

LAB 1
CONSTRUCT A SIMPLE NETWORK



Name: Nguyễn Võ Thuận Thiên

ID: B2005893

Group: M01

Submission: an ID_NAME_Lab01.pdf file describes clearly how did you solve the problem

Exercise 0: My home directory

Answer: \$cd

A screenshot of a Linux desktop environment, specifically Ubuntu, running in a virtual machine. The desktop has a dark theme. On the left, there's a dock with icons for the Dash, Home, and other applications. In the center, a terminal window is open with the title "Terminal". The terminal shows the command "thienng@thienng-VirtualBox:~\$ cd" being typed. The status bar at the bottom of the terminal window indicates the date and time as "Thg 10 13 08:14".

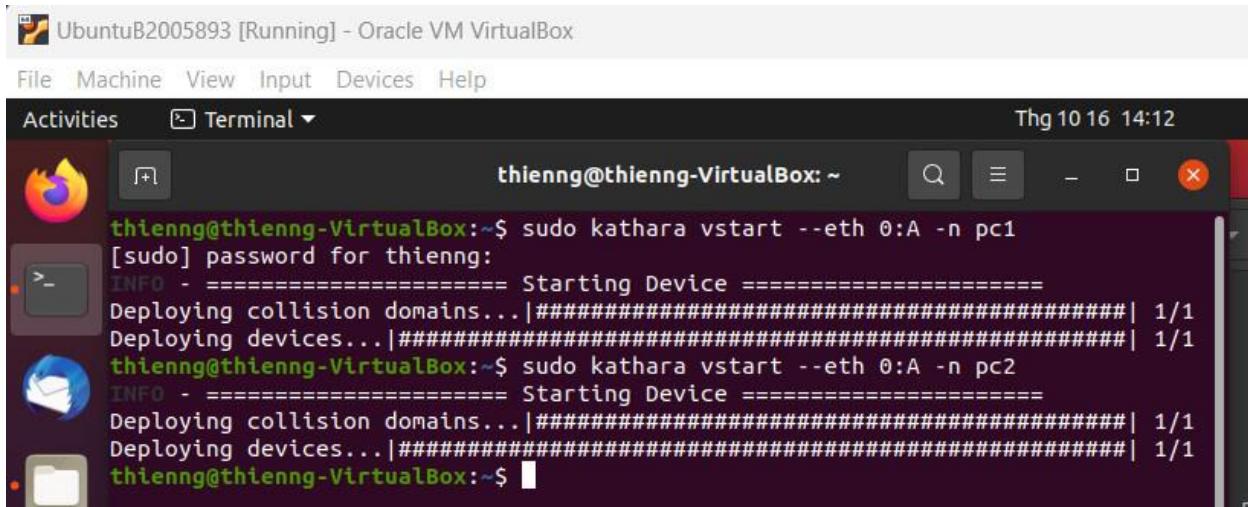
```
thienng@thienng-VirtualBox:~$ cd
```

Exercise 1: Construct a simple network with two hosts connected to the same collision domain

Answer:

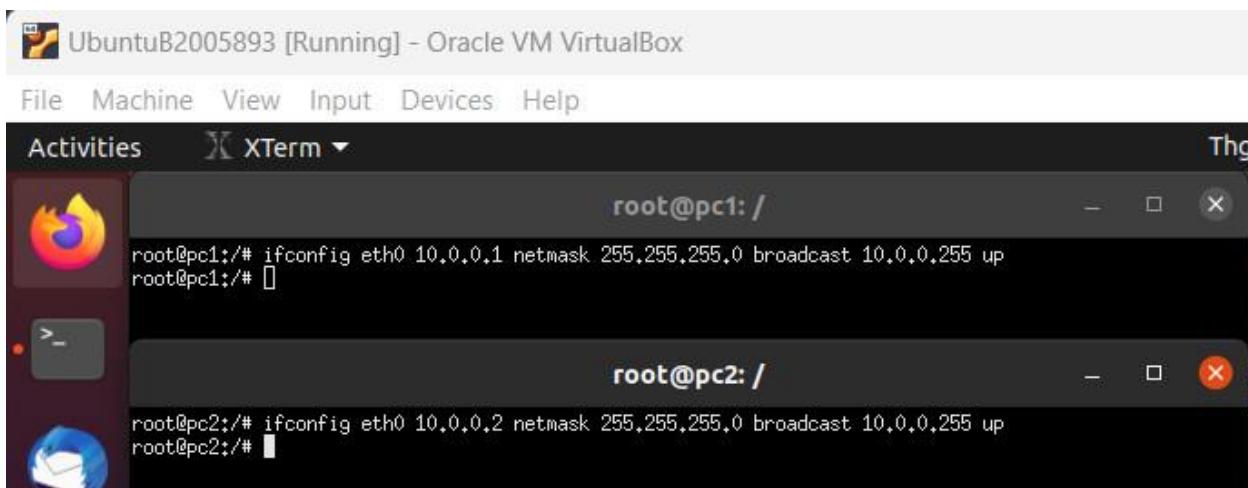
1. Creating the VMs

```
$ sudo kathara vstart --eth 0:A -n pc1  
$ sudo kathara vstart --eth 0:A -n pc2
```



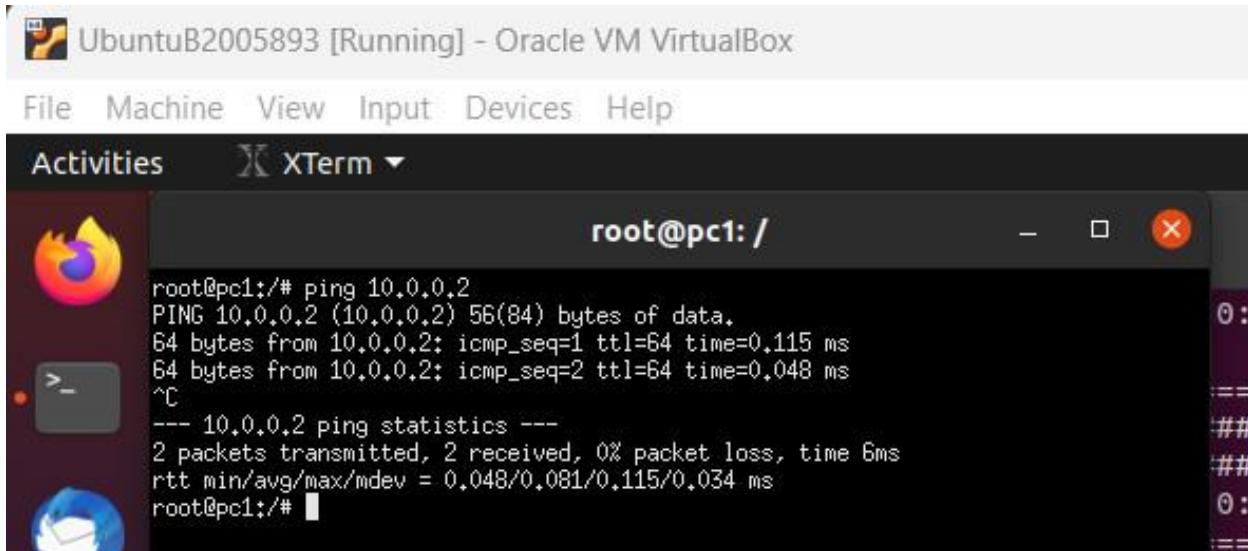
2. Configuring network interfaces

From pc1: #ifconfig eth0 10.0.0.1 netmask 255.255.255.0 broadcast 10.0.0.255 up
From pc2: #ifconfig eth0 10.0.0.2 netmask 255.255.255.0 broadcast 10.0.0.255 up



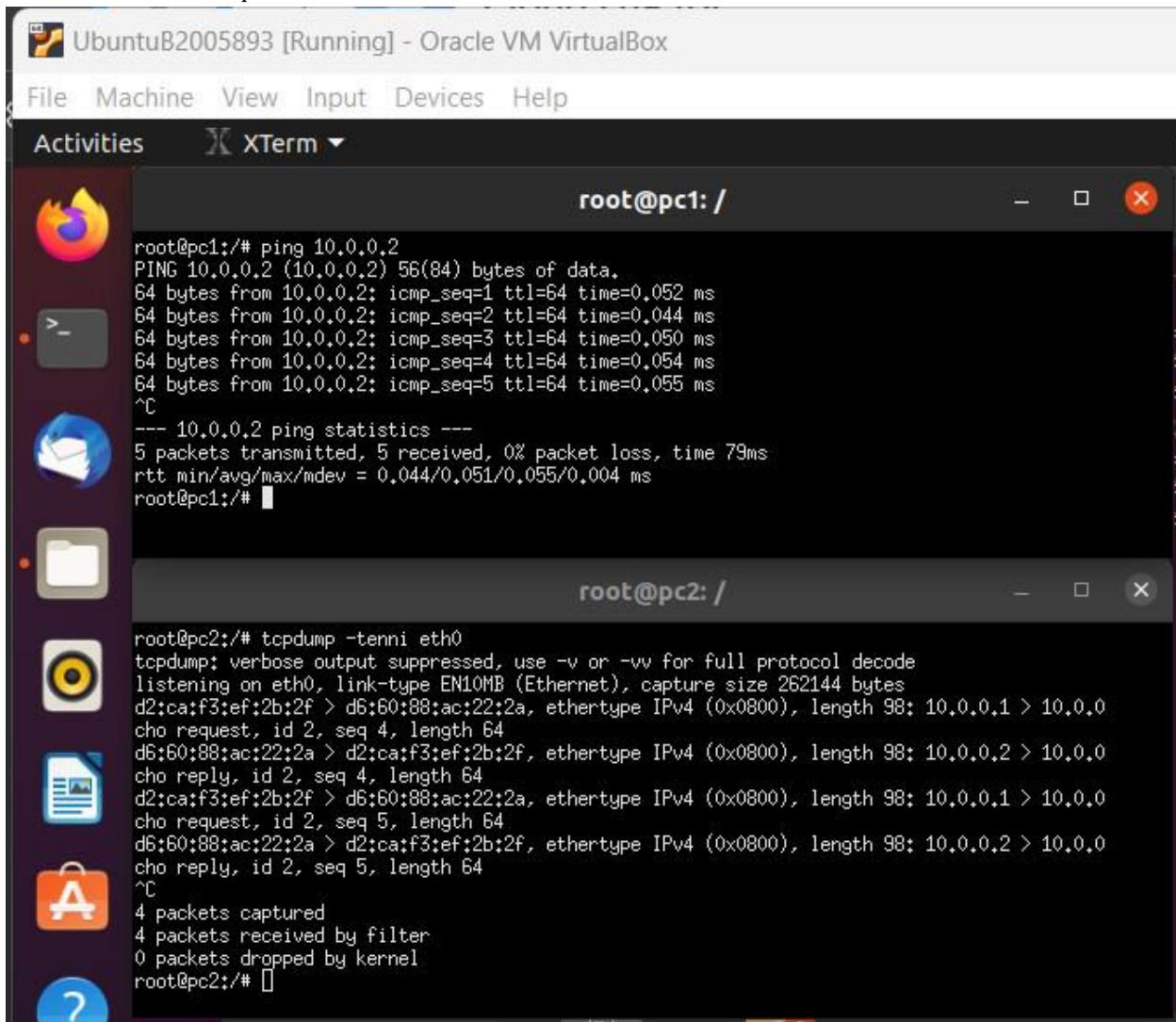
3. Ping

From pc1: # ping 10.0.0.2



```
root@pc1:/# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.115 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.048 ms
^C
--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 6ms
rtt min/avg/max/mdev = 0.048/0.081/0.115/0.034 ms
root@pc1:/#
```

4. A look at the packets



```
root@pc1:/# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.054 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.055 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 79ms
rtt min/avg/max/mdev = 0.044/0.051/0.055/0.004 ms
root@pc1:/#
```



```
root@pc2:/# tcpdump -tleni eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
d2:ca:f3:ef:2b:2f > d6:60:88:ac:22:2a, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0
cho request, id 2, seq 4, length 64
d6:60:88:ac:22:2a > d2:ca:f3:ef:2b:2f, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0
cho reply, id 2, seq 4, length 64
d2:ca:f3:ef:2b:2f > d6:60:88:ac:22:2a, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0
cho request, id 2, seq 5, length 64
d6:60:88:ac:22:2a > d2:ca:f3:ef:2b:2f, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0
cho reply, id 2, seq 5, length 64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
root@pc2:/#
```

Ping from pc1. at the same time, sniff from pc2 (ctrl+C to interrupt)

On pc1: # ping 10.0.0.2

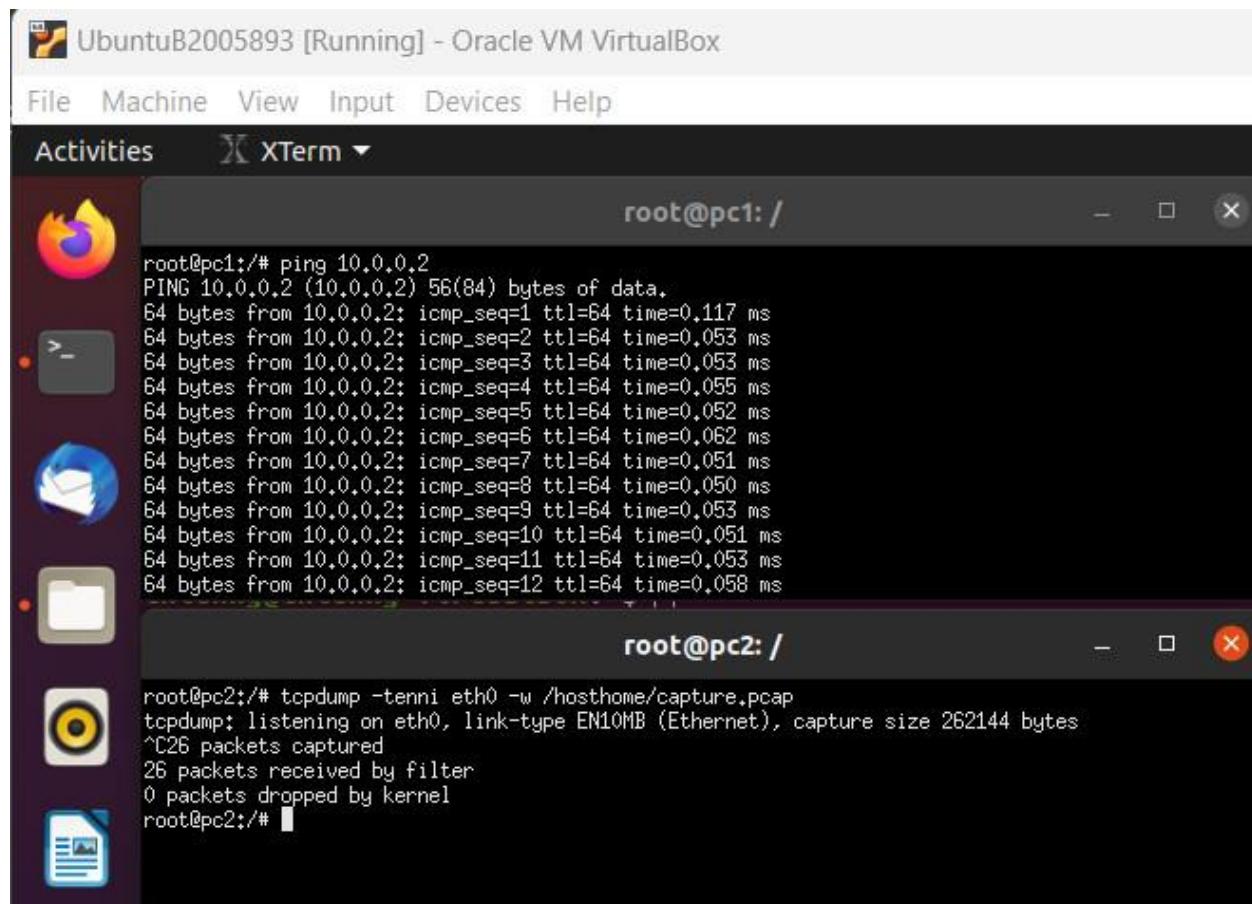
On pc2: # tcpdump -telli eth0

5. *Looking at the packets with a graphical interface*

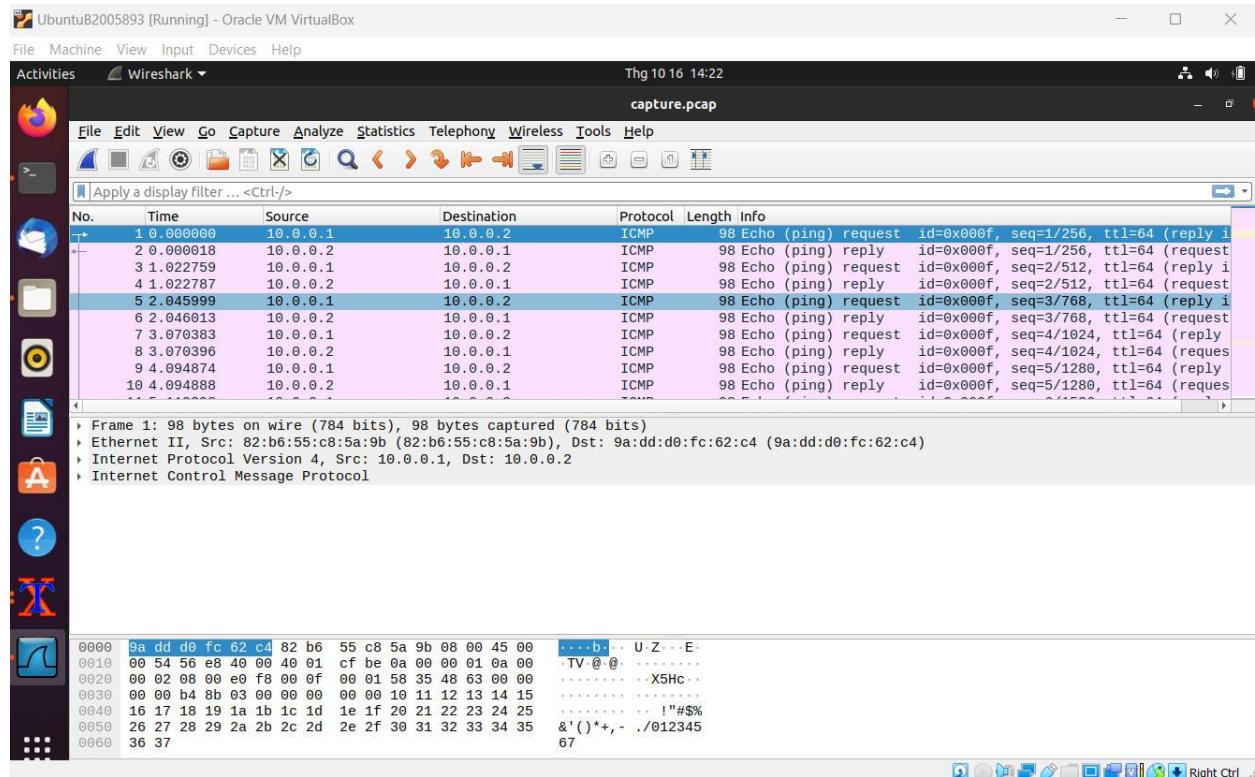
Same as (4), but store sniffed packets into file capture.pcap (on the host machine)

On pc1: # ping 10.0.0.2

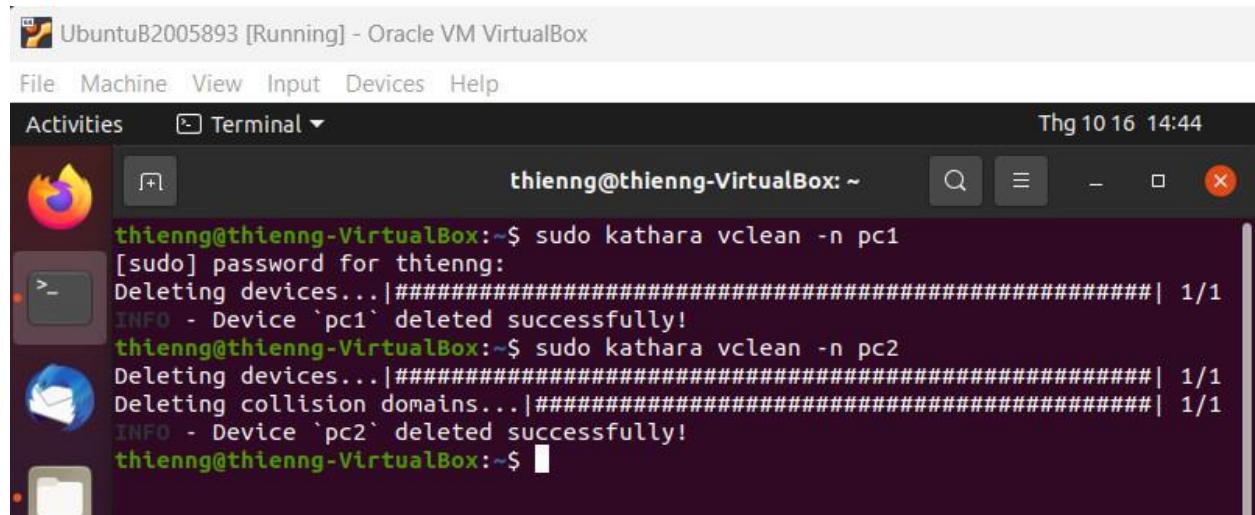
On pc2: # tcpdump -telli eth0 -w /hosthome/capture.pcap



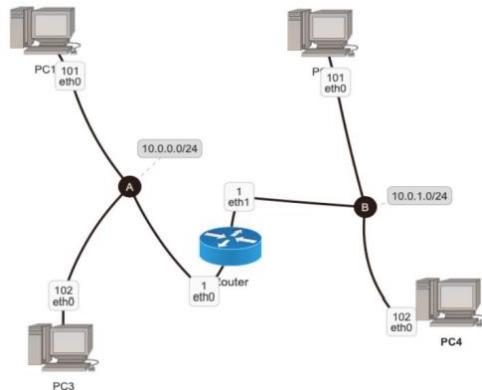
6. Open capture.pcap on real host machine using wireshark



7. Delete the VMs



Exercise 2: Construct the following network



1. Create specific files and folders

```
$ touch lab.conf pc1.startup pc2.startup pc3.startup pc4.startup
$ mkdir pc1 pc2 pc3 pc4
```

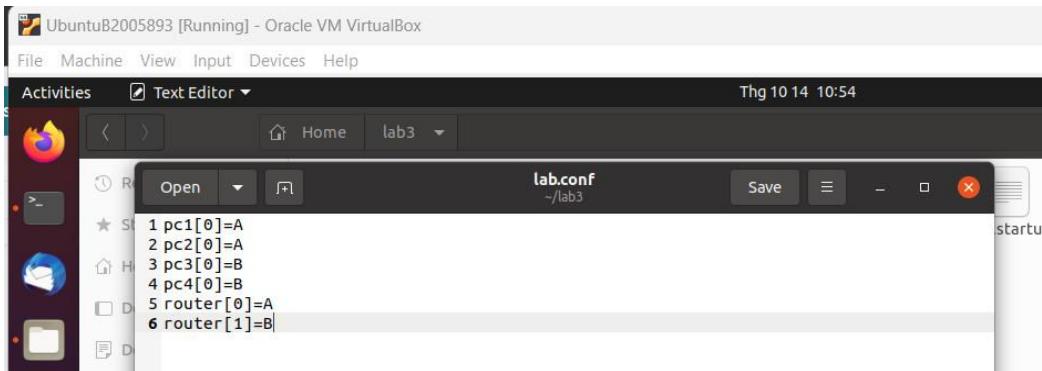
```
UbuntuB2005893 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Thg 10 14 10:53
thienng@thienng-VirtualBox:~/lab3
thienng@thienng-VirtualBox:~/lab3$ touch lab.conf pc1.startup pc2.startup pc3.startup pc4.startup
thienng@thienng-VirtualBox:~/lab3$ mkdir pc1 pc2 pc3 pc4
thienng@thienng-VirtualBox:~/lab3$
```

2. Here is the tree

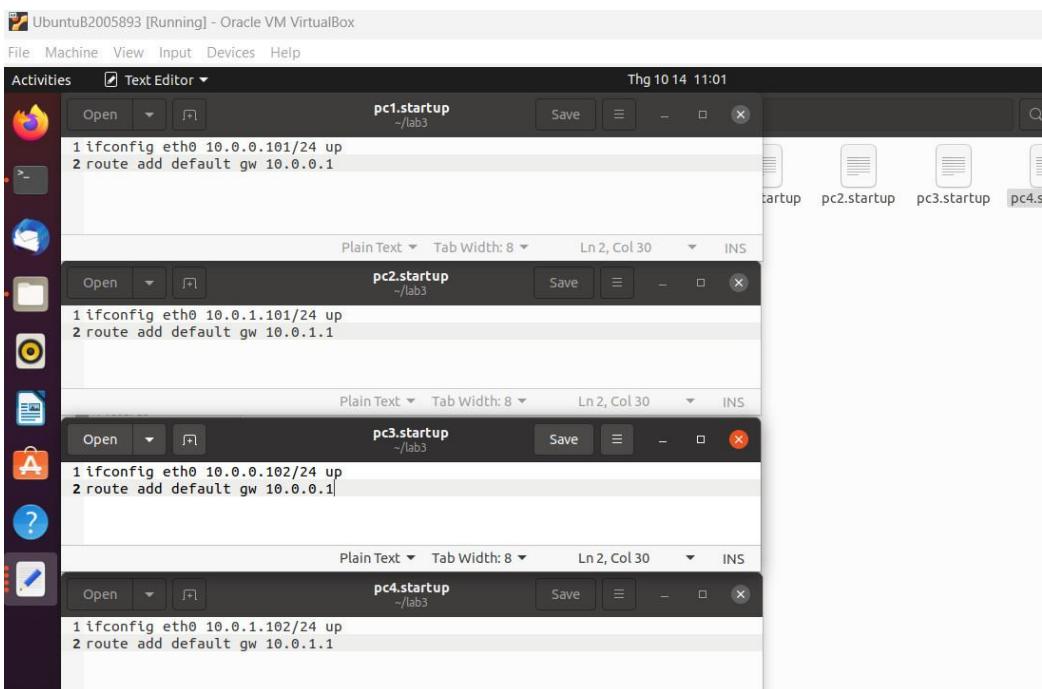
```
$ ls
```

```
UbuntuB2005893 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Thg 10 14 11:11
thienng@thienng-VirtualBox:~/lab3
thienng@thienng-VirtualBox:~/lab3$ sudo kathara lstart
[sudo] password for thienng:
INFO - ===== Starting Network Scenario =====
Deploying collision domains...|#####
Deploying devices...|#####
thienng@thienng-VirtualBox:~/lab3$ tree
.
├── lab.conf
├── pc1
├── pc2
├── pc3
├── pc4
└── router.startup
    └── shared
.
5 directories, 6 files
thienng@thienng-VirtualBox:~/lab3$
```

3. Enter network configurations into files



```
1 pc1[0]=A
2 pc2[0]=A
3 pc3[0]=B
4 pc4[0]=B
5 router[0]=A
6 router[1]=B
```

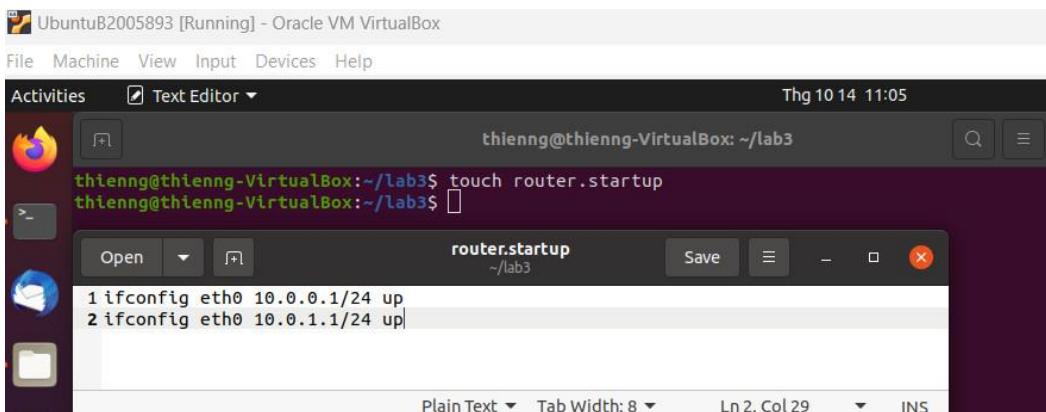


```
pc1.startup
1 ifconfig eth0 10.0.0.101/24 up
2 route add default gw 10.0.0.1

pc2.startup
1 ifconfig eth0 10.0.1.101/24 up
2 route add default gw 10.0.1.1

pc3.startup
1 ifconfig eth0 10.0.0.102/24 up
2 route add default gw 10.0.0.1

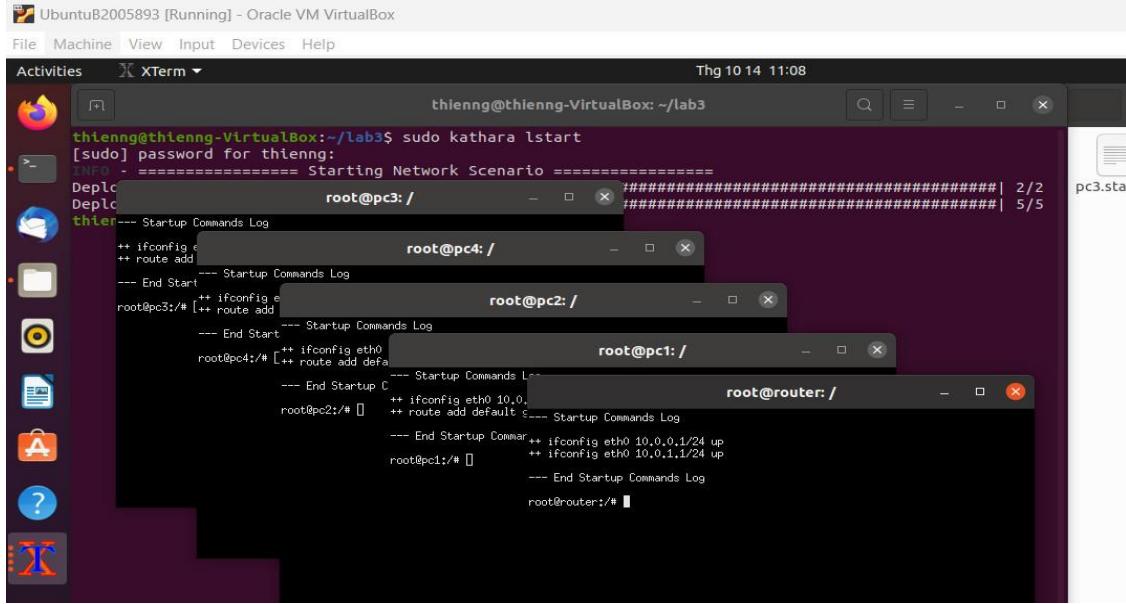
pc4.startup
1 ifconfig eth0 10.0.1.102/24 up
2 route add default gw 10.0.1.1
```



```
thienng@thienng-VirtualBox:~/lab3$ touch router.startup
thienng@thienng-VirtualBox:~/lab3$ cat router.startup
1 ifconfig eth0 10.0.0.1/24 up
2 ifconfig eth0 10.0.1.1/24 up
```

4. Start kathara

```
$sudo kathara lstart
```

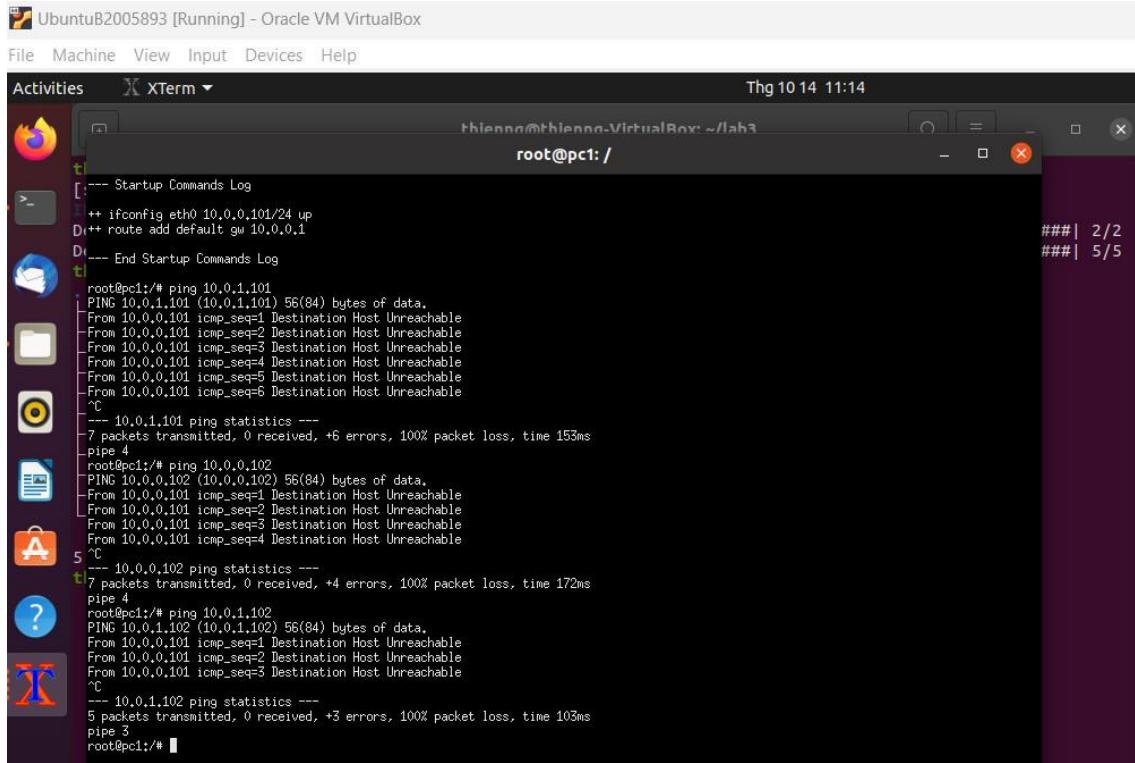


The screenshot shows a desktop environment with several open terminal windows. The title bar of the main window indicates it is running on a host named 'pc1'. The terminal output shows the command \$sudo kathara lstart being run, followed by logs from other hosts: pc2, pc3, and router. The logs include startup commands like ifconfig and route add, and network interface configuration details.

```
thienng@thienng-VirtualBox:~/lab3$ sudo kathara lstart
[sudo] password for thienng:
INFO - ===== Starting Network Scenario =====
Deployer          root@pc3: /
Deployer          root@pc4: /
thienng--- Startup Commands Log
++ ifconfig eth0      root@pc4: /
++ route add default  root@pc4: /
--- End Start        --- Startup Commands Log
root@pc3:/# [++ route add
root@pc3:/# [++ ifconfig eth0      root@pc2: /
--- End Start        --- Startup Commands Log
root@pc4:/# [++ route add defa
root@pc4:/# [++ ifconfig eth0 10.0.0.1/24 up
root@pc2:/# [++ ifconfig eth0 10.0.0.1/24 up
root@pc1:/# [++ ifconfig eth0 10.0.0.1/24 up
root@pc1:/# [++ ifconfig eth0 10.0.1.1/24 up
--- End Startup Comma
root@pc1:/# [++ ifconfig eth0 10.0.0.1/24 up
root@router:/# [++ ifconfig eth0 10.0.1.1/24 up
--- End Startup Commands Log
root@router:/# [
```

5. Ping to other vms from pc1

```
# ping 10.0.1.101
# ping 10.0.0.102
# ping 10.0.1.102
```

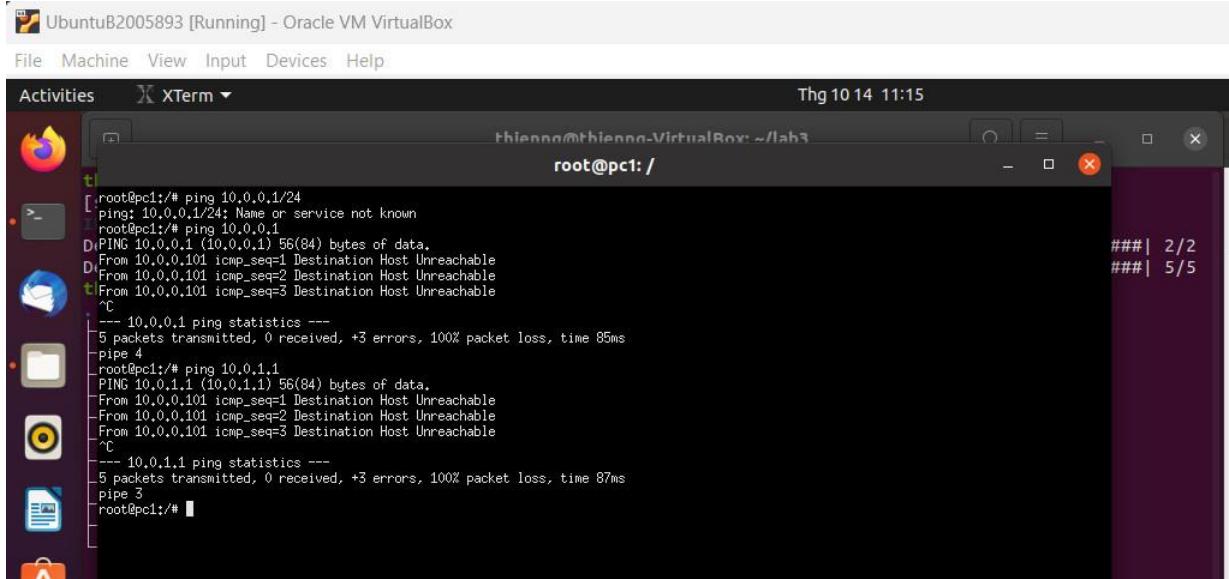


The screenshot shows a desktop environment with a terminal window running on host 'pc1'. The user has issued three ping commands: # ping 10.0.1.101, # ping 10.0.0.102, and # ping 10.0.1.102. The terminal output shows the results of these pings, including the number of packets transmitted, received, errors, and the round-trip time (RTT). The logs also show the startup commands for the host.

```
thienng@thienng-VirtualBox:~/lab3$ ping 10.0.1.101
PING 10.0.1.101 (10.0.1.101) 56(84) bytes of data.
From 10.0.0.101 icmp_seq=1 Destination Host Unreachable
From 10.0.0.101 icmp_seq=2 Destination Host Unreachable
From 10.0.0.101 icmp_seq=3 Destination Host Unreachable
From 10.0.0.101 icmp_seq=4 Destination Host Unreachable
From 10.0.0.101 icmp_seq=5 Destination Host Unreachable
From 10.0.0.101 icmp_seq=6 Destination Host Unreachable
^C
--- 10.0.1.101 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 153ms
PIPE 4
root@pc1:/# ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
From 10.0.0.101 icmp_seq=1 Destination Host Unreachable
From 10.0.0.101 icmp_seq=2 Destination Host Unreachable
From 10.0.0.101 icmp_seq=3 Destination Host Unreachable
From 10.0.0.101 icmp_seq=4 Destination Host Unreachable
5 ^C
--- 10.0.0.102 ping statistics ---
7 packets transmitted, 0 received, +4 errors, 100% packet loss, time 172ms
PIPE 4
root@pc1:/# ping 10.0.1.102
PING 10.0.1.102 (10.0.1.102) 56(84) bytes of data.
From 10.0.0.101 icmp_seq=1 Destination Host Unreachable
From 10.0.0.101 icmp_seq=2 Destination Host Unreachable
From 10.0.0.101 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.1.102 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 103ms
PIPE 3
root@pc1:/# [
```

6. Ping to 2 interfaces of the router

```
# ping 10.0.0.1  
# ping 10.0.1.1
```



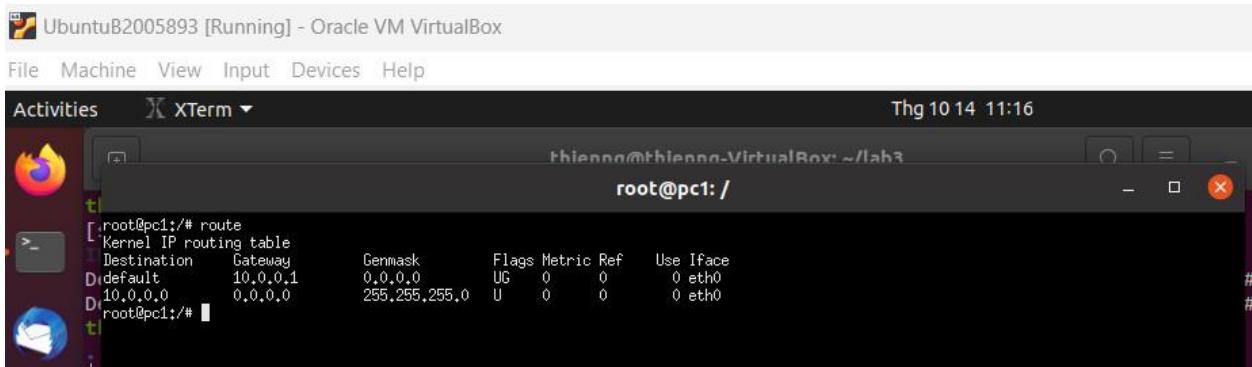
The screenshot shows a terminal window titled "UbuntuB2005893 [Running] - Oracle VM VirtualBox". The window title bar includes "File Machine View Input Devices Help" and the date/time "Thg 10 14 11:15". The terminal window title is "Activities XTerm". The terminal content shows:

```
root@pc1:/# ping 10.0.0.1/24
ping: 10.0.0.1/24: Name or service not known
root@pc1:/# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 10.0.0.101 icmp_seq=1 Destination Host Unreachable
From 10.0.0.101 icmp_seq=2 Destination Host Unreachable
From 10.0.0.101 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 85ms
pipe 4
root@pc1:/# ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
From 10.0.0.101 icmp_seq=1 Destination Host Unreachable
From 10.0.0.101 icmp_seq=2 Destination Host Unreachable
From 10.0.0.101 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.1.1 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 87ms
pipe 3
root@pc1:/#
```

On the right side of the terminal window, there are two progress bars: one for "2/2" and another for "5/5".

7. Route pc1

```
# route
```

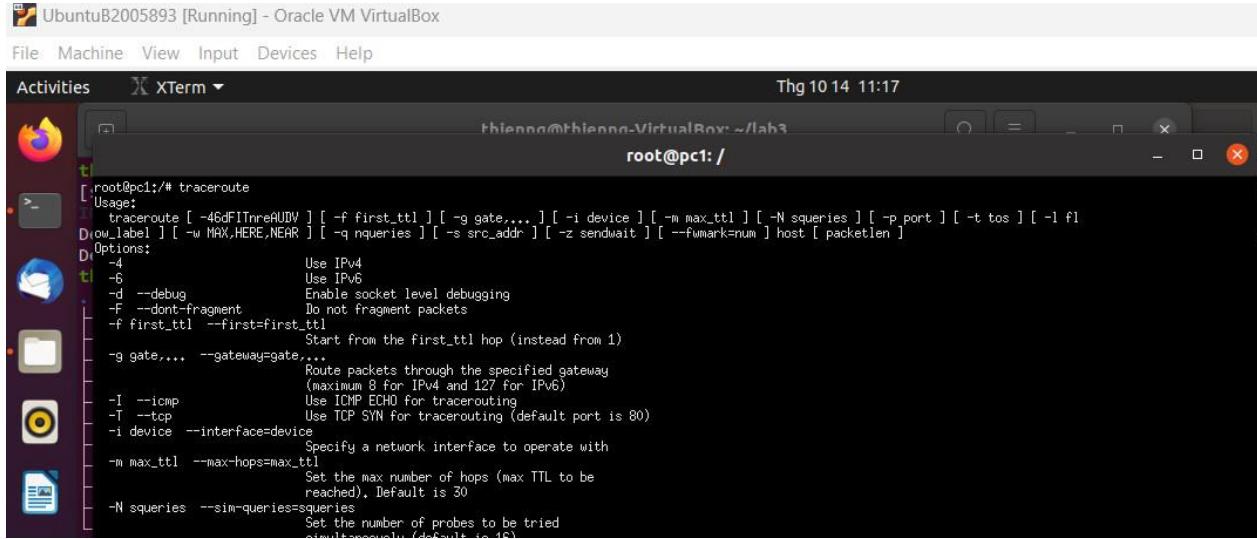


The screenshot shows a terminal window titled "UbuntuB2005893 [Running] - Oracle VM VirtualBox". The window title bar includes "File Machine View Input Devices Help" and the date/time "Thg 10 14 11:16". The terminal window title is "Activities XTerm". The terminal content shows:

```
root@pc1:/# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
Default        10.0.0.1        0.0.0.0         UG    0      0        0 eth0
10.0.0.0        0.0.0.0        255.255.255.0   U     0      0        0 eth0
root@pc1:/#
```

8. Traceroute pc1

```
#traceroute
```

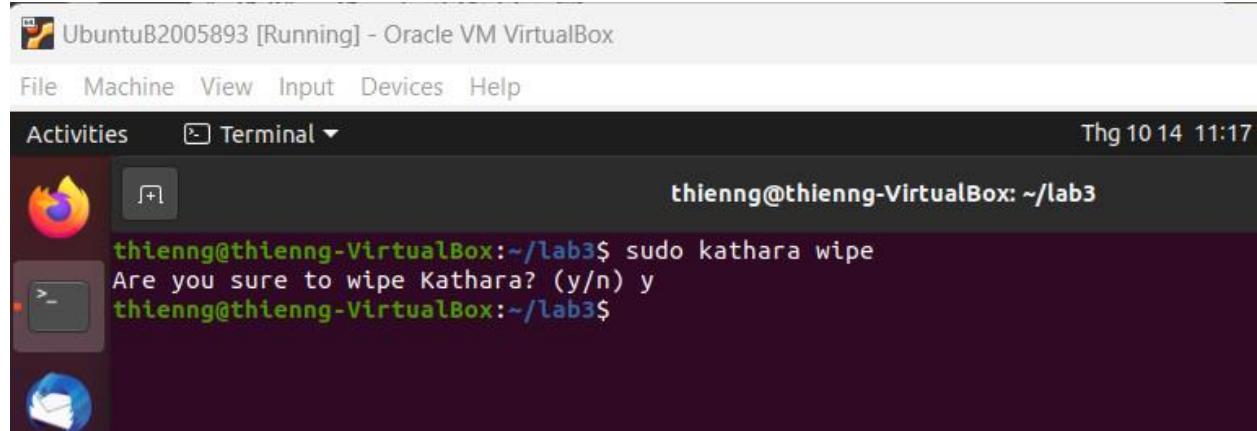


A screenshot of a terminal window titled "XTerm" running on an Ubuntu 20.04 LTS system. The window shows the command "traceroute" being run with no arguments, displaying its usage information. The usage text includes options for IPv4 (-4), IPv6 (-6), debugging (-d), fragment control (-F), starting from a specific TTL (-f), gateway selection (-g), protocol selection (-I/-T), interface selection (-i), maximum TTL (-m), and probe count (-N). The terminal window also shows the date and time as "Thg 10 14 11:17".

```
[root@pc1:/# traceroute
Usage:
traceroute [ -4dFITrreAUDV ] [ -f first_ttl ] [ -g gate,... ] [ -i device ] [ -m max_ttl ] [ -N queries ] [ -p port ] [ -t tos ] [ -l fl
Drow_label ] [ -w MAX_HERE_NEAR ] [ -q nqueries ] [ -s src_addr ] [ -z sendwait ] [ --fmark-num ] host [ packetlen ]
Options:
-D          Use IPv4
-d          Use IPv6
-d          --debug           Enable socket level debugging
-F          --dont-fragment   Do not fragment packets
-f first_ttl --first=first_ttl
                   Start from the first_ttl hop (instead from 1)
-g gate,...  --gateway=gate...
                   Route packets through the specified gateway
                   (maximum 8 for IPv4 and 127 for IPv6)
-I          --icmp           Use ICMP ECHO for tracerouting
-T          --tcp            Use TCP SYN for tracerouting (default port is 80)
-i device   --interface=device
                   Specify a network interface to operate with
-m max_ttl --max-hops=max_ttl
                   Set the max number of hops (max TTL to be
                   reached), Default is 30
-N queries  --sim-queries=queries
                   Set the number of probes to be tried
                   simultaneously (default is 10)
```

9. Delete all vms

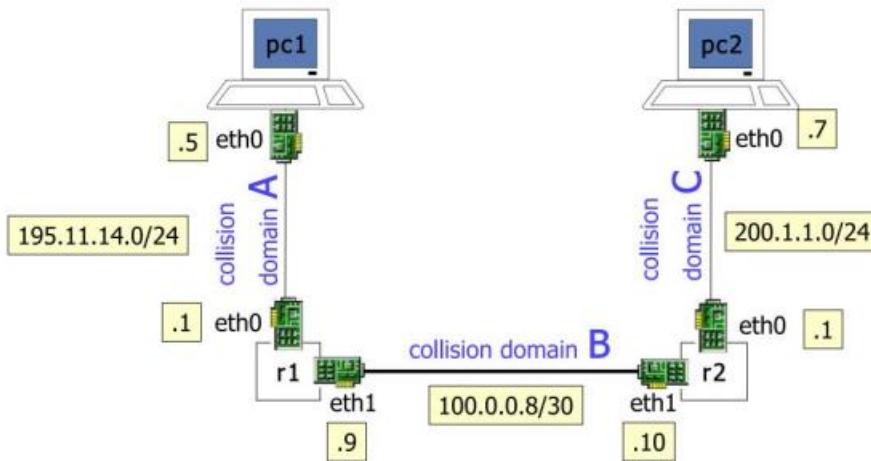
```
$ sudo kathara wipe
```



A screenshot of a terminal window titled "Terminal" running on an Ubuntu 20.04 LTS system. The window shows the command "sudo kathara wipe" being run by a user named "thienng". A confirmation prompt "Are you sure to wipe Kathara? (y/n)" is displayed, followed by the user's response "y". The terminal window also shows the date and time as "Thg 10 14 11:17".

```
thienng@thienng-VirtualBox:~/lab3$ sudo kathara wipe
Are you sure to wipe Kathara? (y/n) y
thienng@thienng-VirtualBox:~/lab3$
```

Exercise 3: Construct the following network



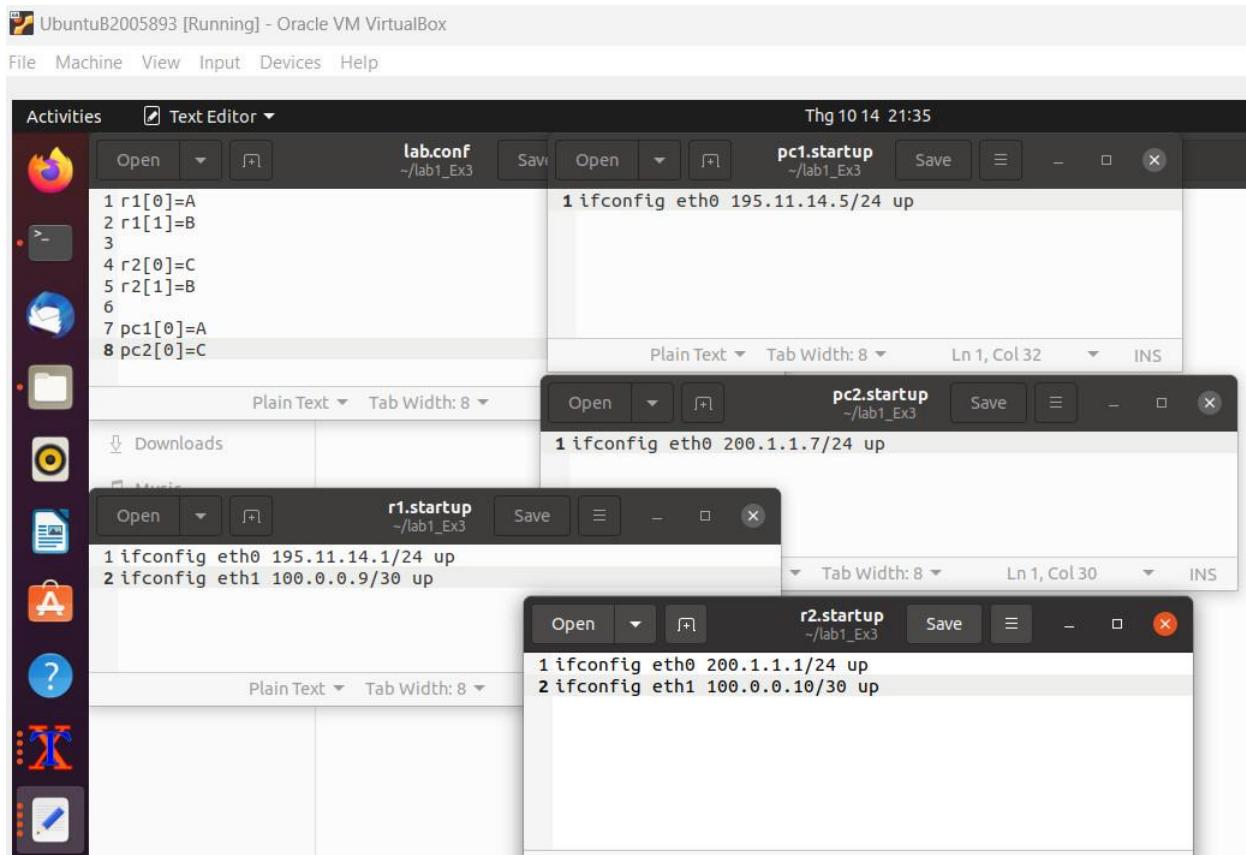
1. Create lab directory hierarchy

```
$ mkdir lab1_Ex3
$ cd ~/lab1_Ex3
$ touch pc1.startup pc2.startup r1.startup r2.startup lab.conf
$ tree
```

```
UbuntuB2005893 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal ▾ Thg 10 14 11:32
thienng@thienng-VirtualBox: ~/lab1_Ex3
thienng@thienng-VirtualBox:~$ mkdir lab1_Ex3
thienng@thienng-VirtualBox:~$ cd ~/lab1_Ex3
thienng@thienng-VirtualBox:~/lab1_Ex3$ touch pc1.startup pc2.startup r1.startup r2.startup lab.conf
thienng@thienng-VirtualBox:~/lab1_Ex3$ tree
.
├── lab.conf
└── startup
    ├── pc1.startup
    ├── pc2.startup
    ├── r1.startup
    └── r2.startup

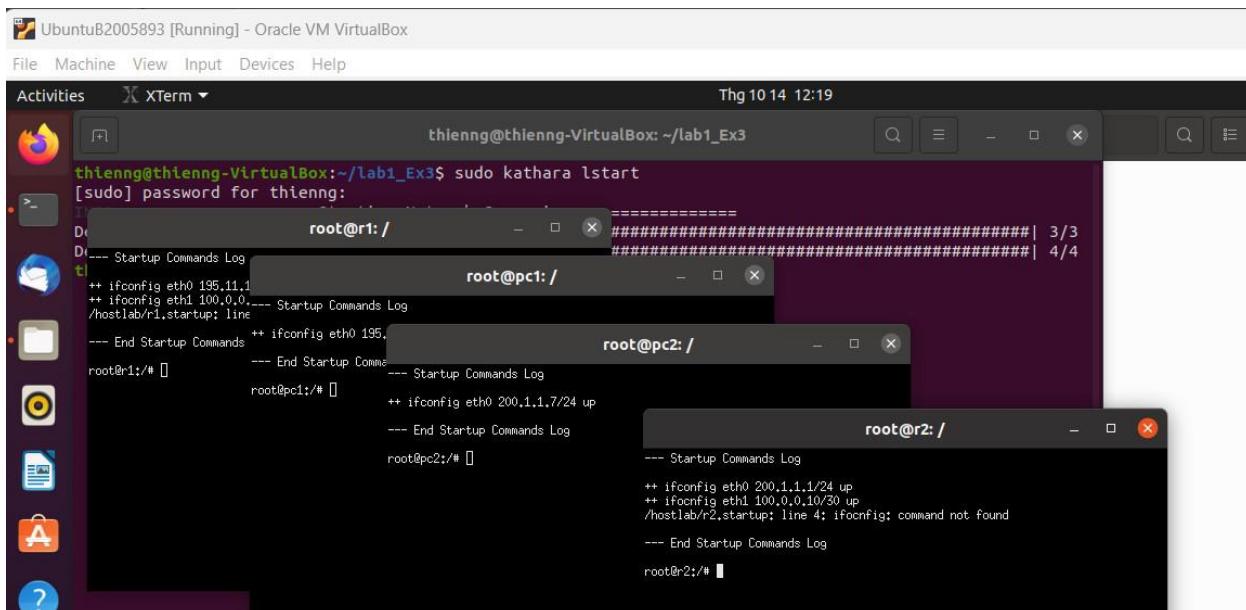
0 directories, 5 files
thienng@thienng-VirtualBox:~/lab1_Ex3$
```

2. Enter network configuration into files



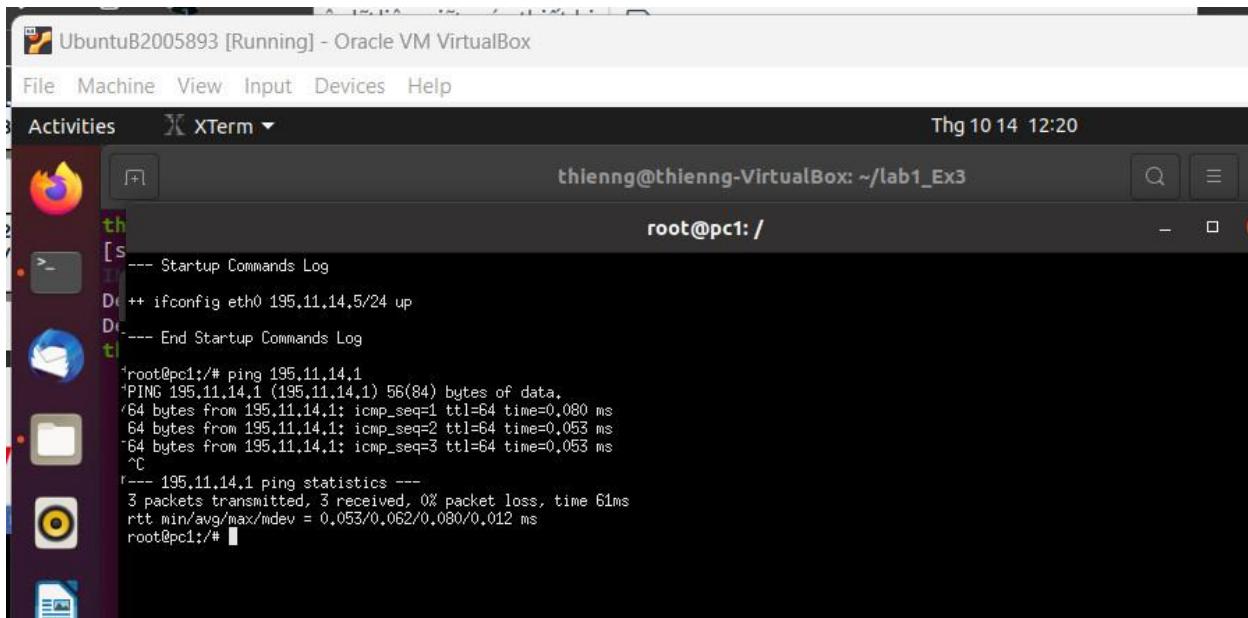
3. Start kathara

```
$ sudo kathara lstart
```



4. Test connectivity (1) – from pc1

```
# ping 195.11.14.1
```

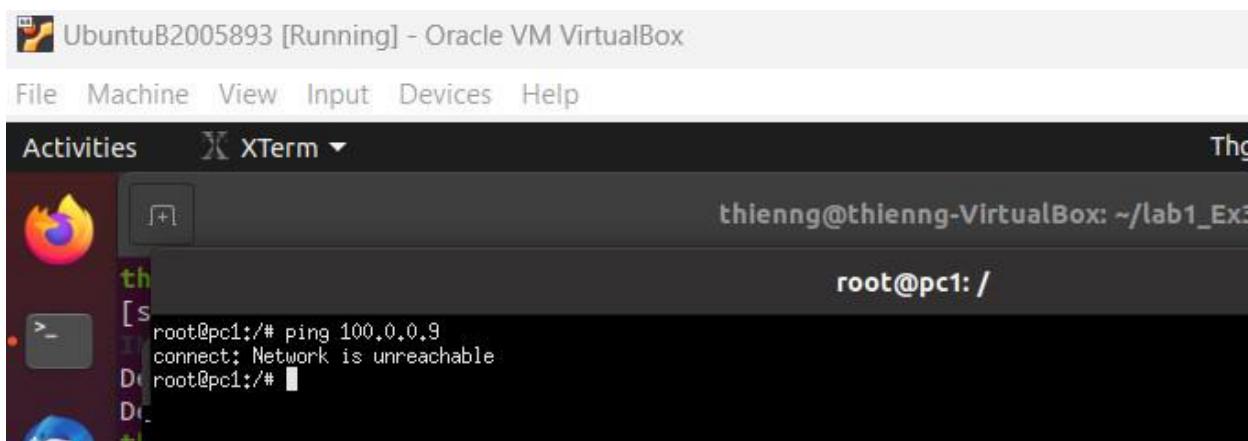


```
thienng@thienng-VirtualBox: ~/lab1_Ex3
root@pc1: /
[sudo] password for root:
[...]
--- Startup Commands Log
D++ ifconfig eth0 195.11.14.5/24 up
D-- --- End Startup Commands Log
[thienng@thienng-VirtualBox ~]$ ping 195.11.14.1
PING 195.11.14.1 (195.11.14.1) 56(84) bytes of data.
64 bytes from 195.11.14.1: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 195.11.14.1: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 195.11.14.1: icmp_seq=3 ttl=64 time=0.053 ms
^C
--- 195.11.14.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 61ms
rtt min/avg/max/mdev = 0.053/0.062/0.080/0.012 ms
root@pc1: #
```

5. Test connectivity (2) – from pc1 to r1

```
$ ping 100.0.0.9
```

It is unreachable because the specified domain either does not exist or could not be contacted.



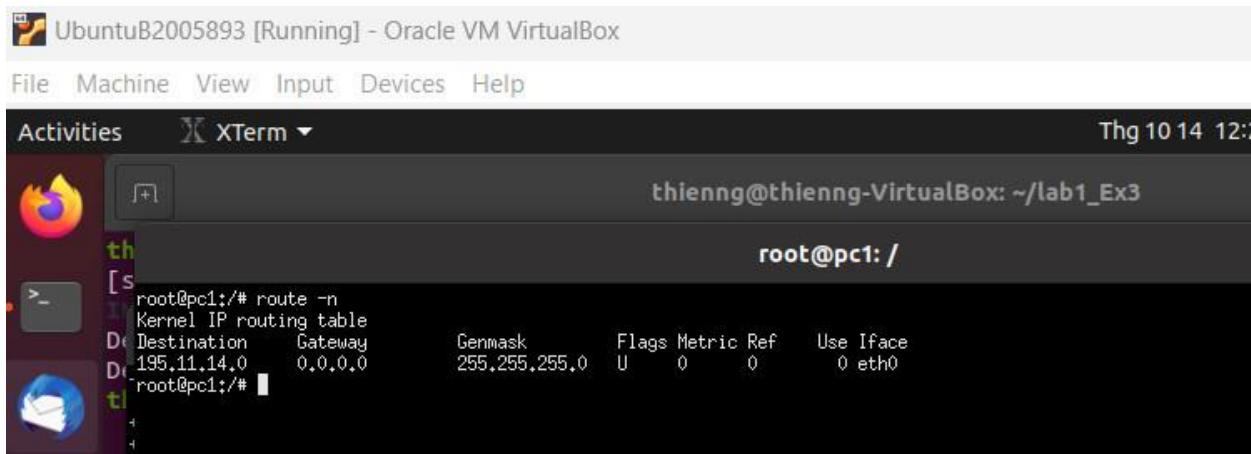
```
thienng@thienng-VirtualBox: ~/lab1_Ex3
root@pc1: /
[sudo] password for root:
[...]
root@pc1: # ping 100.0.0.9
connect: Network is unreachable
root@pc1: #
```

6. Solution to make pc1 can reach to r1

a. Inspecting routing table

From pc1:

```
# route -n
```

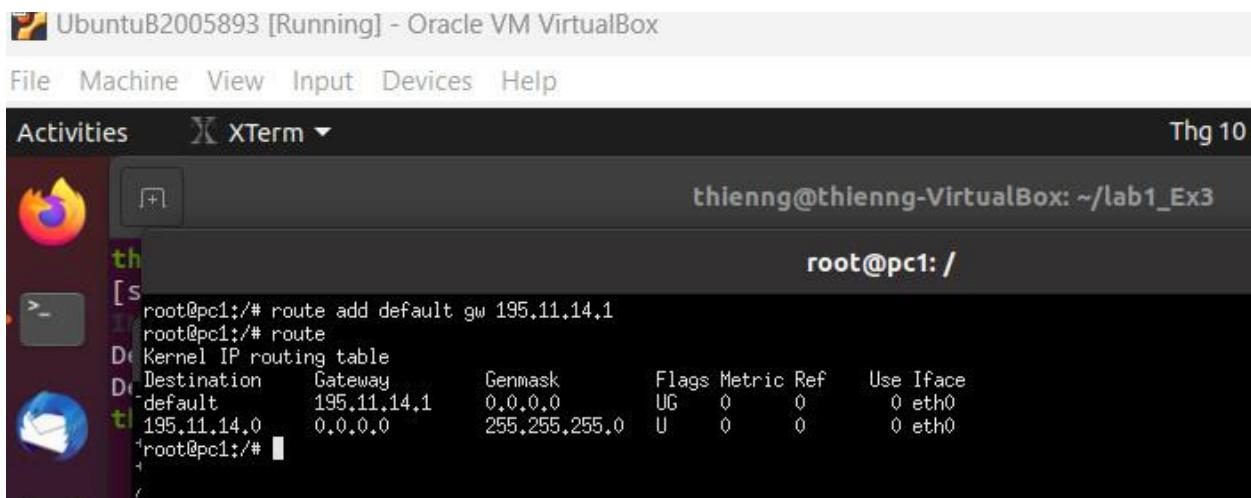


```
root@pc1:/# route -n
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref    Use Iface
195.11.14.0    0.0.0.0    255.255.255.0 U     0      0        0 eth0
root@pc1:/#
```

b. Defaute routes on PC1

```
# route add default gw 195.11.14.1
```

```
# route
```



```
root@pc1:/# route add default gw 195.11.14.1
root@pc1:/# route
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref    Use Iface
default         195.11.14.1  0.0.0.0    UG     0      0        0 eth0
195.11.14.0    0.0.0.0    255.255.255.0 U     0      0        0 eth0
root@pc1:/#
```

7. Now you can ping with 100.0.0.9

```

root@pc1:/# ping 195.11.14.1
PING 195.11.14.1 (195.11.14.1) 56(84) bytes of data.
64 bytes from 195.11.14.1: icmp_seq=1 ttl=64 time=0.071 ms
64 bytes from 195.11.14.1: icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from 195.11.14.1: icmp_seq=3 ttl=64 time=0.050 ms
^C
--- 195.11.14.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 31ms
rtt min/avg/max/mdev = 0.050/0.057/0.071/0.009 ms
root@pc1:/# ping 100.0.0.9
connect: Network is unreachable
root@pc1:/# route -n
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref  Use Iface
0.0.0.0          195.11.14.1  0.0.0.0      UG        0      0    0 eth0
195.11.14.0     0.0.0.0      255.255.255.0 U         0      0    0 eth0
root@pc1:/# route add default gw 195.11.14.1
root@pc1:/# route -n
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref  Use Iface
0.0.0.0          195.11.14.1  0.0.0.0      UG        0      0    0 eth0
195.11.14.0     0.0.0.0      255.255.255.0 U         0      0    0 eth0
root@pc1:/# ping 100.0.0.9
PING 100.0.0.9 (100.0.0.9) 56(84) bytes of data.
64 bytes from 100.0.0.9: icmp_seq=1 ttl=64 time=0.091 ms
64 bytes from 100.0.0.9: icmp_seq=2 ttl=64 time=0.045 ms
64 bytes from 100.0.0.9: icmp_seq=3 ttl=64 time=0.073 ms
^C
--- 100.0.0.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 51ms
rtt min/avg/max/mdev = 0.045/0.089/0.091/0.021 ms
root@pc1:/#

```

8. Ping 100.0.0.10 is not reach

#ping 100.0.0.10

```

root@pc1:/# ping 100.0.0.10
PING 100.0.0.10 (100.0.0.10) 56(84) bytes of data.
^C
--- 100.0.0.10 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 4ms
root@pc1:/#

```

9. While pinging from pc1 100.0.0.10 sniff on interface eth1 of r2

From r2: # tcpdump -tnni eth1

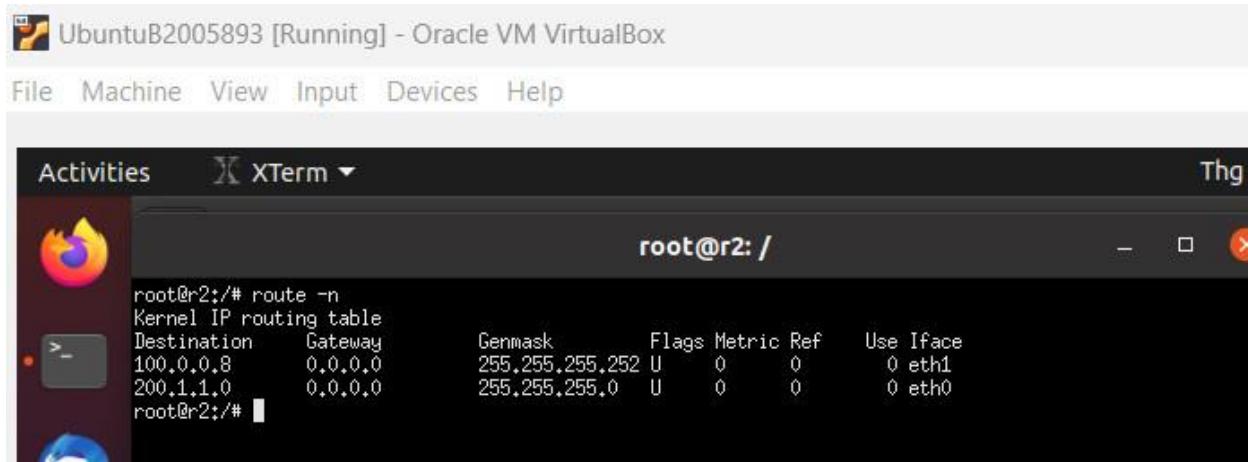
```

root@r2:/# tcpdump -tnni eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
5:a:b6:35:fe:4d:3c > f2:75:e4:15:70:e4, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 100.0.0.10: ICMP echo request, id 7, seq 2, length 64
5:a:b6:35:fe:4d:3c > f2:75:e4:15:70:e4, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 100.0.0.10: ICMP echo request, id 7, seq 3, length 64
5:a:b6:35:fe:4d:3c > f2:75:e4:15:70:e4, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 100.0.0.10: ICMP echo request, id 7, seq 4, length 64
5:a:b6:35:fe:4d:3c > f2:75:e4:15:70:e4, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 100.0.0.10: ICMP echo request, id 7, seq 5, length 64
5:a:b6:35:fe:4d:3c > f2:75:e4:15:70:e4, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 100.0.0.10: ICMP echo request, id 7, seq 6, length 64
5:a:b6:35:fe:4d:3c > f2:75:e4:15:70:e4, ethertype ARP (0x0806), length 42: Request who-has 100.0.0.10 tell 100.0.0.9, length 28
f2:75:e4:15:70:e4 > 5:a:b6:35:fe:4d:3c, ethertype ARP (0x0806), length 42: Reply 100.0.0.10 is-at f2:75:e4:15:70:e4, length 28
^C
7 packets captured
7 packets received by filter
0 packets dropped by kernel
root@r2:/#

```

10. r2's routing table

```
# route -n
```



The screenshot shows a terminal window titled "root@r2: /". The command "route -n" was run, displaying the kernel IP routing table. The output is as follows:

```
root@r2:/# route -n
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref Use Iface
100.0.0.8      0.0.0.0    255.255.255.252 U        0      0      0 eth1
200.1.1.0      0.0.0.0    255.255.255.0   U        0      0      0 eth0
root@r2:/#
```

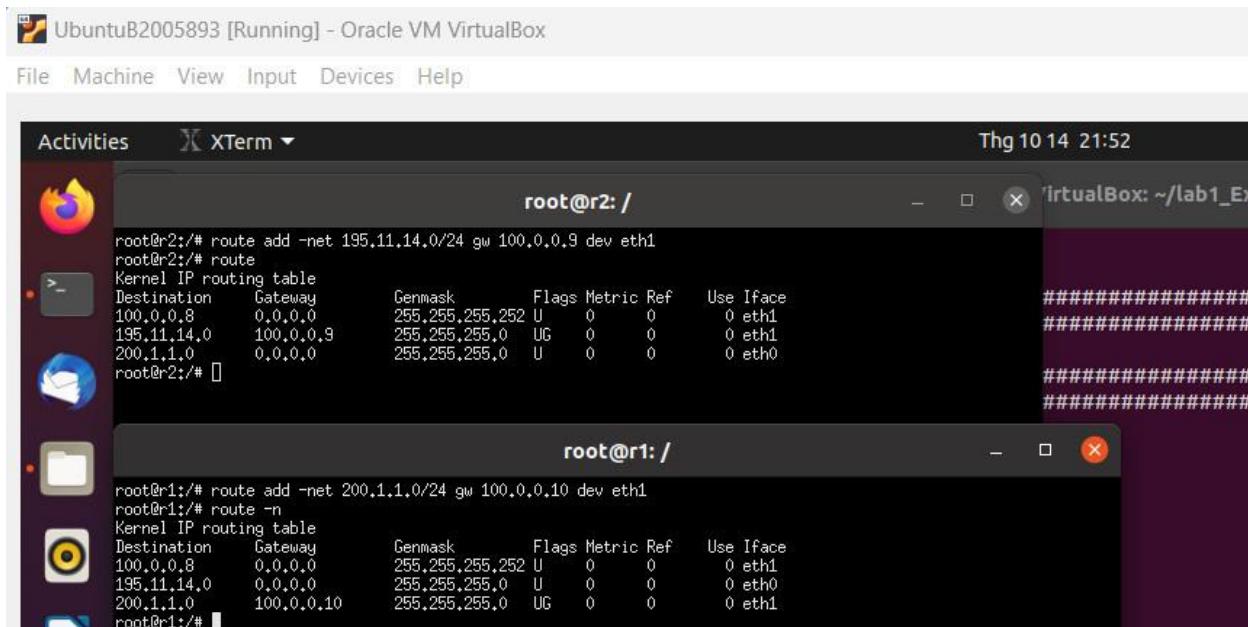
11. Configuring the static route of r1 and r2

a. From r1:

```
# route add -net 200.1.1.0/24 gw 100.0.0.10 dev eth1
# route -n
```

b. From r2:

```
#route add -net 195.11.14.0/24 gw 100.0.0.9 dev eth1
#route -n
```



The screenshot shows two terminal windows. The top window is titled "root@r2: /" and the bottom window is titled "root@r1: /". Both windows show the execution of route configuration commands and the resulting kernel IP routing tables.

In the top window (r2), the command "route add -net 195.11.14.0/24 gw 100.0.0.9 dev eth1" was run, followed by "route -n" to display the table. The output is:

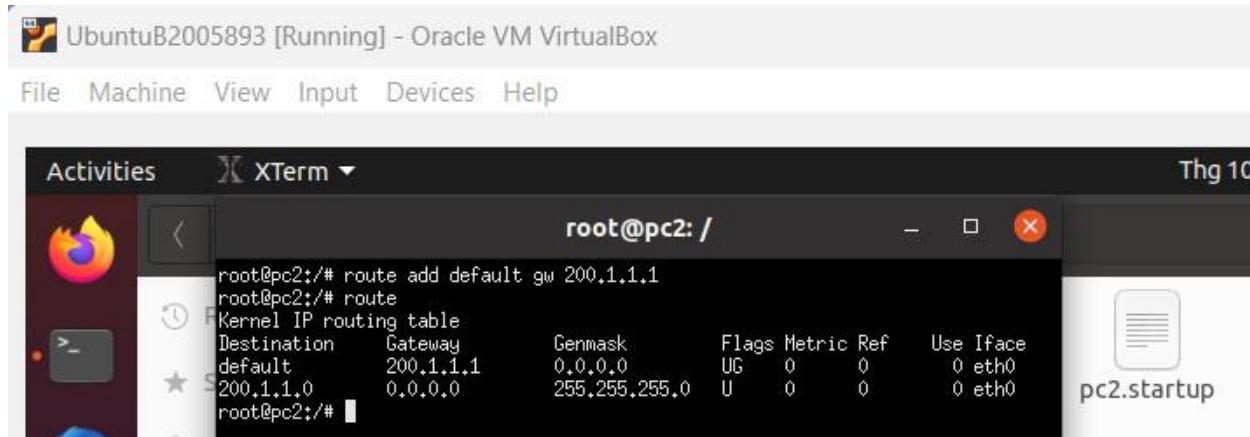
```
root@r2:/# route add -net 195.11.14.0/24 gw 100.0.0.9 dev eth1
root@r2:/# route -n
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref Use Iface
100.0.0.8      0.0.0.0    255.255.255.252 U        0      0      0 eth1
195.11.14.0    100.0.0.9  255.255.255.0   UG       0      0      0 eth1
200.1.1.0      0.0.0.0    255.255.255.0   U        0      0      0 eth0
root@r2:/#
```

In the bottom window (r1), the command "route add -net 200.1.1.0/24 gw 100.0.0.10 dev eth1" was run, followed by "route -n" to display the table. The output is:

```
root@r1:/# route add -net 200.1.1.0/24 gw 100.0.0.10 dev eth1
root@r1:/# route -n
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref Use Iface
100.0.0.8      0.0.0.0    255.255.255.252 U        0      0      0 eth1
195.11.14.0    0.0.0.0    255.255.255.0   UG       0      0      0 eth0
200.1.1.0      100.0.0.10 255.255.255.0   UG       0      0      0 eth1
root@r1:/#
```

12. Set default of pc2

```
# route add default gw 200.1.1.1
```



The screenshot shows the Oracle VM VirtualBox interface with a running Ubuntu 20.04 LTS VM named "UbuntuB2005893". An XTerm window is open, showing the root prompt. The user has run the command `route add default gw 200.1.1.1`. The terminal then displays the kernel's IP routing table:

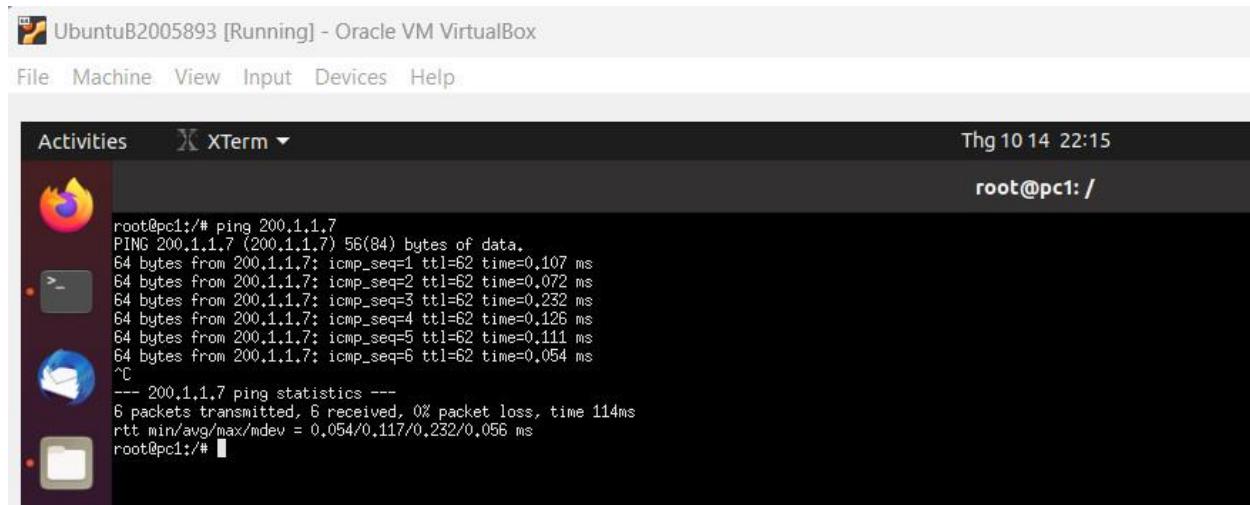
```
root@pc2:/# route
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
default         200.1.1.1      0.0.0.0       UG    0      0        0 eth0
200.1.1.0      0.0.0.0        255.255.255.0 U     0      0        0 eth0
root@pc2:/#
```

13. Test static route

The PCs can now reach each others

From pc1:

```
# ping 200.1.1.7
```

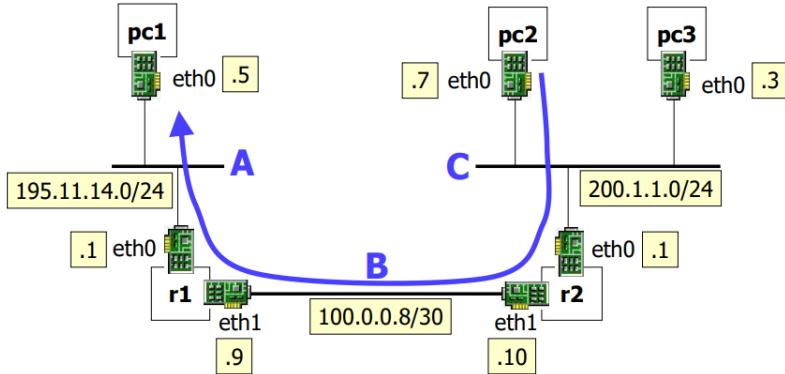


The screenshot shows the Oracle VM VirtualBox interface with a running Ubuntu 20.04 LTS VM named "UbuntuB2005893". An XTerm window is open, showing the root prompt. The user has run the command `ping 200.1.1.7`. The terminal shows the ping statistics:

```
root@pc1:/# ping 200.1.1.7
PING 200.1.1.7 (200.1.1.7) 56(84) bytes of data.
64 bytes from 200.1.1.7: icmp_seq=1 ttl=62 time=0.107 ms
64 bytes from 200.1.1.7: icmp_seq=2 ttl=62 time=0.072 ms
64 bytes from 200.1.1.7: icmp_seq=3 ttl=62 time=0.232 ms
64 bytes from 200.1.1.7: icmp_seq=4 ttl=62 time=0.126 ms
64 bytes from 200.1.1.7: icmp_seq=5 ttl=62 time=0.111 ms
64 bytes from 200.1.1.7: icmp_seq=6 ttl=62 time=0.054 ms
^C
--- 200.1.1.7 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 114ms
rtt min/avg/max/mdev = 0.054/0.117/0.232/0.066 ms
root@pc1:/#
```

14. Delete all vms with \$ sudo kathara wipe

Exercise 4: Study arp protocol



1. Lab configuration

```

$ mkdir pc1 pc2 pc3 r1 r2
$ touch lab.conf pc1.startup pc2.startup pc3.startup r1.startup r2.startup
$ cat pc1.startup
$ cat pc2.startup
$ cat pc3.startup
$ cat r1.startup
$ cat r2.startup
  
```

```

Ubuntu82005893 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Thu 10 15 19:45
thienng@thienng-VirtualBox:~/lab1_Ex4$ ls
lab.conf  pc1  pc1.startup  pc2  pc2.startup  pc3  pc3.startup  r1  r1.startup  r2  r2.startup  router1  router2  shared
thienng@thienng-VirtualBox:~/lab1_Ex4$ cat lab.conf
r1[0]=A
r1[1]=B
r2[0]=C
r2[1]=B
pc1[0]=A
pc2[0]=C
pc3[0]=C
thienng@thienng-VirtualBox:~/lab1_Ex4$ cat pc1.startup
ifconfig eth0 195.11.14.5 up
route add default gw 195.11.14.1
thienng@thienng-VirtualBox:~/lab1_Ex4$ cat pc2.startup
ifconfig eth0 200.1.1.7 up
route add default gw 200.1.1.1
thienng@thienng-VirtualBox:~/lab1_Ex4$ cat pc3.startup
ifconfig eth0 200.1.1.3 up
route add default gw 200.1.1.1
thienng@thienng-VirtualBox:~/lab1_Ex4$ cat r1.startup
ifconfig eth0 195.11.14.1 up
ifconfig eth1 100.0.0.9 netmask 255.255.255.252 broadcast 100.0.0.11 up
route add -net 195.11.14.0 netmask 255.255.255.0 gw 100.0.0.10 dev eth1
thienng@thienng-VirtualBox:~/lab1_Ex4$ cat r2.startup
ifconfig eth0 200.1.1.1 up
ifconfig eth1 100.0.0.10 netmask 255.255.255.252 broadcast 100.0.0.11 up
route add -net 195.11.14.0 netmask 255.255.255.0 gw 100.0.0.9 dev eth1
thienng@thienng-VirtualBox:~/lab1_Ex4$ 
  
```

2. Start kathara and inspecting the arp cache (LOCAL TRAFFIC)

a. From pc3:

```
# arp  
# ping 200.1.1.7  
# arp -n
```

b. From pc2:

```
# arp -n
```

UbuntuB2005893 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities XTerm ▾

root@pc3: /

```
root@pc3:/# arp  
root@pc3:/# ping 200.1.1.7  
PING 200.1.1.7 (200.1.1.7) 56(84) bytes of data.  
64 bytes from 200.1.1.7: icmp_seq=1 ttl=64 time=0.099 ms  
64 bytes from 200.1.1.7: icmp_seq=2 ttl=64 time=0.073 ms  
^C  
--- 200.1.1.7 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 4ms  
rtt min/avg/max/mdev = 0.073/0.086/0.099/0.013 ms  
root@pc3:/# arp -n  
Address          Hwtype  Hwaddress      Flags Mask      Iface  
200.1.1.7        ether   92:1c:69:12:0b:ba  C          eth0  
root@pc3:/# 
```

root@pc2: /

```
root@pc2:/# arp -n  
Address          Hwtype  Hwaddress      Flags Mask      Iface  
200.1.1.3        ether   62:0f:93:ab:15:7a  C          eth0  
root@pc2:/# 
```

3. Inspecting the arp cache (NON LOCAL TRAFFIC)

a. From pc2:

```
# ping 195.11.14.5
```

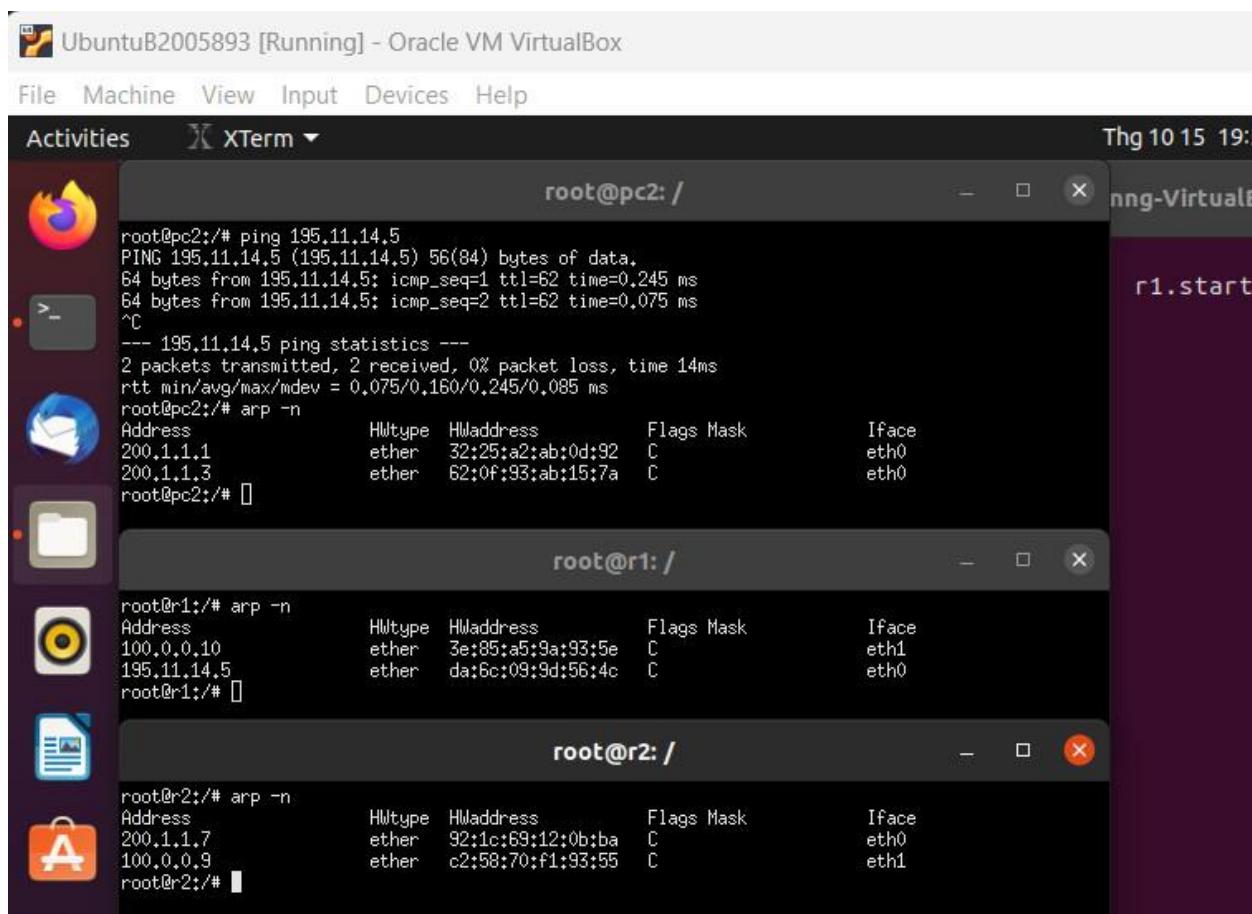
```
# arp -n
```

b. From r1:

```
# arp -n
```

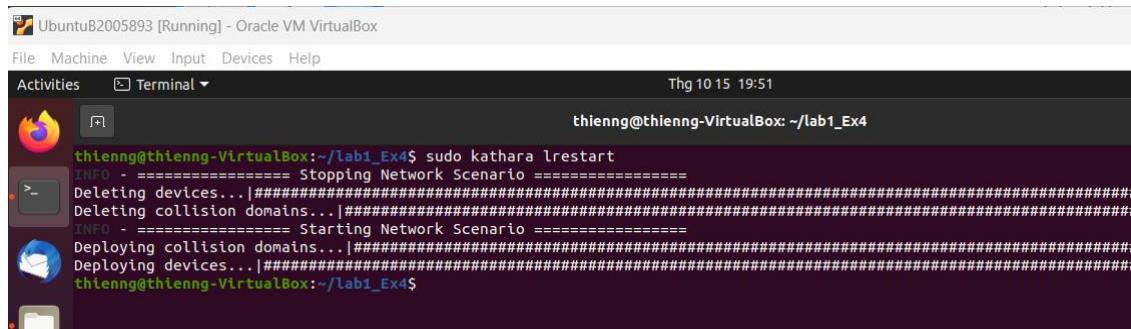
c. From r2:

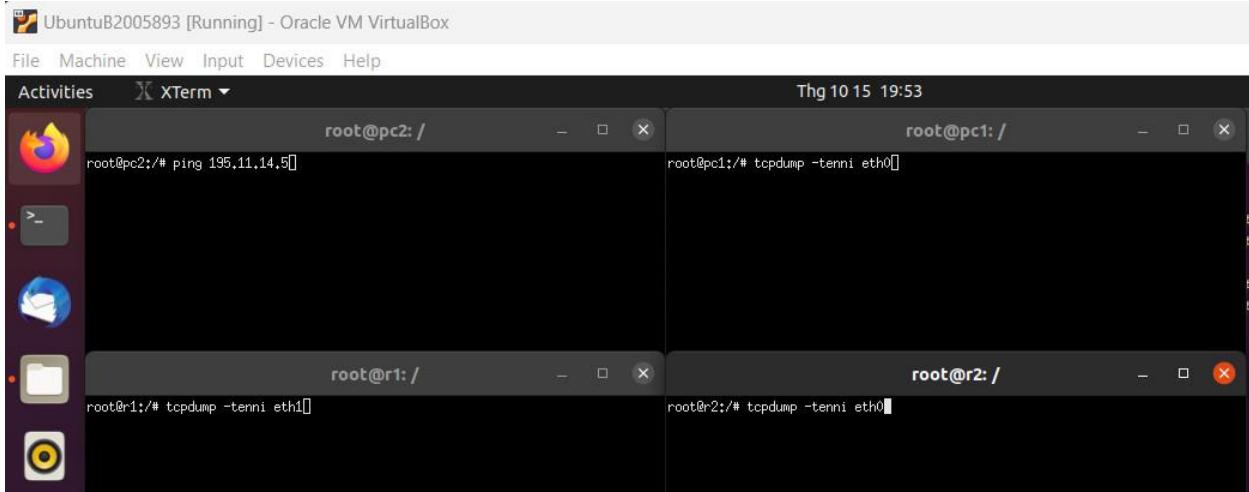
```
# arp -n
```



4. Restart the lab and prepare to sniff

```
$ sudo kathara restart
```





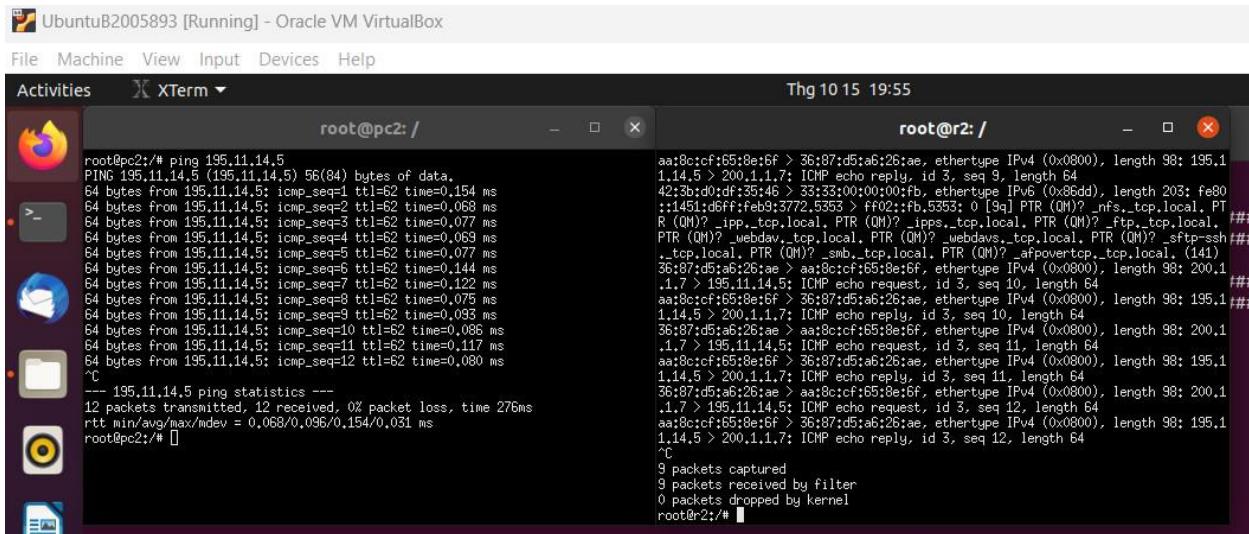
5. Sniffing arp traffic

a. On collision domain C

While pinging from pc2 195.11.14.5 sniff on interface eth0 of r2

From pc2: # ping 195.11.14.5

From r2: # tcpdump -tenni eth0



b. On collision domain B

While pinging from pc2 195.11.14.5 sniff on interface eth1 of r1

From pc2: # ping 195.11.14.5

From r1: # tcpdump -telli eth1

```
root@r1:/# tcpdump -telli eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
ee:a6:4a:90:b2:28 > 36:2b:ea:be:d8:90, ethertype IPv4 (0x0800), length 98: 200.1
..1.7 > 195.11.14.5: ICMP echo request, id 16, seq 4, length 64
36:2b:ea:be:d8:90 > ee:a6:4a:90:b2:28, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 200.1.1.7: ICMP echo reply, id 16, seq 4, length 64
ee:a6:4a:90:b2:28 > 36:2b:ea:be:d8:90, ethertype IPv4 (0x0800), length 98: 200.1
..1.7 > 195.11.14.5: ICMP echo request, id 16, seq 5, length 64
36:2b:ea:be:d8:90 > ee:a6:4a:90:b2:28, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 200.1.1.7: ICMP echo reply, id 16, seq 5, length 64
ee:a6:4a:90:b2:28 > 36:2b:ea:be:d8:90, ethertype IPv4 (0x0800), length 98: 200.1
..1.7 > 195.11.14.5: ICMP echo request, id 16, seq 6, length 64
36:2b:ea:be:d8:90 > ee:a6:4a:90:b2:28, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 200.1.1.7: ICMP echo reply, id 16, seq 6, length 64
36:2b:ea:be:d8:90 > ee:a6:4a:90:b2:28, ethertype ARP (0x0806), length 42: Request who-has 100.0.0.10 tell 100.0.0.9, length 28
ee:a6:4a:90:b2:28 > 36:2b:ea:be:d8:90, ethertype ARP (0x0806), length 42: Reply 100.0.0.10 is-at ee:a6:4a:90:b2:28, length 28
82:3e:55:53:3d:70 > 33:32:00:00:fb, ethertype IPv6 (0x86dd), length 203: fe80::803e:55ff:fe53:3d70.5353 > ff02::fb.5353: 0 [0q] PTR (QM)? _nfs._tcp.local. PTR (QM)? _http._tcp.local. PTR (QM)? _www._tcp.local. PTR (QM)? _smb._tcp.local. PTR (QM)? _afpovertcp._tcp.local. PTR (QM)? _sftp-ssh._tcp.local. PTR (QM)? _lsmb._tcp.local. PTR (QM)? _afpovertcp._tcp.local. (141)
^C
9 packets captured
9 packets received by filter
0 packets dropped by kernel
root@r1:/#
```

c. On collision domain A

While pinging from pc2 195.11.14.5 sniff on interface eth0 of pc1

From pc2: # ping 195.11.14.5

From pc1: # tcpdump -telli eth0

```
root@pc1:/# tcpdump -telli eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
4e:4b:b5:9e:f6:fb > 7a:5b:52:59:92:04, ethertype IPv4 (0x0800), length 98: 200.1
..1.7 > 195.11.14.5: ICMP echo request, id 17, seq 3, length 64
7a:5b:52:59:92:04 > 4e:4b:b5:9e:f6:fb, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 200.1.1.7: ICMP echo reply, id 17, seq 3, length 64
4e:4b:b5:9e:f6:fb > 7a:5b:52:59:92:04, ethertype IPv4 (0x0800), length 98: 200.1
..1.7 > 195.11.14.5: ICMP echo request, id 17, seq 4, length 64
7a:5b:52:59:92:04 > 4e:4b:b5:9e:f6:fb, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 200.1.1.7: ICMP echo reply, id 17, seq 4, length 64
4e:4b:b5:9e:f6:fb > 7a:5b:52:59:92:04, ethertype IPv4 (0x0800), length 98: 200.1
..1.7 > 195.11.14.5: ICMP echo request, id 17, seq 5, length 64
7a:5b:52:59:92:04 > 4e:4b:b5:9e:f6:fb, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 200.1.1.7: ICMP echo reply, id 17, seq 5, length 64
4e:4b:b5:9e:f6:fb > 7a:5b:52:59:92:04, ethertype IPv4 (0x0800), length 98: 200.1
..1.7 > 195.11.14.5: ICMP echo request, id 17, seq 6, length 64
7a:5b:52:59:92:04 > 4e:4b:b5:9e:f6:fb, ethertype IPv4 (0x0800), length 98: 195.1
1.14.5 > 200.1.1.7: ICMP echo reply, id 17, seq 6, length 64
7a:5b:52:59:92:04 > 4e:4b:b5:9e:f6:fb, ethertype ARP (0x0806), length 42: Request who-has 195.11.14.1 tell 195.11.14.5, length 28
4e:4b:b5:9e:f6:fb > 7a:5b:52:59:92:04, ethertype ARP (0x0806), length 42: Request who-has 195.11.14.5 tell 195.11.14.1, length 28
7a:5b:52:59:92:04 > 4e:4b:b5:9e:f6:fb, ethertype ARP (0x0806), length 42: Reply 195.11.14.5 is-at 7a:5b:52:59:92:04, length 28
4e:4b:b5:9e:f6:fb > 7a:5b:52:59:92:04, ethertype ARP (0x0806), length 42: Reply 195.11.14.5 is-at 4e:4b:b5:9e:f6:fb, length 28
^C
12 packets captured
12 packets received by filter
0 packets dropped by kernel
root@pc1:/#
```

6. Arp implementation details

From r2:

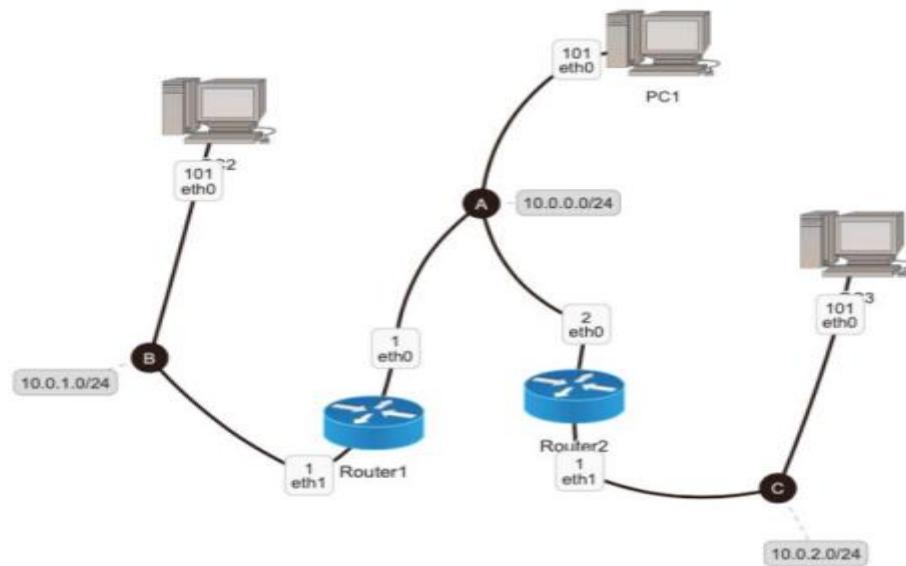
```
# tcpdump -telli eth0
```

```
root@r2:/# tcptrace -t eth0
tcpdump, verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
[...]
root@r2:/#
```

7. Delete all vms with \$ sudo kathara wipe

A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal title is "thienng@thienng-VirtualBox: ~/lab1_Ex4". The command "sudo kathara wipe" is being typed, followed by a confirmation prompt "Are you sure to wipe Kathara? (y/n) y". The terminal window has a dark purple background and includes standard desktop icons like Home, File Manager, and Dash.

Exercise 5: Construct the following network



1. Lab configuration

UbuntuB2005893 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Thg 10 15 14:52

```

thienng@thienng-VirtualBox:~/lab1_Ex5$ ls
lab.conf  pc1  pc1.startup  pc2  pc2.startup  pc3  pc3.startup  router1  router1.startup
router2  router2.startup  shared
thienng@thienng-VirtualBox:~/lab1_Ex5$ cat lab.conf
pc1[0]=A
pc2[0]=B
pc3[0]=C
router1[0]=A
router1[1]=B
router2[0]=A
router2[1]=C
thienng@thienng-VirtualBox:~/lab1_Ex5$ cat router1.startup
ifconfig eth0 10.0.0.1/24 up
ifconfig eth1 10.0.1.1/24 up
route add -net 10.0.2.0/24 gw 10.0.0.2
thienng@thienng-VirtualBox:~/lab1_Ex5$ cat router2.startup
ifconfig eth0 10.0.0.2/24 up
ifconfig eth1 10.0.2.1/24 up
route add -net 10.0.1.0/24 gw 10.0.0.1
thienng@thienng-VirtualBox:~/lab1_Ex5$ cat pc1.startup
ifconfig eth0 10.0.0.101/24 up
route add -net 10.0.1.0/24 gw 10.0.0.1
route add -net 10.0.2.0/24 gw 10.0.0.2
thienng@thienng-VirtualBox:~/lab1_Ex5$ cat pc2.startup
ifconfig eth0 10.0.1.101/24 up
route add default gw 10.0.1.1
thienng@thienng-VirtualBox:~/lab1_Ex5$ cat pc3.startup
ifconfig eth0 10.0.2.101/24 up
route add default gw 10.0.2.1
thienng@thienng-VirtualBox:~/lab1_Ex5$ 
```

2. Route and ping to check

The screenshot shows three terminal windows in a Linux desktop environment:

- root@pc1:/**: Shows the output of `route -n` and `ping 10.0.1.101`. It lists routes to 10.0.0.0, 10.0.1.0, and 10.0.2.0 via eth0. Pings to 10.0.1.101 show 0% loss.
- root@pc2:/**: Shows the output of `route -n` and `ping 10.0.2.101`. It lists routes to 0.0.0.0, 10.0.1.0, and 10.0.2.0 via eth0. Pings to 10.0.2.101 show 0% loss.
- root@pc3:/**: Shows the output of `route -n` and `ping 10.0.1.101`. It lists routes to 0.0.0.0, 10.0.2.0, and 10.0.0.0 via eth0. Pings to 10.0.1.101 show 0% loss.

3. Route the routers

The screenshot shows two terminal windows in a Linux desktop environment:

- root@router1:/**: Shows the output of `route -n`. It lists routes to 10.0.0.0, 10.0.1.0, and 10.0.2.0 via eth0.
- root@router2:/**: Shows the output of `route -n`. It lists routes to 10.0.0.0, 10.0.1.0, and 10.0.2.0 via eth0.

4. Already install wireshark

The screenshot shows a terminal window in a Linux desktop environment:

```

thienng@thienng-VirtualBox:~/lab1_Ex5$ cd
thienng@thienng-VirtualBox:~$ sudo apt install wireshark
[sudo] password for thienng:
Sorry, try again.
[sudo] password for thienng:
Reading package lists... Done
Building dependency tree
Reading state information... Done
wireshark is already the newest version (3.2.3-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
thienng@thienng-VirtualBox:~$
```

5. *Tcpdump*

On pc2, type:

```
# tcpdump -s 1536 -w /hostlab/Ex5_pc2.pcap
```

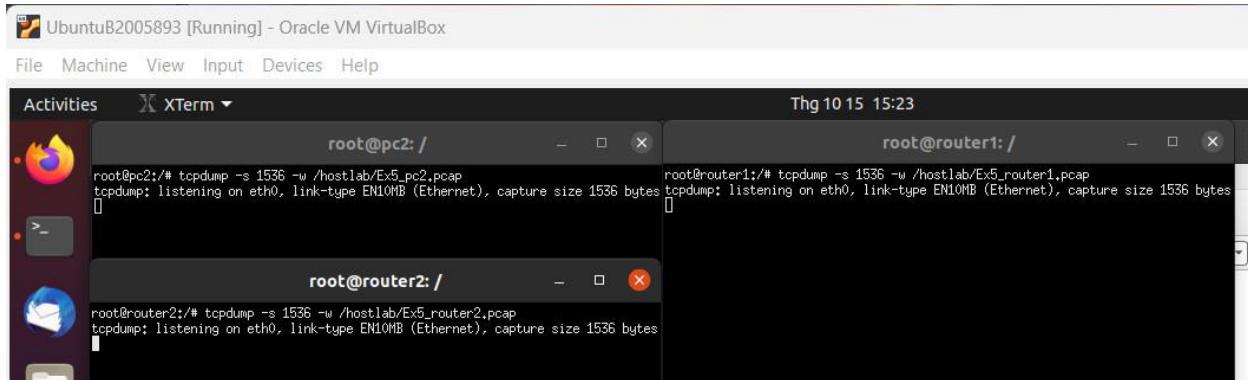
On router1, type:

```
# tcpdump -s 1536 -w /hostlab/BT5_router1.pcap
```

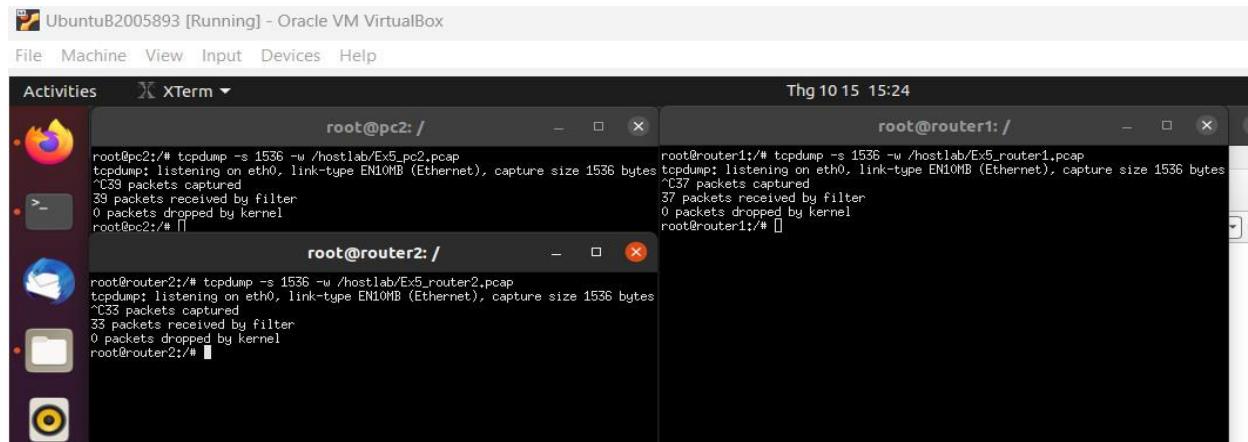
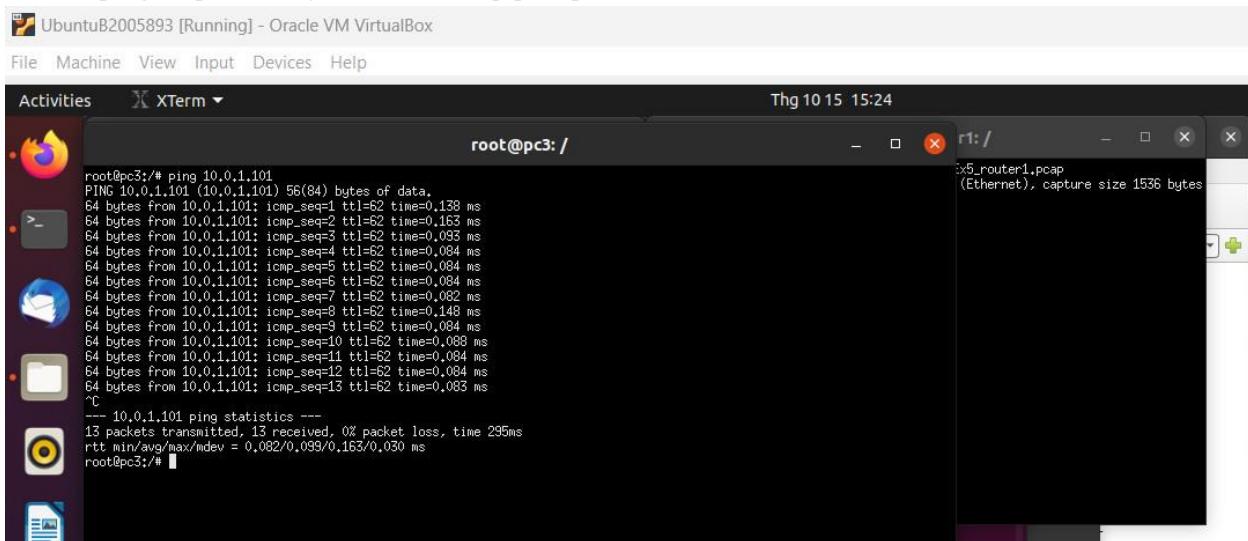
On router2, type:

```
# tcpdump -s 1536 -w /hostlab/BT5_router2.pcap
```

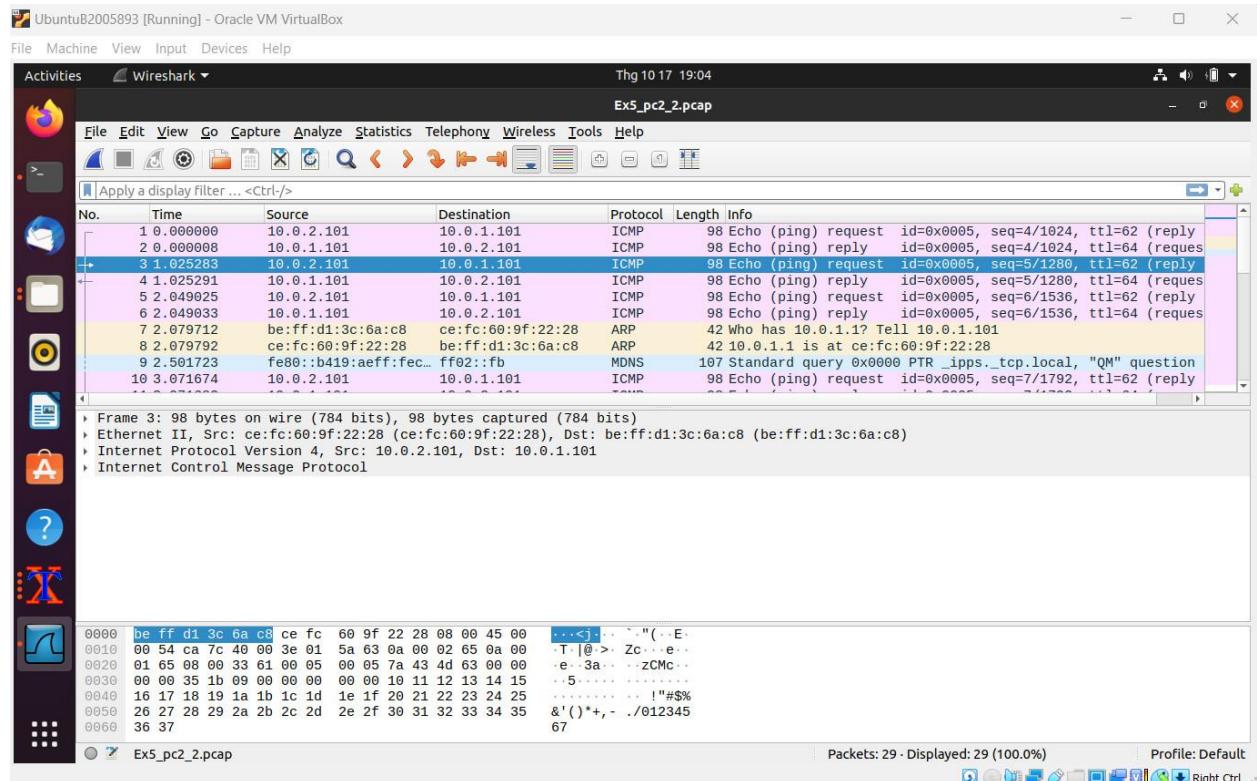
→ All packets are save in .pcap files which are in the /shared folder



6. *ping on pc3, wait for 10s then stop pc3, pc2, router1, route2*



7. Open pc2.pcap on wireshark

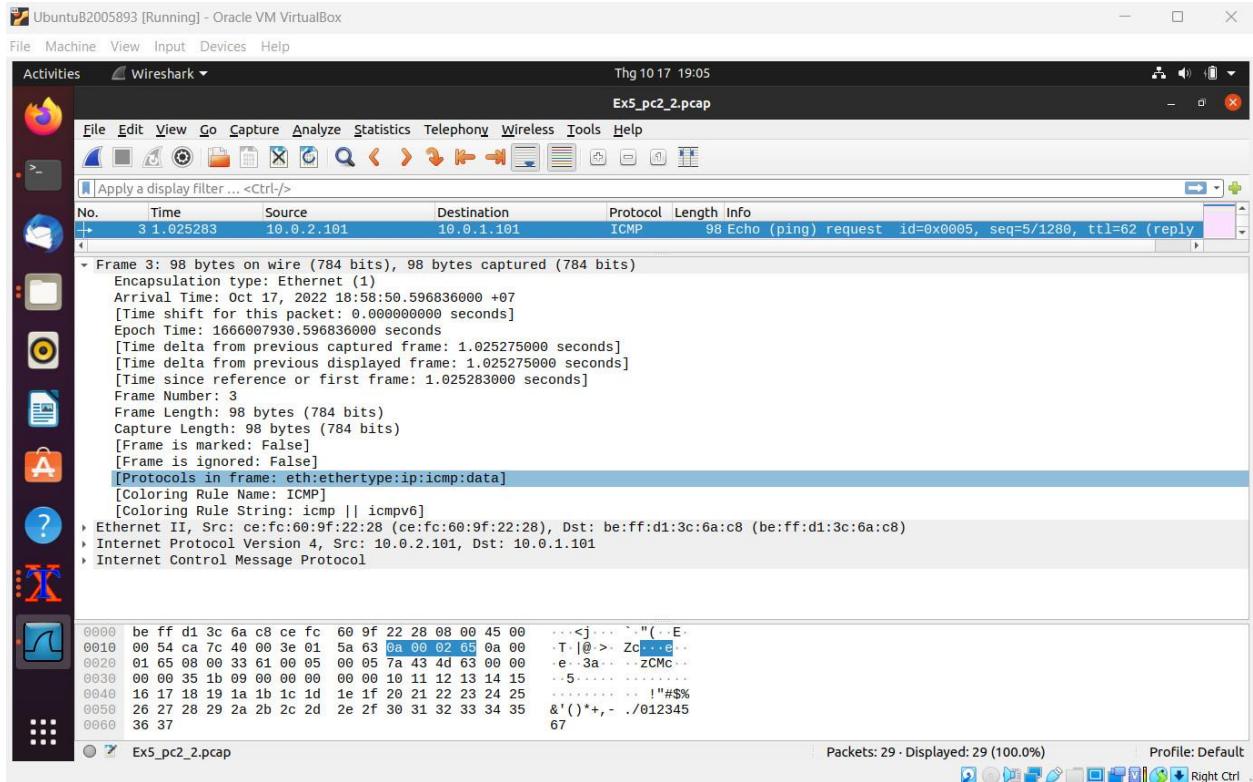


QUESTIONS

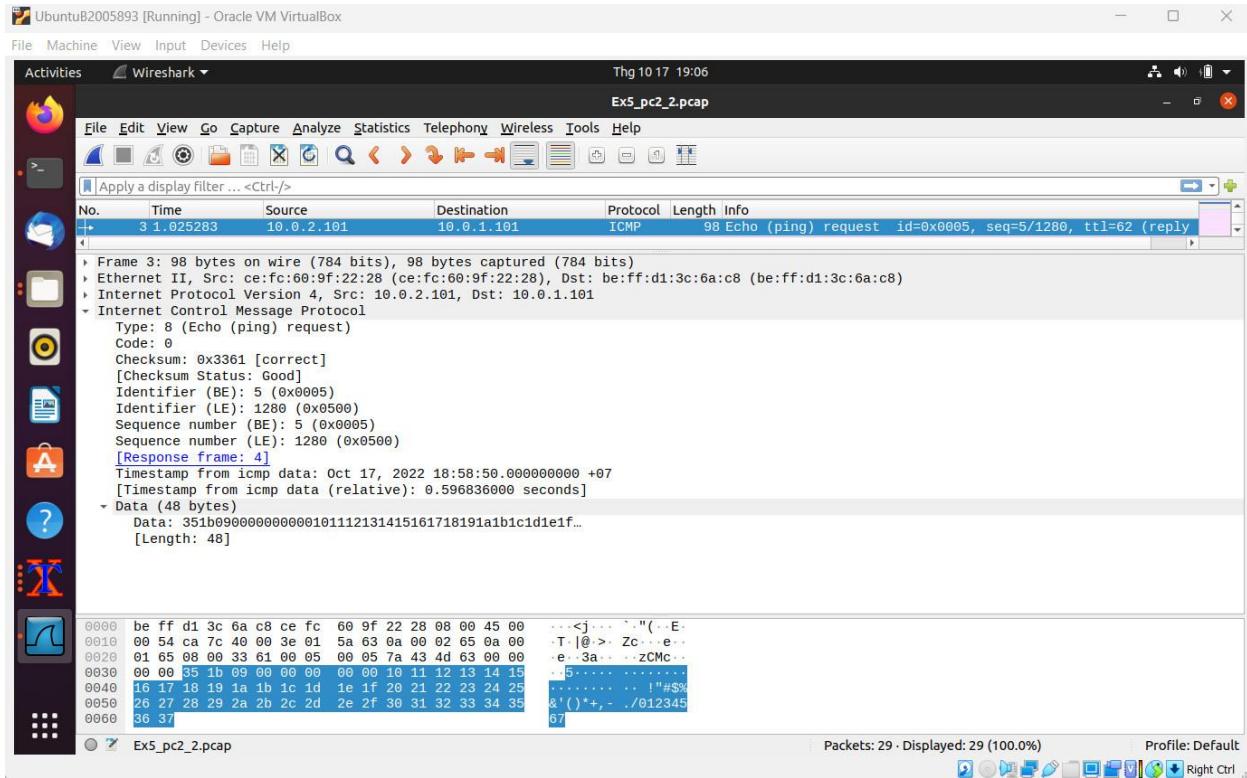
Reuse the network of Exercise 5 and answer questions. Also select the third frame of ICMP

1. Size of frame in bytes?

→ 98 bytes

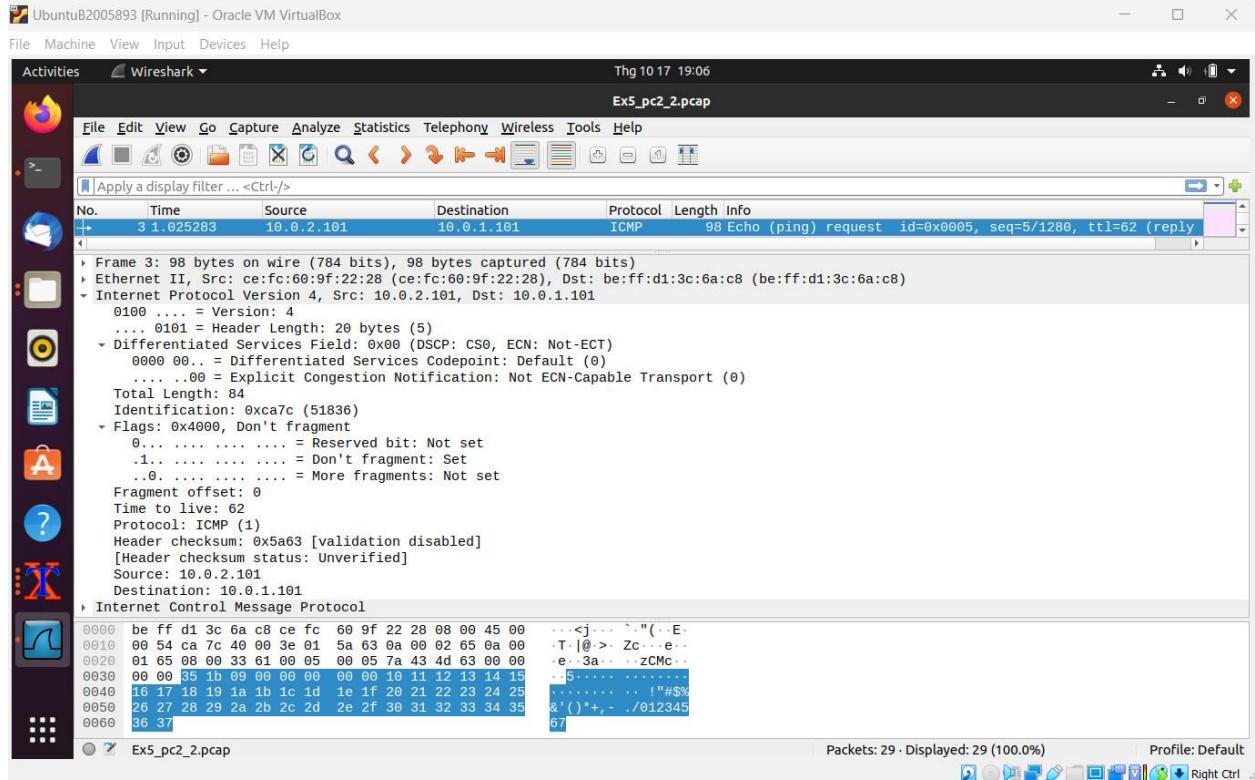


2. Select Header Internet Control Message Protocol → which protocol is using? On which layer of the OSI model does this protocol operate? What is the content of the message? How long is this message in bytes?



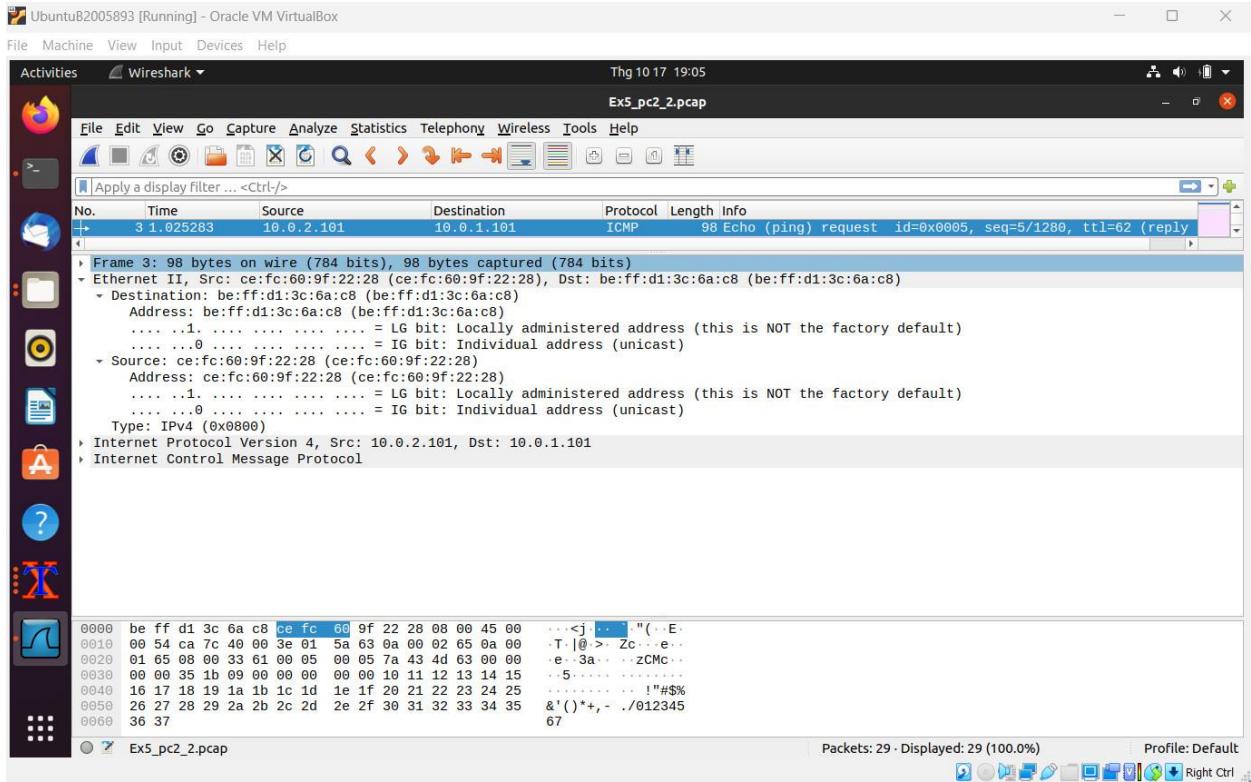
- It is using the ICMP protocol.
- ICMP (Internet Control Message Protocol) is located at the Network layer(layer3) of the OSI model
- The message contains (Time, Source, Destination, Protocol, Length and Info) of the frame
- This message is 48 bytes

3. Select Header Internet Protocol Version 4



- What are the IP addresses of the source and destination hosts?
 - ➔ Source IP address: 10.0.2.101
 - ➔ Destination IP address: 10.0.1.101
- What is the length of the IP packet header? What fields does the Header include? How long is each field (Bytes)?
 - ➔ IP packet header is 20 bytes length
 - ➔ The Header include the source ip, destination ip, differentiated services field and flags
 - ➔ Each field is 5 bytes
- What is the length of the Total Length field (Bytes)?
 - ➔ The total length is 84 bytes

4. Select Header Ethernet II



a. What are the MAC addresses of the source and the destination hosts?

- ➔ The MAC address of source is ce:fc:60:9f:22:28
 - ➔ The MAC address of destination is be:ff:d1:3c:6a:c8
- b. What is the Type value?

- ➔ The Type value is 0x0800