. Configuring a Linux PC as an IP router involves two steps: (1) modifying the configuration of Linux, so that IP forwarding is enabled and (2) configuring the routing table . Figure 3.1 shows the network topology used in Parts 1 - 4 of this lab. PC1 and PC4 are used as hosts, and PC2 and Router1 are set up as IP routers. The PCs and the Cisco router are connected by three Ethernet hubs. In Lab 3, all routing table entries are manually configured, a procedure known as static routing.

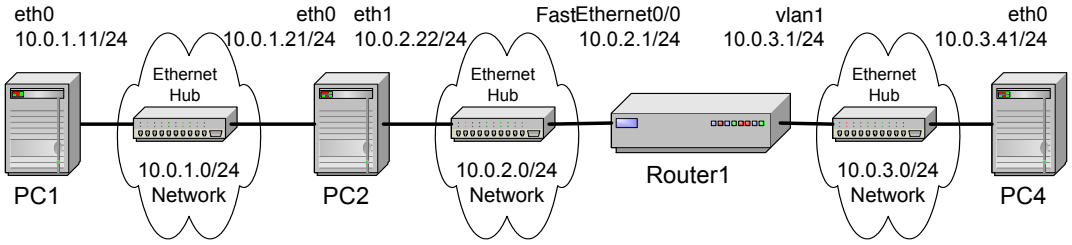

Figure 3.1: Network configuration for Parts 1-4

Linux PC	eth0	eth1
PC1	10.0.1.11/24	Disabled
PC2	10.0.1.21/24	10.0.2.22/24
PC3	10.0.3.41/24	Disabled
Cisco Router	**FastEthernet0/0**	**vlan1**
Router1	10.0.2.1/24	10.0.3.1/24

Table 3.1: IP addresses for Parts 1-4

Exercise 1-A. Network setup

1. Connect the Ethernet interfaces of the Linux PCs and the Cisco router as shown in Figure 3.1. Configure the IP addresses of the interfaces as given in Table 3.1.
2. Start to capture traffic on PC1 with Wireshark.
3. Issue a `ping` command from PC1 to PC2, Router1 and PC4. Save the output of each `ping` command.

```
C1% ping -c 5 10.0.1.21
C1% ping -c 5 10.0.2.1
C1% ping -c 5 10.0.3.41
```

4. Save the captured wireshark output.

Use the saved data to answer the following questions:

Question 1.A.1)

What is the output on PC1 when the `ping` commands are issued?

From PC1 to PC2, the ping is normal: a request followed by a reply. For the other pings, the network is unreachable.

Question 1.A.2)
Which packets, if any, are captured by Wireshark?
For the ping from PC1 to PC2, the ICMP and ARP packets are captured by Wireshark. For the other pings no packets were captured.

Question 1.A.3)
Do you observe any ARP or ICMP packets? If so, what do they indicate?
The first ping is the happy-day scenario: everything works fine. For the other two pings, no ARP or ICMP messages are sent because PC1 doesn't know how to route the packets to PC4 and the router.

Question 1.A.4)
Which destinations are not reachable? Explain.
PC4 and Router1 are not reachable from PC1 because the routing table on PC1 has no entries concerning the routing from PC1 to these devices.

Exercise 1-b. Configuring a Linux PC as a router

On a Linux system, IP forwarding is enabled when the file `/proc/sys/net/ipv4/ip_forward` contains a 1 and disabled when it contains a 0. Hence, enabling IP forwarding is done by writing a 1 in the file, with the command

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

The command echo writes the given argument, here, the string "1" to the standard output. Using the redirect operator (>) and a file name, the output of the command is written to a file. IP forwarding is disabled with the command

```
PC1% echo "0" > /proc/sys/net/ipv4/ip_forward
```

The command has an immediate effect. However, changes are not permanent and are lost when the system is rebooted. Modifying the IP forwarding state permanently requires changes to the configuration file `/etc/sysctl.conf`. IP forwarding is enabled if the file contains a line `net.ipv4.ip_forward = 1`, and IP forwarding is disabled when the line does not exist or the file contains the line `net.ipv4.ip_forward = 0`. Changes to the configuration file `/etc/sysctl.conf` take effect the next time when Linux is rebooted.

Enable PC2 as an IP router using the command:

```
PC2% echo "1" > /proc/sys/net/ipv4/ip_forward
```

Exercise 1-c. Setting static routing table entries for a Linux PC

Next, you must set up the routing tables of the Linux PCs. PC1 and PC4 are hosts, and PC2 is an IP router. The routing tables are configured so that they conform to the network topology shown in Figure 3.1 and Table 3.1. The routes are configured manually, which is also referred to as static routing.

Configuring static routes in Linux is done with the command `route`, which has numerous options for viewing, adding, deleting or modifying routing entries. The various uses of the `route` command are summarized below.

- Add a routing table entry for the network prefix identified by IP address netaddress and netmask mask. The next hop is identified by IP address gw_address or by interface *iface*.

```
route add -net netaddress netmask mask gw gw_address
route add -net netaddress netmask mask dev iface
```

- Add a host route entry for IP address hostaddress with next hop identified by IP address gw_address or by interface *iface*.

```
route add -host hostaddress gw gw_address
route add -host hostaddress dev iface
```

- Set the default route to IP address gw_address.

```
route add default gw gw_address
```

- Delete an existing route from the routing table. It is not necessary to type all arguments. If enough arguments are provided so that it can be matched with an existing routing entry, the first entry that matches the given arguments is deleted.

```
route del -net netaddress netmask mask gw gw_address
route del -host hostaddress gw gw_address
route del default gw gw_address
```

- Display the current routing table with extended fields. The command is identical to the `netstat -r` command.

```
route -e
netstat -r
```

- Display the routing table cache.

```
route -C
```

The command for adding a route for the network prefix 10.21.0.0/16 with next hop address 10.11.1.4 is

```
PC1% route add -net 10.21.0.0 netmask 255.255.0.0 gw 10.11.1.4
```

The command to add a host route to IP address 10.0.2.31 with the next hop set to 10.0.1.21 is

```
PC1% route add -host 10.0.2.31 gw 10.0.1.21
```

The command to add the IP address 10.0.4.4 as the default gateways is done with the command

```
PC1% route add default gw 10.0.4.4
```

The commands to delete the entries created with the above commands are

```
PC1% route del -net 10.21.0.0 netmask 255.255.0.0 PC1%route del -host 10.0.2.31
PC1% route del default
```

There is no simple way to delete all entries in the routing table. One method to flush the routing table is to disable the interface and then enable the interface, as in

```
PC1% ifconfig eth0 down up
```

The following commands are helpful to get information on routing and to find mistakes in the routing setup:

`ping IPaddress`
Tests if IPaddress can be reached.

`traceroute IPaddress`
Displays the route to the interface IPaddress.

When the commands are issued interactively in a Linux Shell, the added entries are valid until Linux is rebooted. To make static routes permanent on Debian-based Linux distributions, the routes need to be entered in the configuration file `/etc/network/interfaces` as post-up commands.

1. Configure the routing table entries of PC1 and PC4. You can either specify a default route or you insert separate routing entries for each remote network. For this exercise, add a route for each individual remote network. As a hint, here is the configuration information for PC4:

```
PC4%route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.3.1
PC4%route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.3.1
```

2. Configure the routing table entries of the IP router PC2. (The correctness of the routing entries will be tested after Router1 has been setup.)
3. Display the routing table of PC1, PC2, and PC4 with netstat -rn and save the output.

Question 1.C.1)
Include the saved output of the routing table. Explain the entries in the routing table and discuss the values of the fields for each entry.

Routing table of PC 1:

```
1 Kernel IP routing table
  Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
3 10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
  10.0.2.0        10.0.1.21       255.255.255.0   UG        0 0          0 eth0
5 10.0.3.0        10.0.1.21       255.255.255.0   UG        0 0          0 eth0
```

traces/1–c.3.PC1.txt

Routing table of PC 2:

```
1 Kernel IP routing table
  Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
3 10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
  10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
5 10.0.3.0        10.0.2.1        255.255.255.0   UG        0 0          0 eth1
```

traces/1–c.3.PC2.txt

Routing table of PC 4:

```
1 Kernel IP routing table
  Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
3 10.0.1.0        10.0.3.1        255.255.255.0   UG        0 0          0 eth0
  10.0.2.0        10.0.3.1        255.255.255.0   UG        0 0          0 eth0
5 10.0.3.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

traces/1–c.3.PC4.txt

The routing table contains information on how to route packets from one network to the next. The destination column contains the network addresses of the destinations. The gateway column contains the address of the router which knows how to route to the corresponding destination network. The genmask column contains the netmasks for the destination networks/hosts. The Iface column contains on which interface the network is connected.

We can clearly see PC1 routing its traffic for networks $10.0.2.0/24$ and $10.0.3.0/24$ through gateway $10.0.1.21$, as this is the address of PC2 on the $10.0.1.0/24$ network. PC2 is connected to two networks with 2 interfaces: eth0 and eth1. Its directly connected to the $10.0.1.0/24$ network on interface eth0, and directly connected to the $10.0.2.0/24$ network on interface eth1. This explains the $0.0.0.0$ gateway addresses for these networks. In order for PC2 to reach the $10.0.3.0/24$ network, it has to use the Cisco router ($10.0.2.1$) as the gateway. PC4 (and the $10.0.3.0/24$ network) are hidden from the others behind the Cisco router. In order to reach the other two networks, PC4 must therefore route its packets for these networks through the Cisco router, available on the $10.0.3.0/24$ network on address $10.0.3.1$.

Configuring a Cisco Router

The setup of the Cisco router is more involved. The first step is to establish a physical connection to the router, so that configuration commands can be entered. There are different ways to connect to a Cisco router. In the Internet Lab, you will establish a serial connection to the router. This is done with a serial cable that connects the serial port of a Linux PC to the console port of a Cisco router. The next step is to run a terminal emulation program on the Linux PC. In the Internet Lab, you use the `minicom` software to access the router. Lastly, you have to type IOS (Internet Operating System) commands using the command line interface of IOS. The network setup for this part is as shown in Figure 3.1 and Table 3.1.

Exercise 2-a. Accessing a Cisco router via the console port with Minicom

Each lab is equipped with 4 cisco 1760 routers and each PC is connected through a serial cable to one of the routers, i.e., PC1 is connected to Router1, PC2 is connected to Router2, etc. You can use the `minicom` command to establish a remote terminal connection to the router. You will use Router1 and PC1 as the console.

Access the console port of Router1 from PC1 using `minicom` by typing:

```
PC1% minicom
```

If the connection is successful, you see a command prompt (User EXEC prompt) from Router1

```
Router1>
```

When you see this prompt, you can type Cisco IOS commands. If the prompt does not appear, then hit Enter key several times.

To terminate a `minicom` session, type `Ctrl-A`, then `Z` which will show a menu. Exit by typing `Q` and following the instructions.

Exercise 2-b. Switching Cisco IOS command modes

This exercise demonstrates how to log into a router and how to operate through the different Cisco IOS command modes. It is important to understand the different modes so you know where you are and what commands are accepted at any time.

1. Start a `minicom` session on PC1 which is connected to Router1 with a serial cable.
2. When PC1 is connected to the router, you see the prompt of the user EXEC mode (`Router>`). To see which commands are available in this mode, type a question mark (`?`):

   ```
   Router1> ?
   ```

3. To view and change system parameters of a Cisco router, you must enter the privileged EXEC mode, by typing:

   ```
   Router1> enable
   Password : <enable secret>
   Router1#
   ```

You need a password, the enable secret, to enter the privileged EXEC mode.

4. To modify system wide configuration parameters, you must enter the global configuration mode. This mode is entered by typing:

```
Router1# configure terminal
Router1(config)#
```

5. To make changes to a network interface, enter the interface configuration mode, with the command:

```
Router1(config)# interface FastEthernet0/0
Router1(config-if)#
```

The name of the interface is provided as an argument. Here, the network interface that is configured is *FastEthernet0/0*.

6. To return from the interface configuration to the global configuration mode, or from the global configuration mode to the privileged EXEC mode, use the exit command:

```
Router1(config-if)# exit
Router1(config)# exit
Router1#
```

The exit command takes you one step up in the command hierarchy. To directly return to the privileged EXEC mode from any configuration mode, use the end command:

```
Router1(config-if)# end Router1#
```

7. To return from the privileged EXEC mode to the user EXEC mode, type:

```
Router1# disable
Router1>
```

8. To terminate the console session from the user EXEC mode, type:

```
Router1> logout
Router1 con0 is now available Press RETURN to get started.
```

Or type `logout` or `exit` from the privileged EXEC mode:

```
Router1# exit
Router1 con0 is now available Press RETURN to get started.
```

Exercise 2-c. Configuring IP interfaces on a Cisco router

For this course we will be working with the Cisco 1760 Router, which is shown in Figure 3.2.

The Cisco 1760 router has the following interfaces.

- 1 Ethernet Port: FastEthernet0/0
- 1 Ethernet Switch Module: vlan1.

The 4 ports of the switch module have the following names *FastEthernet0/1*, *FastEthernet0/2*, *FastEthernet0/3*, *FastEthernet0/4*. Note that you can use the shorthand *FA0/X* instead of writing *FastEthernet0/X*.

The easiest way to configure the router is to:

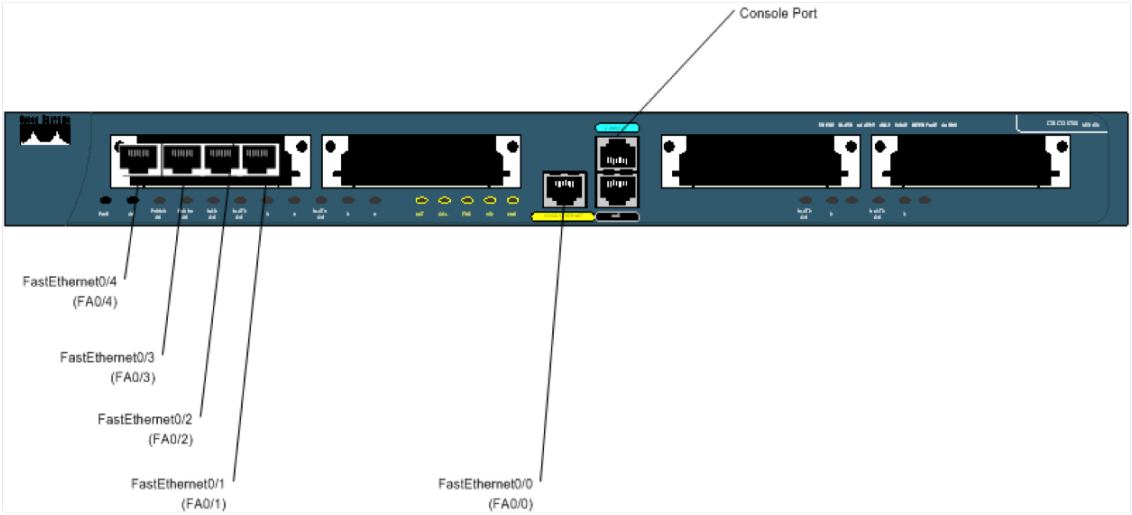

Figure 3.2: Cisco 1760

- Enable the onboard interface *FA0/0* and give it an IP address.
- Turn on one of the ports of the switch module, we recommend you to always use *FA0/1*.
- Enable the *Vlan1* interface and assign it an IP address.
- We also recommend not changing any of the VLAN settings on the switch module.

In IOS this becomes:

```
Router1(config)# interface FastEthernet0/1
Router1(config-if)# no shutdown
Router1(config-if)# interface vlan1
Router1(config-if)# ip address 10.0.2.1 255.255.255.0
Router1(config-if)# no shutdown
```

Figure 3.3 shows a logical representation of the internal operation of the Cisco1760, and how the virtual interface *Vlan1* can be configured with an IP address.

The following exercises use basic commands from the Cisco IOS that are needed to configure a Cisco router.

1. Start a minicom session on PC1 which is connected to Router1 with a serial cable.
2. Configure Router1 with the IP addresses given in Table 3.1.

```
Router1> enable
Password: <enable secret>
Router1# configure terminal
Router1(config)# no ip routing
Router1(config)# ip routing
Router1(config)# interface FastEthernet0/0
Router1(config-if)# ip address 10.0.2.1 255.255.255.0
```

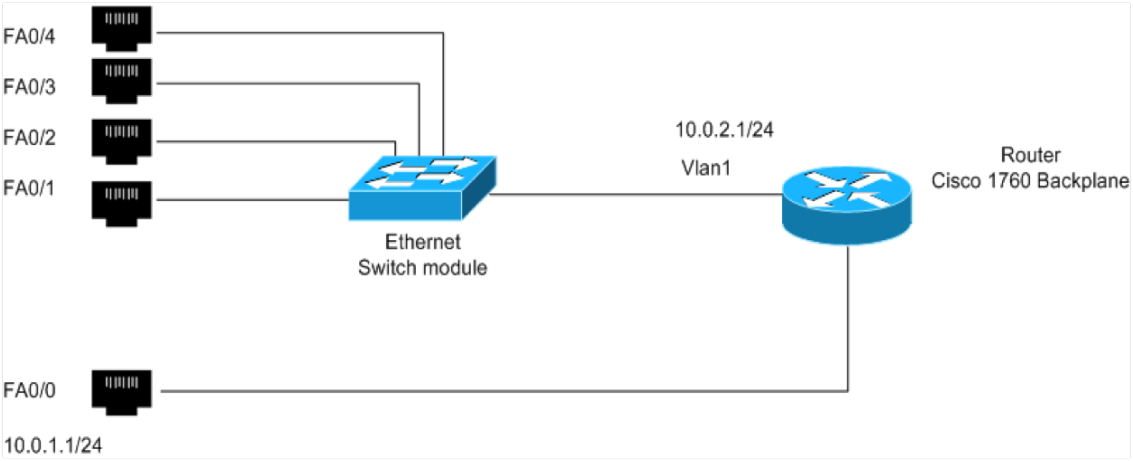

Figure 3.3: Cisco 1760 Switch Module

```
Router1(config-if)# no shutdown
Router1(config-if)# interface FastEthernet0/1
Router1(config-if)# no shutdown
Router1(config-if)# interface vlan1
Router1(config-if)# ip address 10.0.3.1 255.255.255.0
Router1(config-if)# no shutdown
Router1(config-if)# end
```

3. When you are done, use the following command to check the changes you made to the router configuration, and save the output:

```
Router1# show interfaces
Router1# show running-config
```

4. Analyze the output to ensure that you have configured the router correctly.

Question 2.C.1)

Include the output from Step 3 in your lab report.

```
FastEthernet0/0 is up, line protocol is up
  Hardware is PQUICC_FEC, address is 000e.83f5.ff4c (bia 000e.83f5.ff4c)
  Internet address is 10.0.2.1/24
  MTU 1500 bytes, BW 100000 Kbit/sec, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Half-duplex, 10Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:03:02, output 00:00:09, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     37 packets input, 3248 bytes
     Received 12 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog
```

```
     0 input packets with dribble condition detected
     449 packets output, 27586 bytes, 0 underruns
     3 output errors, 0 collisions, 1 interface resets
     0 unknown protocol drops
     0 babbles, 0 late collision, 0 deferred
     3 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
FastEthernet0/1 is up, line protocol is up
  Hardware is Fast Ethernet, address is 0012.00d4.bf30 (bia 0012.00d4.bf30)
  MTU 1500 bytes, BW 10000 Kbit/sec, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Half-duplex, 10Mb/s
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     57 packets input, 4668 bytes, 0 no buffer
     Received 28 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     1707 packets output, 109878 bytes, 0 underruns
     0 output errors, 0 collisions, 3 interface resets
     0 unknown protocol drops
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
FastEthernet0/2 is administratively down, line protocol is down
  Hardware is Fast Ethernet, address is 0012.00d4.bf31 (bia 0012.00d4.bf31)
  MTU 1500 bytes, BW 100000 Kbit/sec, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Auto-duplex, Auto-speed
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 unknown protocol drops
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
FastEthernet0/3 is administratively down, line protocol is down
  Hardware is Fast Ethernet, address is 0012.00d4.bf32 (bia 0012.00d4.bf32)
  MTU 1500 bytes, BW 100000 Kbit/sec, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Auto-duplex, Auto-speed
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
```

```
    Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
 89   Queueing strategy: fifo
    Output queue: 0/40 (size/max)
 91   5 minute input rate 0 bits/sec, 0 packets/sec
    5 minute output rate 0 bits/sec, 0 packets/sec
 93      0 packets input, 0 bytes, 0 no buffer
       Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
 95      0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
       0 input packets with dribble condition detected
 97      0 packets output, 0 bytes, 0 underruns
       0 output errors, 0 collisions, 2 interface resets
 99      0 unknown protocol drops
       0 babbles, 0 late collision, 0 deferred
101      0 lost carrier, 0 no carrier
       0 output buffer failures, 0 output buffers swapped out
103 FastEthernet0/4 is administratively down, line protocol is down
    Hardware is Fast Ethernet, address is 0012.00d4.bf33 (bia 0012.00d4.bf33)
105   MTU 1500 bytes, BW 100000 Kbit/sec, DLY 100 usec,
       reliability 255/255, txload 1/255, rxload 1/255
107   Encapsulation ARPA, loopback not set
    Keepalive set (10 sec)
109   Auto-duplex, Auto-speed
    ARP type: ARPA, ARP Timeout 04:00:00
111   Last input never, output never, output hang never
    Last clearing of "show interface" counters never
113   Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
    Queueing strategy: fifo
115   Output queue: 0/40 (size/max)
    5 minute input rate 0 bits/sec, 0 packets/sec
117   5 minute output rate 0 bits/sec, 0 packets/sec
       0 packets input, 0 bytes, 0 no buffer
119      Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
       0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
121      0 input packets with dribble condition detected
       0 packets output, 0 bytes, 0 underruns
123      0 output errors, 0 collisions, 2 interface resets
       0 unknown protocol drops
125      0 babbles, 0 late collision, 0 deferred
       0 lost carrier, 0 no carrier
127      0 output buffer failures, 0 output buffers swapped out
    Vlan1 is up, line protocol is up
129   Hardware is EtherSVI, address is 000e.83f5.ff4c (bia 000e.83f5.ff4c)
    Internet address is 10.0.3.1/24
131   MTU 1500 bytes, BW 100000 Kbit/sec, DLY 1000000 usec,
       reliability 255/255, txload 1/255, rxload 1/255
133   Encapsulation ARPA, loopback not set
    ARP type: ARPA, ARP Timeout 04:00:00
135   Last input 00:05:20, output never, output hang never
    Last clearing of "show interface" counters never
137   Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
    Queueing strategy: fifo
139   Output queue: 0/40 (size/max)
    5 minute input rate 0 bits/sec, 0 packets/sec
141   5 minute output rate 0 bits/sec, 0 packets/sec
       42 packets input, 3540 bytes, 0 no buffer
143      Received 19 broadcasts, 0 runts, 0 giants, 0 throttles
       0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
145      28 packets output, 2310 bytes, 0 underruns
       0 output errors, 3 interface resets
147      0 unknown protocol drops
       0 output buffer failures, 0 output buffers swapped out
149
151
153
```

```
155
157
159
161
163
    Building configuration...
165
    Current configuration : 842 bytes
167 !
    version 12.4
169 service timestamps debug datetime msec
    service timestamps log datetime msec
171 no service password-encryption
    !
173 hostname Router1
    !
175 boot-start-marker
    boot system flash:c1700-advipservicesk9-mz.124-25d.bin
177 boot-end-marker
    !
179 enable password mvkbj1n
    !
181 no aaa new-model
    ip cef
183 !
    !
185 !
    !
187 ip auth-proxy max-nodata-conns 3
    ip admission max-nodata-conns 3
189 !
    !
191 !
    !
193 !
    !
195 !
    !
197 !
    !
199 !
    !
201 !
    !
203 !
    !
205 !
    !
207 !
    !
209 !
    !
211 interface FastEthernet0/0
     ip address 10.0.2.1 255.255.255.0
213  speed auto
    !
215 interface FastEthernet0/1
    !
217 interface FastEthernet0/2
     shutdown
219 !
    interface FastEthernet0/3
221  shutdown
```

```
!
interface FastEthernet0/4
 shutdown
!
interface Vlan1
 ip address 10.0.3.1 255.255.255.0
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
no cdp run
!
!
!
control-plane
!
!
!
!
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
end
```

traces/2–c.3.txt

Exercise 2-d. Setting static routing table entries on a Cisco router

Next you must add static routes to the routing table of Router1. The routing table must be configured so that it conforms to the network topology shown in Figure 3.1 and Table 3.1.

The IOS command to configure static routing is `ip route`. The command can be used to show, clear, add or delete entries in the routing table. Below is a summary of the commands.

The following can be executed in the privileged EXEC mode.:

`show ip route`
: *Display the contents of the routing table.*

`clear ip route *`
: *Delete all routing table entries.*

`show ip cache`
: *Display the routing cache.*

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

`ip route-cache`
Enable route caching. By default, route caching is enabled on a router.

`no ip route-cache`
Disable route caching.

`ip route destination mask gw_address`
Add a static routing table entry to destination with netmask `mask`*. The argument* `gw_address` *is the IP address of the next hop router.*

`ip route destination mask Iface`
Add a static routing table entry to destination with netmask `mask`*. Here, the next hop information is the name of a network interface (e.g.,* FastEthernet0/0*).*

`no ip route destination mask gw_address no ip route destination mask Iface`
Delete the route table entry with `destination`*,* `mask`*, and* `gw_address` *or* `Iface` *from the routing table.*

We next show some examples for adding and deleting routing table entries in IOS. Compare these commands to the corresponding Linux commands in Part 2, Exercise 1-c. As in Linux, whenever an IP address is configured for a network interface, routing table entries for the directly connected network are added automatically.

The command for adding a route for the network prefix 10.21.0.0/16 with 10.11.1.4 as the next hop address is

```
Router1(config)#ip route 10.21.0.0 255.255.0.0 10.11.1.4
```

The command to add a host route to IP address 10.0.2.31 with the next hop set to 10.0.1.21 is

```
Router1(config)#ip route 10.0.2.31 255.255.255.255 10.0.1.21
```

In IOS,a host route is identified by a 32-bit prefix. The command to add the IP address 10.0.4.4 as the default gateways is done with the command.

```
    Router1(config) #ip route 0.0.0.0 0.0.0.0 10.0.4.4
```

Finally, commands to delete the above entries use the `no ip route` command.

```
Router1(config)# no ip route 10.21.0.0 255.255.0.0 10.11.1.4
Router1(config)# no ip route 10.0.2.31 255.255.255.255 10.0.1.21
Router1(config)# no ip route 0.0.0.0 0.0.0.0 10.0.4.4
```

1. Display the content of the routing table with `show ip route`. Note the routing entries that are already present. Save the output.
2. Add routing entries to Router1, so that the router forwards datagrams for the configuration shown in Figure 3.1. Routing entries should exist for the following networks:
   - 10.0.1.0/24

- 10.0.2.0/24
- 10.0.3.0/24

3. Display the routing table again with `show ip route` and save the output.

Question 2.D.1)

Include the saved output of the routing table from Step 1 and Step 2. Explain the fields of the routing table entries of the Cisco router. Explain how the routing table has changed from Step 1 to Step 3.

Step 1:

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 2 subnets
C       10.0.2.0 is directly connected, FastEthernet0/0
C       10.0.3.0 is directly connected, Vlan1
```

traces/2–d.1.txt

Step 2/3:

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 3 subnets
C       10.0.2.0 is directly connected, FastEthernet0/0
C       10.0.3.0 is directly connected, Vlan1
S       10.0.1.0 [1/0] via 10.0.2.22
                 is directly connected, FastEthernet0/0
```

traces/2–d.3.txt

The first part of an entry describes the type of route. In our case we have either C or S (C meaning that the route to the subnet is directly connected to the interface , S meaning that the entry is static (added by us)). The second part ($10.0.x.0$ is directly connected...) describes on which interface packets should be routed, according to the subnet.

Finalizing and Exploring the Router Configuration

If the configuration of PC2 and Router1 was done correctly, it is now possible to send IP datagrams between any two machines in the network shown in Figure 3.1. However, if the network is not configured properly, you need to debug and test your setup. The table below illustrates several common problems that may arise. Since it is impossible to cover all scenarios, network debugging is a crucial skill that you need to obtain for your lab experiments to work well.

Problem	**Possible Causes**	**Debugging**
Traffic does not reach destinations on local network	Network interface not configured correctly.	Verify the interface configuration with `show protocols` (in IOS) or `ifconfig` (in Linux)
	Incorrectly connected, faulty, or loose cables.	Most interface cards and Ethernet hubs have green LED status lights. Check if the status lights are on. Verify the connection of the cables. Verify that no cross-over cables are used.
Traffic reaches router, but is not forwarded to remote networks	IP forwarding is not enabled.	Use `show protocols` (in IOS) or look into `/proc/sys/net/ipv4/ip\_forward` (in Linux) to display the forwarding status
	Routing tables are not configured correctly.	Display routing tables with `show ip route` (in IOS) or `netstat -rn` (in Linux). Run `traceroute` between all hosts and routers.
ICMP Request messages reaches destination, but ICMP Reply does not reach source	Routing tables are not correctly configured for the reverse path.	Display routing tables with `show ip route` (in IOS) or `netstat -rn` (in Linux). Run `ping` and `traceroute` in both directions.
A change in the routing table has no effect on the flow of traffic.	The ARP cache has old entries.	Delete the ARP cache with `clear arp` (in IOS) or delete entries with `arp -d` (in Linux).

Exercise 3-A. Finalizing the network setup

Test the network configuration by issuing ping commands from each host and router to every other host and router. If some ping commands do not work, you need to modify the configuration

of routers and hosts. If all ping commands are successful, the network configuration is correct, and you can proceed to the next step.

Exercise 3-B. Testing routes with traceroute

1. Start an Wireshark session on PC1.
2. Execute a `traceroute` command from PC1 to PC4, and save the output.

   ```
   PC1% traceroute 10.0.3.41
   ```

   Observe how `traceroute` gathers information on the route.
3. Stop the traffic capture of Wireshark and save the traffic generated by the `traceroute` command.
4. Save the routing table of PC1, PC4, PC2 and Router1.

Question 3.B.1)
Use the Wireshark output and the previously saved routing table to explain the operation of `traceroute`.
Traceroute sends 3 packets for each TTL increasing from 1 to 30 or until it has reached its destination. It targets an unused port so the destination does not interpret it as a valuable data packet. Traceroute can then determine the route based on the ICMP messages generated by the TTL of the packets expiring, and it will know it has reached its destination when it receives a port unreachable message.

Exercise 3-C. Observe MAC addresses at a router

When a router forwards an IP datagram from one Ethernet segment to another, it does not modify the IP destination address. However, the destination Ethernet address in the Ethernet header is modified at a router.

This exercise requires manipulations to the ARP cache. The `arp` command in Linux was covered in Lab 2. Below are the corresponding IOS commands for Cisco routers.

The following can can be executed in the privileged EXEC mode:

`ip arp`
Display the contents of the ARP cache

`clear arp`
Delete the entire ARP cache

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

`arp IPaddress`
Add an entry for `IPaddress` to the ARP cache

`no arp IPaddress`
Delete the ARP entry for `IPaddress` from the ARP cache

1. Erase all ARP entries on PC1, PC2, PC4 and Router1.

2. Run Wireshark on both PC1 (interface *eth0*) and PC4 (interface *eth0*).

3. Issue a `ping` command on PC1 to PC4.

```
PC1% ping -c 5 10.0.3.41
```

4. Save the packet transmissions triggered by the `ping` command, including ARP requests, ARP reply, ICMP echo request, ICMP echo reply on both PC1 and PC4.

Question 3.C.1)
Determine the source and destination addresses in the Ethernet and IP headers, for the ICMP Echo Request messages that were captured at PC1.
Source: $10.0.1.11$ $(68:05:ca:1a:7c:70)$
Destination: $10.0.3.41$ $(68:05:ca:1a:7c:77)$

Question 3.C.2)
Determine the source and destination addresses in the Ethernet and IP headers, for the ICMP Echo Request message that were captured at PC4.
Source: $10.0.1.11$ $(00:0e:83:f5:ff:4c)$
Destination: $10.0.3.41$ $(68:05:ca:1a:7c:6f)$

Question 3.C.3)
Use your answers above to explain how the source and destination Ethernet and IP addresses are changed when a datagram is forwarded by a router.
The IP addresses of the packets do not change as IP addresses are necessary to identify (remote) hosts. The ethernet headers can be modified, as this works on a link-to-link basis.

Exercise 3-D. Multiple matches in the routing table

A router or host uses a routing table to determine the next hop of the path of an IP datagram. In Linux, routing table entries are sorted in the order of decreasing prefix length, and are read from top to bottom. In this exercise, you determine how an IP router or Linux PC resolves multiple matching entries in a routing table.

1. Add the following routes to the routing table of PC1:

```
PC1% route add -net 10.0.0.0 netmask 255.255.0.0 gw 10.0.1.71
PC1% route add -host 10.0.3.9 gw 10.0.1.81
```

   From Exercise 1-C there should be a network route for the network prefix 10.0.3.0/24. If there is no such route, then add the following entry:

```
PC1% route add -net 10.0.3.0 netmask 255.255.255.0 gw 10.0.1.61
```

2. Referring to the routing table, determine how many matches exist for the following IP addresses:

```
10.0.3.9
10.0.3.14
10.0.4.1
```

3. Start an Wireshark session on PC1, and issue the following ping commands from PC1:

```
PC1% ping -c 1 10.0.3.9
PC1% ping -c 1 10.0.3.14
PC1% ping -c 1 10.0.4.1
```

Note that gateways with IP addresses 10.0.1.61, 10.0.1.71, and 10.0.1.81 do not exist. However, PC1 still sends ARP Request packets for these IP addresses.

4. Save the output of Wireshark and PC1's routing table.

Question 3.D)

Use the saved output to indicate the number of matches for each of the IP addresses above. Explain how PC1 resolves multiple matches in the routing table. Only include relevant output data in your report to support your analysis of the data.
PC 1's routing table:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0        10.0.1.71       255.0.0.0       UG        0 0          0 eth0
10.0.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0
10.0.2.0        10.0.1.21       255.255.255.0   UG        0 0          0 eth0
10.0.3.0        10.0.1.21       255.255.255.0   UG        0 0          0 eth0
10.0.3.9        10.0.1.81       255.255.255.255 UGH       0 0          0 eth0
```

traces/3–D.4.PC1.txt

$10.0.3.9$ matches with 3 entries: $10.0.0.0/16$, $10.0.3.0/24$ and $10.0.3.9/32$. It will pick $10.0.3.9/32$ as it has the longest prefix match. $10.0.3.14$ matches with 2 entries: $10.0.0.0/16$and $10.0.3.0/24$. It will pick $10.0.3.0/24$ as it has the longest prefix match. $10.0.4.1$ matches with only 1 entry: $10.0.0.0/16$, and will be routed here.

Exercise 3-E. Default Routes

1. Delete the routing table entries added in Step 1 of Exercise 3-D above. (Otherwise, the entries interfere with the remaining exercises in this lab.)
2. Add default routes on PC1 and PC2.
   - On PC1, add a default route with PC2 as the default gateway.
   - On PC2, add a default route with Router1 as the default gateway.
3. Start to capture traffic on PC1 (on *eth0*) and PC2 (on both *eth0* and *eth1*) with Wireshark.
4. Issue a `ping` command from PC1 to a host on a network that does not exist.

```
PC1% ping -c 5 10.0.10.110
```

5. Save the Wireshark output.

Question 3.E.1)

What is the output on PC1, when the ping command is issued?
The output tells us that the destination host is unreachable.

Question 3.E.2)

Determine how far the ICMP Echo Request message travels?
The echo request travels as far as the Cisco router. At PC1, the request is put on the default gateway (to PC2). At PC2, the packet is put on the default gateway aswell (to the

Cisco router). The request cannot be routed further at the router, since there is no entry that corresponds to the destination address.
The requests messages were seen at PC1 and PC2 ("traces/3-E.5.PC1.pcapng" and "traces/3-E.5.PC2.pcap").
When setting the default gateway, the pcs tried checking for updates, sending traffic to ubuntu archives.

Question 3.E.3)

Which ICMP Echo Reply message returns to PC1?
A reply message saying that the host is not reachable is returned to PC1, as seen in the file "3-E.5.PC1.pcapng".

Proxy ARP

Proxy Address Resolution Protocol (Proxy ARP) is a method by which a router can forward traffic without using its routing table. Proxy ARP is a configuration option, where an IP router responds to ARP Requests that arrive from one of its connected networks for a host that is on another of its connected networks. Without Proxy ARP enabled, an ARP Request for a host on a different network is unsuccessful, since routers do not forward ARP packets to another network.

In this part, you explore how Proxy ARP enables routers to forward an IP datagram even though the sender of the datagram is not aware that the IP datagram should be forwarded to a router. Proxy ARP is enabled and disabled separately on each interface. In IOS, proxy ARP is enabled by default.

The commands to enable and disable Proxy ARP in the IOS Interface configuration mode are:

```
ip proxy-arp
no ip proxy-arp
```

Exercise 4.

1. Erase the ARP table and the routing table of PC4.
2. Set the netmask of PC4 to 255.0.0.0, so that PC4 assumes it belongs to network 10.0.0.0/8, instead of belonging to the network 10.0.3.0/24.
3. Run Wireshark on PC4 (*eth0*), PC2 (*eth1*), and PC1 (*eth0*). Set a display or capture filter to only display ICMP and ARP packets.
4. Issue a `ping` from PC4 to PC1:

   ```
   PC4% ping -c 2 10.0.1.11
   ```

   Explore the captured data and interpret the outcome. Even though PC4 had no default routing entry in its table for Router1, it was still able to connect to PC1, i.e., you should not observe a “network unreachable” error message.
5. Save the ARP table of PC4 and the packets captured by Wireshark on the hosts.
6. Explore the captured data and interpret the outcome.
7. Now, disable Proxy ARP on both interfaces of Router1. Is it still feasible to issue a ping from PC4 to PC1?
8. Reset the network mask of PC4 to its original value of 255.255.255.0. Then, re-enable Proxy ARP on Router1.

Question 4.1)

Use the captured data to explain the outcome of the exercise. Use the data to explain how Proxy ARP allowed PC4 to communicate with PC1. Include only relevant data from your saved output.

When PC4 tries to ping PC1 in the first scenario, it tries to send an ARP request to get the physical address of PC1. However, PC1 is not directly reachable by PC4. The ARP request message reaches the Cisco router, which in turn will forward the request to its other network (because proxy ARP is enabled). A response to the ARP request eventually reaches back to PC4. This can be seen by inspecting the packets at PC1 and PC4 (the ARP request PC4 sent can be seen at PC1 as well).

```
1       1 0.000000000    10.0.3.41             10.0.1.11             ICMP     98
            Echo (ping) request  id=0x0d4b, seq=1/256, ttl=62 (reply in 2)

3   Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
    Ethernet II, Src: 68:05:ca:1a:7c:77, Dst: 68:05:ca:1a:7c:70
5   Internet Protocol Version 4, Src: 10.0.3.41, Dst: 10.0.1.11
    Internet Control Message Protocol
7
        2 0.000034595    10.0.1.11             10.0.3.41             ICMP     98
            Echo (ping) reply    id=0x0d4b, seq=1/256, ttl=64 (request in 1)
9
    Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
11  Ethernet II, Src: 68:05:ca:1a:7c:70, Dst: 68:05:ca:1a:7c:77
    Internet Protocol Version 4, Src: 10.0.1.11, Dst: 10.0.3.41
13  Internet Control Message Protocol

15      3 1.000572621    10.0.3.41             10.0.1.11             ICMP     98
            Echo (ping) request  id=0x0d4b, seq=2/512, ttl=62 (reply in 4)

17  Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
    Ethernet II, Src: 68:05:ca:1a:7c:77, Dst: 68:05:ca:1a:7c:70
19  Internet Protocol Version 4, Src: 10.0.3.41, Dst: 10.0.1.11
    Internet Control Message Protocol
21
        4 1.000603230    10.0.1.11             10.0.3.41             ICMP     98
            Echo (ping) reply    id=0x0d4b, seq=2/512, ttl=64 (request in 3)
23
    Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
25  Ethernet II, Src: 68:05:ca:1a:7c:70, Dst: 68:05:ca:1a:7c:77
    Internet Protocol Version 4, Src: 10.0.1.11, Dst: 10.0.3.41
27  Internet Control Message Protocol

29      5 5.007449757    68:05:ca:1a:7c:77     68:05:ca:1a:7c:70     ARP      60
            Who has 10.0.1.11? Tell 10.0.1.21

31  Frame 5: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 68:05:ca:1a:7c:77, Dst: 68:05:ca:1a:7c:70
33  Address Resolution Protocol (request)

35      6 5.007474640    68:05:ca:1a:7c:70     68:05:ca:1a:7c:77     ARP      42
            10.0.1.11 is at 68:05:ca:1a:7c:70

37  Frame 6: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
    Ethernet II, Src: 68:05:ca:1a:7c:70, Dst: 68:05:ca:1a:7c:77
39  Address Resolution Protocol (reply)
```

traces/4.5.PC1.txt

```
1       1 0.000000000    IntelCor_1a:7c:6f     Broadcast             ARP      42
            Who has 10.0.1.11? Tell 10.0.3.41

3   Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
    Ethernet II, Src: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f), Dst: Broadcast (ff:ff:ff:
        ff:ff:ff)
5   Address Resolution Protocol (request)

7       2 0.001249403    CiscoInc_f5:ff:4c     IntelCor_1a:7c:6f     ARP      60
            10.0.1.11 is at 00:0e:83:f5:ff:4c

9   Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
    Ethernet II, Src: CiscoInc_f5:ff:4c (00:0e:83:f5:ff:4c), Dst: IntelCor_1a:7c:6f
        (68:05:ca:1a:7c:6f)
11  Address Resolution Protocol (reply)

13      3 0.001262486    10.0.3.41             10.0.1.11             ICMP     98
            Echo (ping) request  id=0x0d4b, seq=1/256, ttl=64 (reply in 4)
```

```
Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f), Dst: CiscoInc_f5:ff:4c
    (00:0e:83:f5:ff:4c)
Internet Protocol Version 4, Src: 10.0.3.41, Dst: 10.0.1.11
Internet Control Message Protocol

     4 0.002664499    10.0.1.11             10.0.3.41             ICMP     98
         Echo (ping) reply    id=0x0d4b, seq=1/256, ttl=62 (request in 3)

Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: CiscoInc_f5:ff:4c (00:0e:83:f5:ff:4c), Dst: IntelCor_1a:7c:6f
    (68:05:ca:1a:7c:6f)
Internet Protocol Version 4, Src: 10.0.1.11, Dst: 10.0.3.41
Internet Control Message Protocol

     5 1.001822475    10.0.3.41             10.0.1.11             ICMP     98
         Echo (ping) request  id=0x0d4b, seq=2/512, ttl=64 (reply in 6)

Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f), Dst: CiscoInc_f5:ff:4c
    (00:0e:83:f5:ff:4c)
Internet Protocol Version 4, Src: 10.0.3.41, Dst: 10.0.1.11
Internet Control Message Protocol

     6 1.003314843    10.0.1.11             10.0.3.41             ICMP     98
         Echo (ping) reply    id=0x0d4b, seq=2/512, ttl=62 (request in 5)

Frame 6: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: CiscoInc_f5:ff:4c (00:0e:83:f5:ff:4c), Dst: IntelCor_1a:7c:6f
    (68:05:ca:1a:7c:6f)
Internet Protocol Version 4, Src: 10.0.1.11, Dst: 10.0.3.41
Internet Control Message Protocol
```

traces/4.5.PC4.txt

PC4's ARP table also contains PC1's MAC address as a result.

```
? (10.0.1.11) at 00:0e:83:f5:ff:4c [ether] on eth0
```

traces/4.5.ARP.PC4.txt

PC4 can ping PC1 without problems after ARP is resolved (normal ICMP packets were captured over the network in all hosts, can be seen in these files: "4.5.PC1.pcapng", "4.5.PC2.pcapng", "4.5.PC1.pcap").

For the second scenario (where proxy ARP is disabled), the ping is not succesful. PC4 will once again try to send an ARP request for PC1's MAC address. The ARP request reaches the router, but the packet will not be forwarded since proxy ARP is disabled. PC4 receives no reply to the ARP request, and will say that the host is unreachable.

```
PING 10.0.1.11 (10.0.1.11) 56(84) bytes of data.
From 10.0.3.41 icmp_seq=1 Destination Host Unreachable
From 10.0.3.41 icmp_seq=2 Destination Host Unreachable

--- 10.0.1.11 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1008ms
pipe 2
```

traces/4.7.ping.PC4.txt

```
     1 0.000000000    IntelCor_1a:7c:6f     Broadcast             ARP      42
         Who has 10.0.1.11? Tell 10.0.3.41

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
```

```
Ethernet II, Src: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f), Dst: Broadcast (ff:ff:ff:
    ff:ff:ff)
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f)
    Sender IP address: 10.0.3.41
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.0.1.11

      2 0.998260631    IntelCor_1a:7c:6f    Broadcast           ARP      42
        Who has 10.0.1.11? Tell 10.0.3.41

Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f), Dst: Broadcast (ff:ff:ff:
    ff:ff:ff)
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f)
    Sender IP address: 10.0.3.41
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.0.1.11

      3 1.998273914    IntelCor_1a:7c:6f    Broadcast           ARP      42
        Who has 10.0.1.11? Tell 10.0.3.41

Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f), Dst: Broadcast (ff:ff:ff:
    ff:ff:ff)
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: IntelCor_1a:7c:6f (68:05:ca:1a:7c:6f)
    Sender IP address: 10.0.3.41
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.0.1.11
```

traces/4.7.PC4.txt

ICMP Route Redirect

ICMP route redirect messages are sent from a router to a host, when a datagram should have been forwarded to a different router or interface. In Linux, an ICMP Route Redirect message updates the routing cache, but not the routing table.

Both the routing cache and the routing table contain information for forwarding traffic. When a Linux system performs a routing table lookup, it first inspects the routing cache. If no matching entry is found in the cache, Linux performs a lookup in the routing table. After each routing table lookup, an entry is added to the routing cache. The routing cache does not aggregate table entries, and there is a separate entry for each destination IP address. As a consequence, a lookup in the routing cache does not require a longest prefix match. An entry in the routing cache is deleted if it has not been used for some time, usually after 10 minutes. When an ICMP redirect message arrives, an entry is added to the routing cache, but no update is performed to the routing table.

The following are the commands to display the contents of the routing cache:

```
route -C
```

In Linux

```
show ip cache
```

In IOS

In this part of the lab, you use three Cisco routers. Figure 3.4 and Table 3.2 describe the network configuration for the exercises below.

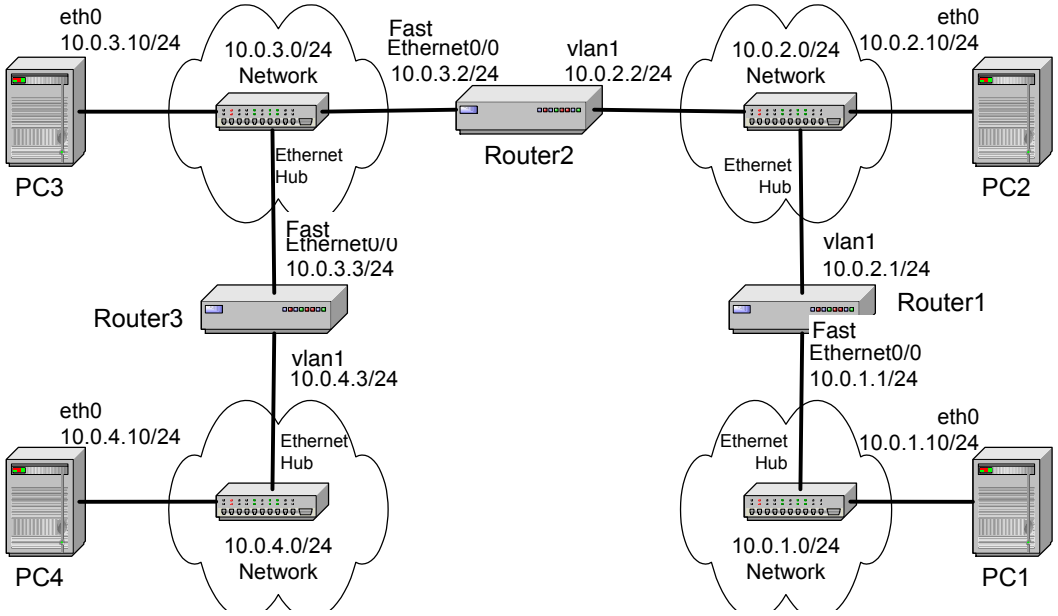

Figure 3.4: Network configuration for Part 5

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
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

Table 3.2: IP addresses for Part 5

Exercise 5.

In the network shown in Figure 3.4, when PC2 sends datagrams with destination 10.0.3.10 (PC3) to 10.0.2.1 (Router1), as opposed to 10.0.2.2 (Router2), then Router1 sends an ICMP Route Redirect message to PC2. The ICMP Route Redirect informs PC2 that it should send datagrams with destination 10.0.3.10 to Router2 instead.

In this exercise, you create the above scenario. First, you will trigger the transmission of an ICMP Route Redirect message and subsequently observe a change to the routing cache.

1. Connect the Ethernet interfaces of the routers and the hosts to the hubs as shown in Figure 3.4.
2. Delete all routing table entries and all ARP cache entries on all PCs and on Router 1.
   - Delete the routing cache on PC1 with the command:
     ```
     PC1% echo "1" > /proc/sys/net/ipv4/route/flush
     ```
   - Delete all static routes on Router 1 with the following commands:
     ```
     Router1(config)# no ip routing
     Router1(config)# ip routing
     ```
   - Build a new static routing entry on Router1 for network prefix 10.0.3.0/24 as follows:
     ```
     Router1(config)# ip route 10.0.3.0 255.255.255.0 10.0.2.2
     ```
3. Setup the routing table of PC2 in such a way that it provokes the transmission of an ICMP Route Redirect message as discussed above.
4. Save the contents of the routing table and the routing cache of Router1, Router2, and PC2.
5. Use Wireshark to capture the ICMP messages being sent, and issue a ping from PC2 to PC3:
   ```
   PC2% ping -c 5 10.0.3.10
   ```
6. Save the network traffic and the contents of the routing table and the routing cache after the ICMP Route Redirect messages.
7. Wait a few minutes and check the contents of the routing cache again. Save the output.

Question 5.1)
Is there a difference between the contents of the routing table and the routing cache immediately after the ICMP Route Redirect message?
This question could not be answered, due to the routing cache always being empty.

Question 5.2)

When you viewed the cache a few minutes later, what did you observe?

This question could not be answered, due to the routing cache always being empty.

Question 5.3)

Describe how the ICMP Route Redirect works using the output you saved. Include only relevant data from your saved output to support your explanations.

When Router1 receives the ICMP request for subnet $10.0.3.0/24$, it determines that the request should not be sent to himself, but to another host (namely Router2's vlan1 interface). Router1 will inform PC2 of this, by sending a redirect ICMP message (as captured in Wireshark). All the captured packets can be found in "traces/5.6.pcapng".

```
16 16.234116546   10.0.2.1              10.0.2.10             ICMP     70     Redirect
(Redirect for host)

Frame 16: 70 bytes on wire (560 bits),70 bytes captured (560 bits) on interface
0
Ethernet II,Src: 00:0d:65:17:01:29,Dst: 68:05:ca:36:31:f0
Internet Protocol Version 4,Src: 10.0.2.1,Dst: 10.0.2.10
Internet Control Message Protocol
```

The captured packet containing the redirect message.

Question 5.4)

Explain how Router1, in the above example, knows that datagrams destined to network 10.0.3.10 should be forwarded to 10.0.2.2?

Router1 knows this because in his routing table, he has an entry for subnet $10.0.3.0/24$. If the router wants to send packets to, for example, IP address $10.0.3.10$, it knows it should send it via gateway $10.0.2.2$. This way Router1 will know where the datagrams should be sent to.

Routing Loops

A potential problem when setting routing tables manually is that routing loops may occur. In this part of the lab, you intentionally configure a routing loop in the configuration of the routing table and observe what happens to network traffic in such a situation.

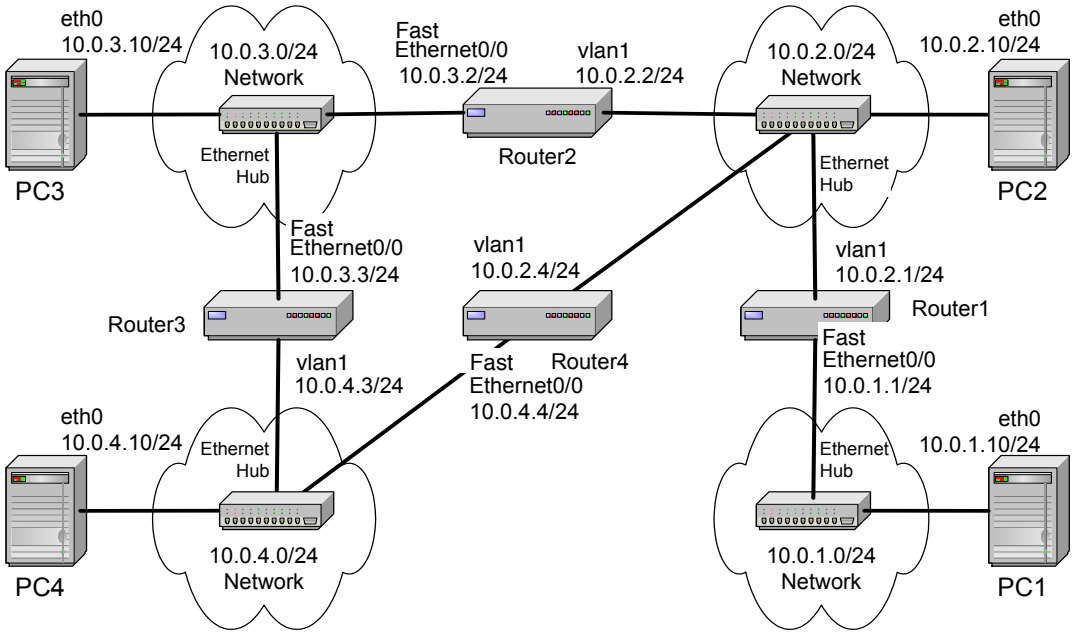

Figure 3.5: Network configuration for Part 6

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

Table 3.3: IP addresses for Part 6

Exercise 6.

1. Add Router4 to the network topology of Part 5 and configure the interfaces as shown in Figure 3.5 and Table 3.3 above.
2. Configure the routing tables of Router2, Router3 and Router4, so that an ICMP Echo Request message generated by a ping from PC4 to PC1 creates an infinite loop. Issue a `traceroute` to verify that a loop exists:

   ```
   PC4% traceroute 10.0.1.10
   ```

   You should observe that the traced path is a loop.
3. Start Wireshark sessions on PC2, PC3, and PC4.
4. Issue a ping from PC4 to

   ```
   PC4% ping -c 1 10.0.1.10
   ```

Observe in Wireshark that the same ICMP Echo Request message is looping.

5. Save the routing tables of Router2, Router3 and Router4. Count the number of times you see the ICMP Echo Request message, as captured by Wireshark on PC4. Save at least two of these ICMP Echo Request messages for the lab report.

Question 6.1)

Are the two ICMP packets that you saved identical? If not, what is different? Include the packet data in your lab report to substantiate your claims.

The packets differ because their TTL values are different. Every time the packet reaches the host again, its TTL is decreased by 3. In other words, every time the packet has to pass through a router, the TTL gets decreased by 1. This results in a decrease of 3 every time the packet completes a cycle.

```
    10 17.122662158   10.0.4.10            10.0.1.10            ICMP     98
       Echo (ping) request  id=0x09fd, seq=1/256, ttl=63 (no response found!)
2
Frame 10: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
4 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
6 Internet Control Message Protocol

8    11 17.123570697   10.0.4.10            10.0.1.10            ICMP     98
       Echo (ping) request  id=0x09fd, seq=1/256, ttl=60 (no response found!)

10 Frame 11: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
12 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
Internet Control Message Protocol
14
    12 17.124312552   10.0.4.10            10.0.1.10            ICMP     98
       Echo (ping) request  id=0x09fd, seq=1/256, ttl=57 (no response found!)
16
Frame 12: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
18 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
20 Internet Control Message Protocol

22   13 17.125021860   10.0.4.10            10.0.1.10            ICMP     98
       Echo (ping) request  id=0x09fd, seq=1/256, ttl=54 (no response found!)

24 Frame 13: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
26 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
Internet Control Message Protocol
28
    14 17.125800575   10.0.4.10            10.0.1.10            ICMP     98
       Echo (ping) request  id=0x09fd, seq=1/256, ttl=51 (no response found!)
30
Frame 14: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
32 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
34 Internet Control Message Protocol

36   15 17.126539234   10.0.4.10            10.0.1.10            ICMP     98
       Echo (ping) request  id=0x09fd, seq=1/256, ttl=48 (no response found!)

38 Frame 15: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
40 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
Internet Control Message Protocol
42
    16 17.127275052   10.0.4.10            10.0.1.10            ICMP     98
       Echo (ping) request  id=0x09fd, seq=1/256, ttl=45 (no response found!)
```

```
44
   Frame 16: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
46 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
   Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
48 Internet Control Message Protocol

50      17 17.127985597    10.0.4.10             10.0.1.10             ICMP     98
            Echo (ping) request  id=0x09fd, seq=1/256, ttl=42 (no response found!)

52 Frame 17: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
   Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
54 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
   Internet Control Message Protocol
56
        18 17.128775906    10.0.4.10             10.0.1.10             ICMP     98
            Echo (ping) request  id=0x09fd, seq=1/256, ttl=39 (no response found!)
58
   Frame 18: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
60 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
   Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
62 Internet Control Message Protocol

64      19 17.129540079    10.0.4.10             10.0.1.10             ICMP     98
            Echo (ping) request  id=0x09fd, seq=1/256, ttl=36 (no response found!)

66 Frame 19: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
   Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
68 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
   Internet Control Message Protocol
70
        20 17.130300709    10.0.4.10             10.0.1.10             ICMP     98
            Echo (ping) request  id=0x09fd, seq=1/256, ttl=33 (no response found!)
72
   Frame 20: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
74 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
   Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
76 Internet Control Message Protocol

78      21 17.131005274    10.0.4.10             10.0.1.10             ICMP     98
            Echo (ping) request  id=0x09fd, seq=1/256, ttl=30 (no response found!)

80 Frame 21: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
   Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
82 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
   Internet Control Message Protocol
84
        22 17.131787230    10.0.4.10             10.0.1.10             ICMP     98
            Echo (ping) request  id=0x09fd, seq=1/256, ttl=27 (no response found!)
86
   Frame 22: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
88 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
   Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
90 Internet Control Message Protocol

92      23 17.132525653    10.0.4.10             10.0.1.10             ICMP     98
            Echo (ping) request  id=0x09fd, seq=1/256, ttl=24 (no response found!)

94 Frame 23: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
   Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
96 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
   Internet Control Message Protocol
98
        24 17.133264886    10.0.4.10             10.0.1.10             ICMP     98
            Echo (ping) request  id=0x09fd, seq=1/256, ttl=21 (no response found!)
100
   Frame 24: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
102 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
```

```
    Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
104 Internet Control Message Protocol

106      25 17.134041804    10.0.4.10             10.0.1.10             ICMP     98
             Echo (ping) request  id=0x09fd, seq=1/256, ttl=18 (no response found!)

108 Frame 25: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
    Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
110 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
    Internet Control Message Protocol
112
         26 17.134843960    10.0.4.10             10.0.1.10             ICMP     98
             Echo (ping) request  id=0x09fd, seq=1/256, ttl=15 (no response found!)
114
    Frame 26: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
116 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
    Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
118 Internet Control Message Protocol

120      27 17.135573552    10.0.4.10             10.0.1.10             ICMP     98
             Echo (ping) request  id=0x09fd, seq=1/256, ttl=12 (no response found!)

122 Frame 27: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
    Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
124 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
    Internet Control Message Protocol
126
         28 17.136315318    10.0.4.10             10.0.1.10             ICMP     98
             Echo (ping) request  id=0x09fd, seq=1/256, ttl=9 (no response found!)
128
    Frame 28: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
130 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
    Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
132 Internet Control Message Protocol

134      29 17.137021118    10.0.4.10             10.0.1.10             ICMP     98
             Echo (ping) request  id=0x09fd, seq=1/256, ttl=6 (no response found!)

136 Frame 29: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
    Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
138 Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
    Internet Control Message Protocol
140
         30 17.137794247    10.0.4.10             10.0.1.10             ICMP     98
             Echo (ping) request  id=0x09fd, seq=1/256, ttl=3 (no response found!)
142
    Frame 30: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
144 Ethernet II, Src: 00:0d:bc:ef:b9:48, Dst: 00:0d:bc:ef:b1:a3
    Internet Protocol Version 4, Src: 10.0.4.10, Dst: 10.0.1.10
146 Internet Control Message Protocol
```

traces/6.5.PC2.txt

In the captured packets you can see that every subsequent packet has a TTL that is 3 lower than the previous packet.

Question 6.2)

Why does the ICMP Echo Request packet not loop forever in the network?

The packet does not loop forever because it is dropped when it reaches a TTL of 0. In the output included in the exercise above, the last time the packet passes by it has a TTL value of 3. The next time it should pass by, it gets dropped instead.

Network Prefixes and Routing

In this exercise you study the role that network prefixes (netmasks) play when hosts determine if a datagram can be directly delivered or if it must be sent to a router.

This part uses the network setup shown in Figure 3.6. The network includes one router, four hosts and two hubs. The IP addresses of all devices are given in Table 3.4. Here, each host has only a default route. In other words, the routing table at a host only knows about the directly connected networks and the default gateway.

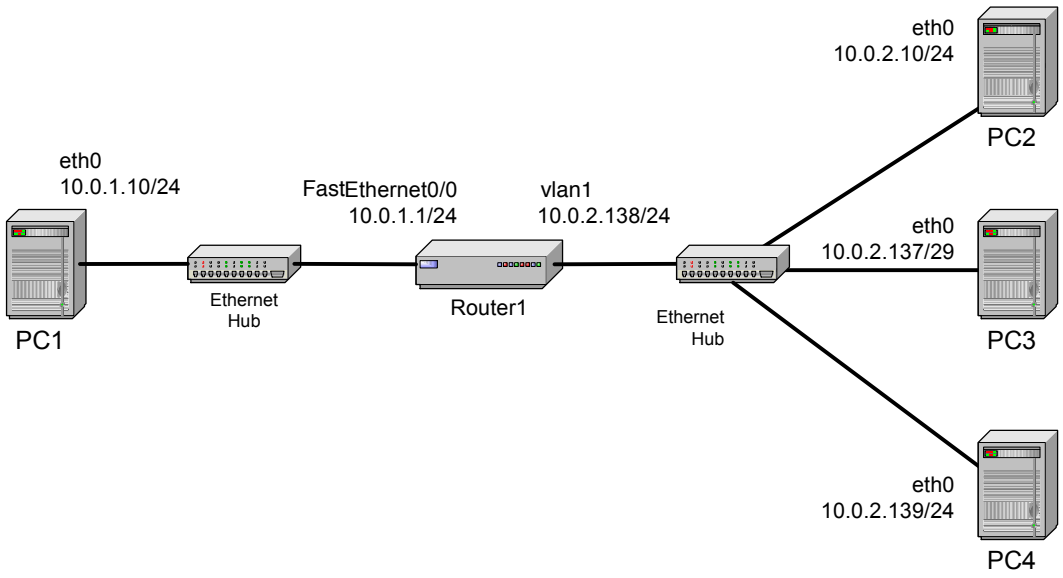

Figure 3.6: Network configuration for Part 7

Linux PC	eth0	eth1
PC1	10.0.1.10/24	Disabled
PC2	10.0.2.10/24	Disabled
PC3	10.0.2.137/29	Disabled
PC4	10.0.2.139/24	Disabled
Cisco Router	**FastEthernet0/0**	**vlan1**
Router1	10.0.1.1/24	10.0.2.138/24

Table 3.4: IP addresses for Part 7

Exercise 7.

In this exercise, you explore how hosts that are connected to the same local area network, but that have different network addresses or netmasks, communicate or fail to communicate.

1. Configure the hosts and the router to conform to the topology shown in Figure 3.6, using the IP addresses as given in Table 3.4. Note that PC2, PC3, and PC4 have different network addresses and different netmasks.

2. Add Router1 as default gateway on all hosts. For example, for PC1, the command is:

```
PC1% route add default gw 10.0.1.1
```

3. Issuing ping commands from PC1:

   a. Clear the ARP table on all hosts.

   b. Start Wireshark on PC1 and on PC4, and set the capture filter to capture ICMP and ARP packets only.

   c. Check the ARP tables, routing tables and routing caches of each host. Save the output. (Make a note that these are the table entries from Step 3 before the ping is issued.)

   d. Issue a `ping` command from PC1 to PC2 and PC3

```
PC1% ping -c 2 10.0.2.10
PC1% ping -c 2 10.0.2.137
```

   e. Save the ARP tables, routing tables and routing caches of each host (Make a note that these are the table entries from Step 3 after the ping is issued.)

   f. Save the output of the ping command at PC1 and the output of Wireshark on PC1 and PC4.

4. Issuing a ping command from PC3 to PC4:

   a. Clear the ARP table on all hosts.

   b. Start Wireshark on PC3, and set the capture filter to capture ICMP and ARP packets only.

   c. Check the ARP tables, routing tables and routing caches of each host. Save the output. (Make a note that these are the table entries from Step 4 before the ping is issued.)

   d. Issue a `ping` from PC3 to PC4.

```
PC3% ping -c 3 10.0.2.139
```

   e. Save the ARP tables, routing tables and routing caches of PC3 (Make a note that these are the table entries from Step 4 after the ping is issued.)

   f. Save the output of the `ping` command and the output of Wireshark on PC3.

5. Repeat Step 4, but this time issue a ping from PC3 to PC2. Note that once an entry is made in the routing cache, you cannot repeat the above experiment and obtain the same results; you have to wait until the routing cache is reset (which take some time).

Question 7.1)

Explain what you observed in Steps 3, 4 and 5. Use the saved data to support your answers. Provide explanations of the observations. Try to explain each observed phenomenon, e.g., if you observe more ICMP Echo Requests than ICMP Echo Replies, try to explain the reason.

Due to the same problem as a few exercises before (checking for updates, attempting to reach ubuntu archives through default gateway), multiple ARP requests kept being sent. We couldn't clear the ARP caches on the PCs properly for this reason. Another problem is that the routing cache in the PCs was always empty.

In the first scenario (step 3), the pings to the 2 hosts are both succesful. There are an equal amount of request and reply messages.

```
      4 2.177070657    10.0.1.10             10.0.2.10             ICMP     98
        Echo (ping) request  id=0x0b89, seq=1/256, ttl=64 (reply in 5)
2
Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
4 Ethernet II, Src: IntelCor_36:33:a0 (68:05:ca:36:33:a0), Dst: CiscoInc_17:01:29
    (00:0d:65:17:01:29)
  Internet Protocol Version 4, Src: 10.0.1.10, Dst: 10.0.2.10
6 Internet Control Message Protocol

8     5 2.178084427    10.0.2.10             10.0.1.10             ICMP     98
        Echo (ping) reply    id=0x0b89, seq=1/256, ttl=63 (request in 4)

10 Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
   Ethernet II, Src: CiscoInc_17:01:29 (00:0d:65:17:01:29), Dst: IntelCor_36:33:a0
    (68:05:ca:36:33:a0)
12 Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.1.10
   Internet Control Message Protocol
14
      7 3.178276617    10.0.1.10             10.0.2.10             ICMP     98
        Echo (ping) request  id=0x0b89, seq=2/512, ttl=64 (reply in 8)
16
   Frame 7: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
18 Ethernet II, Src: IntelCor_36:33:a0 (68:05:ca:36:33:a0), Dst: CiscoInc_17:01:29
    (00:0d:65:17:01:29)
   Internet Protocol Version 4, Src: 10.0.1.10, Dst: 10.0.2.10
20 Internet Control Message Protocol

22    8 3.179310868    10.0.2.10             10.0.1.10             ICMP     98
        Echo (ping) reply    id=0x0b89, seq=2/512, ttl=63 (request in 7)

24 Frame 8: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
   Ethernet II, Src: CiscoInc_17:01:29 (00:0d:65:17:01:29), Dst: IntelCor_36:33:a0
    (68:05:ca:36:33:a0)
26 Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.1.10
   Internet Control Message Protocol
28
     12 8.992819999    10.0.1.10             10.0.2.137            ICMP     98
        Echo (ping) request  id=0x0b8b, seq=1/256, ttl=64 (reply in 13)
30
   Frame 12: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
32 Ethernet II, Src: IntelCor_36:33:a0 (68:05:ca:36:33:a0), Dst: CiscoInc_17:01:29
    (00:0d:65:17:01:29)
   Internet Protocol Version 4, Src: 10.0.1.10, Dst: 10.0.2.137
34 Internet Control Message Protocol

36    13 8.993836239    10.0.2.137            10.0.1.10             ICMP     98
        Echo (ping) reply    id=0x0b8b, seq=1/256, ttl=63 (request in 12)

38 Frame 13: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
   Ethernet II, Src: CiscoInc_17:01:29 (00:0d:65:17:01:29), Dst: IntelCor_36:33:a0
    (68:05:ca:36:33:a0)
40 Internet Protocol Version 4, Src: 10.0.2.137, Dst: 10.0.1.10
   Internet Control Message Protocol
42
     14 9.993996410    10.0.1.10             10.0.2.137            ICMP     98
        Echo (ping) request  id=0x0b8b, seq=2/512, ttl=64 (reply in 15)
44
   Frame 14: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
46 Ethernet II, Src: IntelCor_36:33:a0 (68:05:ca:36:33:a0), Dst: CiscoInc_17:01:29
    (00:0d:65:17:01:29)
   Internet Protocol Version 4, Src: 10.0.1.10, Dst: 10.0.2.137
48 Internet Control Message Protocol

50    15 9.995009615    10.0.2.137            10.0.1.10             ICMP     98
        Echo (ping) reply    id=0x0b8b, seq=2/512, ttl=63 (request in 14)

52 Frame 15: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
```

```
Ethernet II, Src: CiscoInc_17:01:29 (00:0d:65:17:01:29), Dst: IntelCor_36:33:a0
    (68:05:ca:36:33:a0)
Internet Protocol Version 4, Src: 10.0.2.137, Dst: 10.0.1.10
Internet Control Message Protocol
```

traces/7.3.f.PC1.txt

The second scenario (step 4) also runs without any problems. There are the same amount of requests as replies. The reason why this is succesful is because PC3 (who has a net-mask of 29) can still see the IP address of PC4, meaning that it can answer to the ARP request from PC4 and send replies to the ICMP requests.

```
      9 15.727824732   68:05:ca:36:39:c7   00:0d:65:17:01:29   ARP   42
        Who has 10.0.2.138? Tell 10.0.2.137

Frame 9: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 00:0d:65:17:01:29
Address Resolution Protocol (request)

     10 15.728977361   00:0d:65:17:01:29   68:05:ca:36:39:c7   ARP   60
        10.0.2.138 is at 00:0d:65:17:01:29

Frame 10: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:36:39:c7
Address Resolution Protocol (reply)

     12 20.314576285   68:05:ca:36:31:f0   00:0d:65:17:01:29   ARP   60
        Who has 10.0.2.138? Tell 10.0.2.10

Frame 12: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 68:05:ca:36:31:f0, Dst: 00:0d:65:17:01:29
Address Resolution Protocol (request)

     13 20.315557454   00:0d:65:17:01:29   68:05:ca:36:31:f0   ARP   60
        10.0.2.138 is at 00:0d:65:17:01:29

Frame 13: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:36:31:f0
Address Resolution Protocol (reply)

     21 40.036401958   68:05:ca:39:e1:2f   00:0d:65:17:01:29   ARP   60
        Who has 10.0.2.138? Tell 10.0.2.139

Frame 21: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 68:05:ca:39:e1:2f, Dst: 00:0d:65:17:01:29
Address Resolution Protocol (request)

     22 40.037804539   00:0d:65:17:01:29   68:05:ca:39:e1:2f   ARP   60
        10.0.2.138 is at 00:0d:65:17:01:29

Frame 22: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:39:e1:2f
Address Resolution Protocol (reply)

     28 50.751852941   68:05:ca:36:39:c7   00:0d:65:17:01:29   ARP   42
        Who has 10.0.2.138? Tell 10.0.2.137

Frame 28: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 00:0d:65:17:01:29
Address Resolution Protocol (request)

     29 50.753373552   00:0d:65:17:01:29   68:05:ca:36:39:c7   ARP   60
        10.0.2.138 is at 00:0d:65:17:01:29

Frame 29: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:36:39:c7
Address Resolution Protocol (reply)
```

```
49      40 70.362761822   68:05:ca:36:31:f0     00:0d:65:17:01:29     ARP      60
            Who has 10.0.2.138? Tell 10.0.2.10

51  Frame 40: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:31:f0, Dst: 00:0d:65:17:01:29
53  Address Resolution Protocol (request)

55      41 70.363984787   00:0d:65:17:01:29     68:05:ca:36:31:f0     ARP      60
            10.0.2.138 is at 00:0d:65:17:01:29

57  Frame 41: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:36:31:f0
59  Address Resolution Protocol (reply)

61      44 75.775849053   68:05:ca:36:39:c7     00:0d:65:17:01:29     ARP      42
            Who has 10.0.2.138? Tell 10.0.2.137

63  Frame 44: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 00:0d:65:17:01:29
65  Address Resolution Protocol (request)

67      45 75.777598183   00:0d:65:17:01:29     68:05:ca:36:39:c7     ARP      60
            10.0.2.138 is at 00:0d:65:17:01:29

69  Frame 45: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:36:39:c7
71  Address Resolution Protocol (reply)

73      50 85.092455511   68:05:ca:39:e1:2f     00:0d:65:17:01:29     ARP      60
            Who has 10.0.2.138? Tell 10.0.2.139

75  Frame 50: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 68:05:ca:39:e1:2f, Dst: 00:0d:65:17:01:29
77  Address Resolution Protocol (request)

79      51 85.093471706   00:0d:65:17:01:29     68:05:ca:39:e1:2f     ARP      60
            10.0.2.138 is at 00:0d:65:17:01:29

81  Frame 51: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:39:e1:2f
83  Address Resolution Protocol (reply)

85      58 100.799840452  68:05:ca:36:39:c7     00:0d:65:17:01:29     ARP      42
            Who has 10.0.2.138? Tell 10.0.2.137

87  Frame 58: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 00:0d:65:17:01:29
89  Address Resolution Protocol (request)

91      59 100.801367050  00:0d:65:17:01:29     68:05:ca:36:39:c7     ARP      60
            10.0.2.138 is at 00:0d:65:17:01:29

93  Frame 59: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:36:39:c7
95  Address Resolution Protocol (reply)

97      68 120.410915162  68:05:ca:36:31:f0     00:0d:65:17:01:29     ARP      60
            Who has 10.0.2.138? Tell 10.0.2.10

99  Frame 68: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:31:f0, Dst: 00:0d:65:17:01:29
101 Address Resolution Protocol (request)

103     69 120.412175192  00:0d:65:17:01:29     68:05:ca:36:31:f0     ARP      60
            10.0.2.138 is at 00:0d:65:17:01:29
```

```
105 Frame 69: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:36:31:f0
107 Address Resolution Protocol (reply)

109      72 125.823848579  68:05:ca:36:39:c7     00:0d:65:17:01:29     ARP      42
             Who has 10.0.2.138? Tell 10.0.2.137

111 Frame 72: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 00:0d:65:17:01:29
113 Address Resolution Protocol (request)

115      73 125.825429258  00:0d:65:17:01:29     68:05:ca:36:39:c7     ARP      60
             10.0.2.138 is at 00:0d:65:17:01:29

117 Frame 73: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:36:39:c7
119 Address Resolution Protocol (reply)

121      75 126.087965088  68:05:ca:36:39:c7     ff:ff:ff:ff:ff:ff     ARP      42
             Who has 10.0.2.139? Tell 10.0.2.137

123 Frame 75: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:39:c7, Dst: ff:ff:ff:ff:ff:ff
125 Address Resolution Protocol (request)

127      76 126.088298722  68:05:ca:39:e1:2f     68:05:ca:36:39:c7     ARP      60
             10.0.2.139 is at 68:05:ca:39:e1:2f

129 Frame 76: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 68:05:ca:39:e1:2f, Dst: 68:05:ca:36:39:c7
131 Address Resolution Protocol (reply)

133      77 126.088304786  10.0.2.137            10.0.2.139            ICMP     98
             Echo (ping) request  id=0x0beb, seq=1/256, ttl=64 (reply in 78)

135 Frame 77: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 68:05:ca:39:e1:2f
137 Internet Protocol Version 4, Src: 10.0.2.137, Dst: 10.0.2.139
    Internet Control Message Protocol
139
         78 126.088676019  10.0.2.139            10.0.2.137            ICMP     98
             Echo (ping) reply    id=0x0beb, seq=1/256, ttl=64 (request in 77)
141
    Frame 78: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
143 Ethernet II, Src: 68:05:ca:39:e1:2f, Dst: 68:05:ca:36:39:c7
    Internet Protocol Version 4, Src: 10.0.2.139, Dst: 10.0.2.137
145 Internet Control Message Protocol

147      79 127.087891429  10.0.2.137            10.0.2.139            ICMP     98
             Echo (ping) request  id=0x0beb, seq=2/512, ttl=64 (reply in 80)

149 Frame 79: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 68:05:ca:39:e1:2f
151 Internet Protocol Version 4, Src: 10.0.2.137, Dst: 10.0.2.139
    Internet Control Message Protocol
153
         80 127.088265107  10.0.2.139            10.0.2.137            ICMP     98
             Echo (ping) reply    id=0x0beb, seq=2/512, ttl=64 (request in 79)
155
    Frame 80: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
157 Ethernet II, Src: 68:05:ca:39:e1:2f, Dst: 68:05:ca:36:39:c7
    Internet Protocol Version 4, Src: 10.0.2.139, Dst: 10.0.2.137
159 Internet Control Message Protocol

161      81 128.087840592  10.0.2.137            10.0.2.139            ICMP     98
             Echo (ping) request  id=0x0beb, seq=3/768, ttl=64 (reply in 82)
```

```
163 Frame 81: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 68:05:ca:39:e1:2f
165 Internet Protocol Version 4, Src: 10.0.2.137, Dst: 10.0.2.139
    Internet Control Message Protocol
167
        82 128.088244918  10.0.2.139           10.0.2.137           ICMP     98
           Echo (ping) reply    id=0x0beb, seq=3/768, ttl=64 (request in 81)
169
    Frame 82: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
171 Ethernet II, Src: 68:05:ca:39:e1:2f, Dst: 68:05:ca:36:39:c7
    Internet Protocol Version 4, Src: 10.0.2.139, Dst: 10.0.2.137
173 Internet Control Message Protocol

175     84 130.132433789  68:05:ca:39:e1:2f     00:0d:65:17:01:29     ARP      60
           Who has 10.0.2.138? Tell 10.0.2.139

177 Frame 84: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 68:05:ca:39:e1:2f, Dst: 00:0d:65:17:01:29
179 Address Resolution Protocol (request)

181     85 130.133467263  00:0d:65:17:01:29     68:05:ca:39:e1:2f     ARP      60
           10.0.2.138 is at 00:0d:65:17:01:29

183 Frame 85: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 00:0d:65:17:01:29, Dst: 68:05:ca:39:e1:2f
185 Address Resolution Protocol (reply)

187     87 131.092440993  68:05:ca:39:e1:2f     68:05:ca:36:39:c7     ARP      60
           Who has 10.0.2.137? Tell 10.0.2.139

189 Frame 87: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
    Ethernet II, Src: 68:05:ca:39:e1:2f, Dst: 68:05:ca:36:39:c7
191 Address Resolution Protocol (request)

193     88 131.092454704  68:05:ca:36:39:c7     68:05:ca:39:e1:2f     ARP      42
           10.0.2.137 is at 68:05:ca:36:39:c7

195 Frame 88: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
    Ethernet II, Src: 68:05:ca:36:39:c7, Dst: 68:05:ca:39:e1:2f
197 Address Resolution Protocol (reply)
```

traces/7.4.f.txt

The third scenario (step 5) is not entirely succesful. The first ICMP echo request does not receive a reply, but the following requests do. PC3 will first try to send the ICMP packets through Router1, because it cannot see PC2 directly (due to its netmask). Router1 sees the first ICMP packet, and will determine that the packet should not be directed to him. As a result of this, the router will send an ICMP redirect message to PC3. This is the reason why there is no reply to the first ICMP request.
After receiving the redirect message, PC3 will send the ICMP packets directly to PC2. The pings are succesful from this point on.

The redirect packet:

```
No.     Time           Source                Destination           Protocol Length
Info
156 263.065460462  10.0.2.138            10.0.2.137            ICMP     70     Redirect
(Redirect for host)

Frame 156: 70 bytes on wire (560 bits),70 bytes captured (560 bits) on interface
0
```

```
Ethernet II,Src: 00:0d:65:17:01:29,Dst: 68:05:ca:36:39:c7
Internet Protocol Version 4,Src: 10.0.2.138,Dst: 10.0.2.137
Internet Control Message Protocol
```

The ping output:

```
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=0.437 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=64 time=0.402 ms
64 bytes from 10.0.2.10: icmp_seq=3 ttl=64 time=0.393 ms

--- 10.0.2.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.393/0.410/0.437/0.030 ms
```

traces/7.5.f.ping.txt

All the packets can be found at "traces/7.5.f.pcapng" (there are alot of 'junk' packets because of the reason listed above).

Question 7.2)

If PC3 had no default entry in its table, would you have seen the same results? Explain for each of the pings above what would have been different.

For the first scenario (step 3) the ping would have failed. PC3 would not have been able to know where to sent the reply packet to because PC1 is located on another subnet, not visible to PC3.

The second scenario (step 4) would still work: PC3 can see PC4 and therefor send packets directly.

The last scenario (step 5) would fail. PC3 cannot see PC2 directly. A ping could not be sent by PC3 for this reason. The default gateway made it so PC3 could send the ICMP packets through the router to PC2.