COM-407: TCP/IP NETWORKING

LAB EXERCISES (TP) 1
INTRODUCTION TO MININET
With Solutions

September 26th, 2024
Deadline: October 9, 2024 at 23.55 PM

Abstract

In this TP you will use Mininet, a virtualized environment used to emulate hosts and switches. Mininet is already installed on the virtual machine created in the previous lab. You will perform basic network configuration tasks and learn to mount a man-in-the-middle attack. For a bonus, you will have the chance to evaluate how the performance of Mininet scales with the number of emulated hosts and routers.

1 BACKGROUND KNOWLEDGE: IPERF

Iperf is a widely used tool for network performance measurement and tuning by measuring the maximum network throughput a server can handle. Iperf has both client and server functionalities, and can create data streams to measure the throughput between the two ends in one or both directions. The data streams can be either TCP or UDP.

1.1 TCP CLIENT AND SERVER

In order to launch an iperf TCP server use:

```
$ iperf -s
```

In order to launch an iperf TCP client use:

```
$ iperf -c iperf_server_ip_address -p port_number
```

1.2 UDP Client and Server

In order to launch an iperf UDP server use:

```
$ iperf -s -u
```

In order to launch an iperf UDP client use:

```
$ iperf -c iperf_server_ip_address -u
```

There are also publicly available iperf servers; in this lab you will use `ping.online.net`. The server listens on ports 5200 to 5209.

Q1/ Answer **Question 1 to 3 in Lab 1 - Part 1** on Moodle

Solution. *Question 1 : Observing traffic coming from ping.online.net:*

On the host machine: the source IP address is 51.158.1.21 *(IP address of* `ping.online.net`*) and the source port number is* 5001*, the destination address is* 192.168.0.14 *(IP address of the host machine) and the destination port number is* 54057*. On the guest machine: the source IP address is* 51.158.1.21 *(IP address of* `ping.online.net`*) and the port number is* 5001*, the destination address is* 10.0.2.15 *(IP address of the guest machine) and the destination port number is* 45868*.*

Local port numbers and addresses will vary.

Question 2 : We observe that the source IP addresses and port numbers are the same in both cases, whereas the destination IP addresses and port numbers are different.
Question 3 : This difference is caused by a NAT between the guest and the host network.

Students using miniX can run Wireshark without using *sudo* by first using the following commands : First login as admin by using the command below and entering as password *iewadmin*

```
$ sudo -s
```

and then execute

```
$ sudo dpkg-reconfigure wireshark-common
$ sudo adduser lca2 wireshark
```

Finally, logout and re-login to apply the changes. You should know be able to run wireshark without being admin. For questions about MiniX, ask the TAs.

2 Mininet: A Virtualized Environment for Networking

Mininet "creates a realistic virtual network, running real kernel, switch and application code, on a single machine...". It is a network emulator that can be used to deploy a full network topology on your computer.

It runs a collection of hosts, switches, and links on a single Linux kernel. For this, it uses lightweight virtualization to make a single system look like a computer network, running the same kernel and user code.

In this lab, we prefer Mininet to other emulation platforms such as GNS3 due to its light-weight and scalability features.

2.1 MININET TUTORIAL: COMMAND LINE INTERFACE

Here is a short tutorial on Mininet. The quickest way to familiarize with Mininet is through its command line interface. The command line interface is used to view the configuration of hosts and switches, and run applications or test on the configured network. It cannot be used to modify the topology, i.e., add or remove nodes or links. To start Mininet, enter:

```
$ sudo mn
```

It starts with the default `minimal` topology that comprises 2 hosts, 1 switch and a software-defined-networking controller (more on this in Lab 4). Alternatively, you could start mininet with one of the standard topologies using the `topo` parameter: `sudo mn --topo=tree`. We will learn about more advanced ways to create and configure networks in Section 2.2. Once Mininet is started, the command prompt changes from `$` to `mininet>`.

At the Mininet prompt, enter `help` to see a list of acceptable commands. Next, we will go through a few important ones.

To see the nodes, enter `nodes`. You will see the following output.

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
```

To see the network connections, enter `net` and to see the information about the nodes, enter `dump`.

Q2/ Answer **Question 1 in Lab 1 - Part 2** on Moodle

Solution. *What are the different links in the network ? There are two links in the network: h1-eth0 to s1-eth1, h2-eth0 to s1-eth2.*

The hosts and switches in Mininet share a common UNIX kernel and file system. As a result, any application or script installed on the Mininet VM is accessible by all the devices. Furthermore, the UNIX command line interface can be used for configuration and running applications. This architecture makes using Mininet very intuitive as we shall see. To execute a command on a particular host or a switch, simply append the UNIX command to the name of the device: `<node/switch> <cmd>`. For instance, to check the list of files in the current directory on host `h1`, use

```
mininet> h1 ls
```

As the hosts share the same filesystem, you can verify that the output for the corresponding command on host `h2` is the same.

Q3/ Answer **Question 2 in Lab 1 - Part 2** on Moodle

Solution. *On h2, what command will you use to check the network interfaces of host h2 ? 'h2 ip addr show' or 'h2 ifconfig'*
Excluding the loopback interface, how many interfaces does h2 have ? h2 has one interface h2-eth0.
The IP addresses of all interfaces by increasing IP are 10.0.0.2 and 127.0.0.1.

Mininet automatically substitutes IP addresses for host names. So, to ping a host `h2` from host `h1`, use:

```
mininet> h1 ping h2
```

Now, in a different terminal, start Wireshark by using the command `sudo wireshark`. In wireshark, choose the interfaces `s1-eth1` and `s1-eth2` and start a capture to see the messages generated by the above ping command.

Q4/ Answer **Question 3 in Lab 1 - Part 2** on Moodle

Solution. *What is the destination IPv4 address of the ICMP echo requests sent from h1 to h2? The destination IPv4 address of the ICMP echo request sent from h1 to h2 is 10.0.0.2*

Besides the traditional ping, Mininet also offers the `pingall` command to check connectivity between all hosts. This is particularly handy in checking if a newly configured network is correctly configured.

To exit, use `mininet> exit`. If Mininet crashes for some reason, you can clean up the topology using the following command.

```
$ sudo mn -c
```

For more information, you can use this walk-through.

2.2 MININET TUTORIAL: PYTHON API

While the command line interface is very useful for configuring a few nodes on-the-fly, it is cumbersome for setting up larger networks. For setting up larger networks, Mininet provides a Python-based API that can be used to create and modify a network with hosts and switches, run tests on the switches and collect results, all with using a single script. Next, we shall see how to get started with this API.

NOTE This tutorial will only cover the Python API needed for Mininet. The commands discussed here are sufficient for understanding and modifying a piece of Python code. This level of understanding is sufficient for Lab 1. You will not be able to write a Python code from scratch based on this tutorial. However, we will visit Python in detail again in Lab 3.

Download the `Lab1` folder from Moodle. In this folder you will find a folder named "scripts"; copy it to your virtual machine shared folder. Here is a sample code for a Mininet network with two hosts interconnected by a switch. You can also find this code in the file `firstNetwork.py` in the folder `/media/lca2/shared/Lab1/scripts`.

The code is annotated with comments for your understanding. Comments in Python starts with `#` (for single line comments) or are enclosed in `"""..."""` (for multi-line comments).

```
#!/usr/bin/python

"""
This example shows how to create a Mininet object and add nodes to
it manually.
"""
#Importing Libraries
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info

#Function definition: This is called from the main function
def firstNetwork():

    #Create an empty network and add nodes to it.
    net = Mininet()
    info( '*** Adding controller\n' )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.0.0.1' )
    h2 = net.addHost( 'h2')

    info( '*** Adding switch\n' )
    s12 = net.addSwitch( 's12' )

    info( '*** Creating links\n' )
    net.addLink( h1, s12 )
    net.addLink( h2, s12 )

    info( '*** Starting network\n')
    net.start()

    #This is used to run commands on the hosts

    info( '*** Starting xterm on hosts\n' )
    h1.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h1 &')
    h2.cmd('xterm -xrm "XTerm.vt100.allowTitleOps: false" -T h2 &')


    info( '*** Running the command line interface\n' )
    CLI( net )

    info( '*** Closing the terminals on the hosts\n' )
    h1.cmd("killall xterm")
    h2.cmd("killall xterm")
```

```
    info( '*** Stopping network' )
    net.stop()

#main Function: This is called when the Python file is run
if __name__ == '__main__':
    setLogLevel( 'info' )
    firstNetwork()
```

Note the two ways of adding a host. For adding `h1`, we specify the name of the host and the ip address in the parameters. For adding `h2`, we specify only the name of the host. When the IP address is not specified, Mininet gives it a default IP Address. In this lab, we will use the latter approach to create a host. Once we create a host, we will configure it through standard Linux commands through the terminal.

To start Mininet with the custom topology, navigate to the directory in which `firstNetwork.py` is present and in the command line enter:

```
$ sudo python3 firstNetwork.py
```

Besides the output on your terminal, you should see two new terminals, one each for the hosts `h1` and `h2`, respectively. From these terminals, you can access `h1` and `h2` just like normal UNIX hosts. For instance, in the terminal for `h2`, enter `sudo wireshark` to open wireshark.

Q5/ Answer **Question 4 in Lab 1 - Part 2** on Moodle

Solution. *What are the interfaces available for capture on wireshark when used on h2 : h2-eth0 and lo are the interfaces available for capture on wireshark on h2*

To close all the terminals and the network, in Mininet, enter:

```
mininet> exit
```

More information on the Python API can be found here.

3 Basic connectivity within a LAN

In this section you will learn how to configure machines in order to achieve connectivity in the IP layer within a Local Area Network (LAN). You will also use *Wireshark* to observe which traffic is sent (or not) in some scenarios. This will help you to find what is configured correctly and what has to be configured additionally in order to achieve the connectivity.

3.1 Wiring and addressing scheme

In Mininet, create a configuration with four hosts PC1, PC2, PC3 and PC4 and three switches s14, s24 and s34 as shown in Figure 1. For this purpose, use the file `lanTopology.py` in the folder `/media/lca2/shared/Lab1/scripts`. Parts of the topology file have already been written for you. Fill in the rest **without specifying the ip addresses of any of the interfaces** (you will do this separately through

commands) and start the network. **Note:** if not specified, the names of the interfaces are given automatically by mininet in the form `<host_name>-ethx`, where `x=0,1,2...` corresponds to the order the interface is defined in the script. Once completed, you should see four terminal windows labeled PC1, PC2, PC3 and PC4. We will use these windows for the remainder of the lab.

During the rest of the course, it might be helpful to have a copy-paste functionality for the mininet hosts. The terminal windows used by mininet are of type `xterm` which does not offer high level of copy-pasting that other terminals have. If you need to copy-paste, you have several options: if you have a mouse, you can copy by simply selecting text from one terminal and paste with the middle click. If you use Windows on your host machine, you can also paste text using `shift+Insert`.

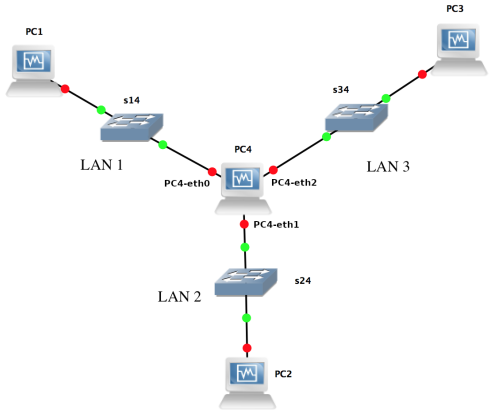

Figure 1: Basic configuration

3.1.1 IPv4 ADDRESSING

For IPv4 addressing, we will use the private IPv4 address space `10.10.0.0/16`.

- For the local network connecting PC4 to PC1 (LAN1), we will use the IPv4 network address prefix `10.10.10.0/24`. The host identifier (the fourth byte) should be `1` to indicate the PC1 and `4` for PC4.
- For the local network connecting PC4 to PC2 (LAN2), we will use the IPv4 network address prefix `10.10.20.0/24`. The host identifier (the fourth byte) should be `2` to indicate the PC2 and `4` for PC4.
- For the local network connecting PC4 to PC3 (LAN3), we will use the IPv4 network address prefix `10.10.30.0/24`. The host identifier (the fourth byte) should be `3` to indicate the PC3 and `4` for PC4.

Q6/ Answer **Question 1 and 2 in Lab 1 - Part 3 (1st part)** on Moodle

Solution. *• Question 1 in Lab 1 - Part 3 (1st part)*
PC4 belongs to 3 IPv4 networks, they are all Local area networks (LANs)
• Question 2 in Lab 1 - Part 3 (1st part)
PC1-eth0: `10.10.10.1`
PC2-eth0: `10.10.20.2`
PC3-eth0: `10.10.30.3`
PC4-eth0: `10.10.10.4`
PC4-eth1: `10.10.20.4`
PC4-eth2: `10.10.30.4`

3.1.2 IPv6 ADDRESSING

Several IPv6 addresses can be obtained by each interface. In the virtual environment we use private addresses, called Unique Local Addresses (ULAs). Such addresses can be used only inside a private domain and are ignored in the public internet. We will use the prefix `fd24:ec43:12ca::/48`, which is registered to EPFL for the SmartGrid project.

- For LAN 1, we will use the IPv6 subnet `fd24:ec43:12ca:c001:10::/80` and addresses `fd24:ec43:12ca:c001:10::X`, with `X` equal to `1` for PC1 and to `4` for PC4.
- For LAN 2, we will use the IPv6 subnet `fd24:ec43:12ca:c001:20::/80` and the addresses `fd24:ec43:12ca:c001:20::X`, with `X` equal to `2` for PC2 and `4` for PC4.
- For LAN 3, we will use the IPv6 subnet `fd24:ec43:12ca:c001:30::/80` and the addresses `fd24:ec43:12ca:c001:30::X`, with `X` equal to `3` for PC3 and `4` for PC4.

Q7/ Answer **Question 3 in Lab 1 - Part 3 (1st part)** on Moodle ***Solution.***

PC1-eth0: `fd24:ec43:12ca:c001:10::1`
PC2-eth0: `fd24:ec43:12ca:c001:20::2`
PC3-eth0: `fd24:ec43:12ca:c001:30::3`
PC4-eth0: `fd24:ec43:12ca:c001:10::4`
PC4-eth1: `fd24:ec43:12ca:c001:20::4`
PC4-eth2: `fd24:ec43:12ca:c001:30::4`

In addition to the Unique Local Addresses, every IPv6 interface also has a link-local address due to the IPv6 Stateless Address Autoconfiguration (SLAAC). Link local addresses are allocated automatically and do not require any configuration; they can be used only for communication in the same LAN.

Link-local addresses are all in the `fe80::/64` subnet. With this addressing scheme, all the network cards in the world are in the same subnet (you can see that the routing table contains by default this subnet)! For this reason, two machines on the same link can communicate directly without the need to configure routing (the host part is unique because it is derived from the MAC address, which is supposed to be unique).

To determine the MAC address for the `PC1-eth0` interface, look at the `link/ether` field after issuing in a terminal the following command:

```
$ ip link show PC1-eth0
```

Q8/ Answer **Question 4 in Lab 1 - Part 3 (1st part)** on Moodle ***Solution.*** *Answer may vary from person to person due to mininet.*

PC1(eth0): `08:00:27:32:76:ae`
PC2(eth0): `08:00:27:ba:76:45`
PC3(eth0): `08:00:27:db:9e:20`
PC4(eth0): `78:31:c1:ce:c5:e0`
PC4(eth1): `08:00:27:7e:b6:e5`
PC4(eth2): `08:00:27:ff:5b:ed`

The SLAAC IPv6 link-local address is formed as follows. The address prefix is `fe80::/64`. For the host part, we use a 64-bit interface identifier in modified EUI-64 format. It is derived from the interface's MAC address by inverting 7th bit and inserting `ff:fe` in the middle. An illustration is given in Figure 2 adapted from the Wikipedia page dedicated to IPv6 addresses.

Q9/ Answer **Question 5 in Lab 1 - Part 3 (1st part)** on Moodle

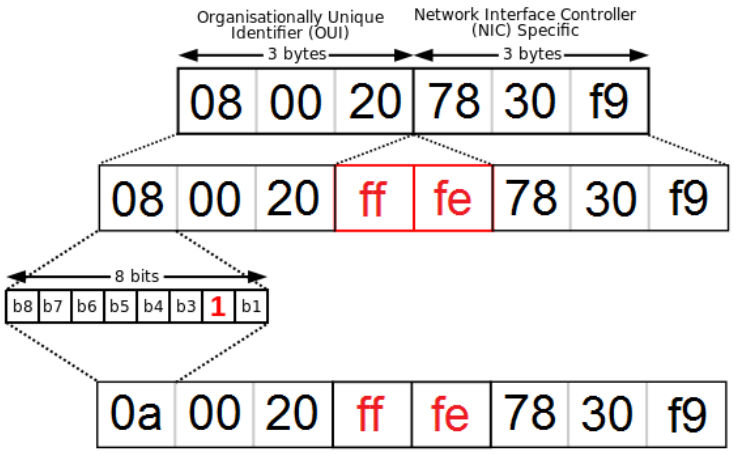

Figure 2: Formation of EUI-64 starting from MAC address of an interface.

Solution. *Here are the possible MAC addresses you had to give the IPv6 link-local address for :*
42:f6:b6:46:59:b5 – fe80::40f6:b6ff:fe46:59b5
c5:4b:7f:7b:5f:c3 – fe80::c74b:7fff:fe7b:5fc3
09:2c:29:f6:aa:c5 – fe80::b2c:29ff:fef6:aac5
6c:fe:15:1a:ea:c9 – fe80::6efe:15ff:fe1a:eac9

3.2 Are you connected? Use Wireshark to see what happens.

In order to have access to the commands that manipulate the network interfaces, you need to be logged in as superuser (root).

To see the network interfaces of a PC, type in its terminal:

```
$ ip link
```

Bring up the interfaces of PC4 by running in PC4's terminal:

```
# ip link set PC4-eth0 up
# ip link set PC4-eth1 up
# ip link set PC4-eth2 up
```

Similarly, bring up the interfaces of PC1, PC2 and PC3.

As seen in Lab0, the `Wireshark` program is used to capture the incomming and outgoing traffic on a network interface. You will now inspect what happens at packet level.

Start a `Wireshark` capture on all interfaces of PC1 and PC4. Since the Wireshark process blocks the command line, you need to run it in detach mode such that command line will not be frozen. For that use:

```
# wireshark &
```

We will first test what happens when pinging unassigned addresses.

To test IPv4 connectivity run a *ping* to the unconfigured router from a terminal on `PC1`:

```
$ ping 10.10.10.4
```

Q10/ Answer **Questions 1 to 4 in Lab 1 - Part 3 (2nd part)** on Moodle

Solution. *• Question 1 in Lab 1 - Part 3 (2nd part)*
`ping` cannot find the destination host since there exists no host with the specified IP address in the same subnet. Therefore, only ARP packets are sent and the ping commands fails.

Test IPv6 connectivity by running a ping *to the link-local IPv6 address of `PC4-eth0` interface (the one that is connected to PC1) from a terminal on PC1. (**Note that MAC addresses of interfaces are changed every time when you start Mininet. Make sure that you use proper MAC addresses in order to get the link-local IPv6 address.**)*

```
$ ping6 <link_local_address_of_PC4_eth0> -I PC1-eth0
```

• Question 2 and 3 in Lab 1 - Part 3 (2nd part)
When you ping PC4-eth0 from PC1 with link-local IPv6, PC1 sends a NS packet to PC4 to resolve the host. PC4 responds with a NA packet to PC1. Finally PC1 pings PC4 sucessfully. In Wireshark we observe the opposite direction PC4 sends a NS packet to PC1 and PC1 responds with a NA packet to PC4. It corresponds to the verification of the neighbor reachability.
In IPv6, link-local addresses are assigned to interfaces by the OS using SLAAC without requiring any configuration, thus the `ping` command works with no problem. In Wireshark we observe echo requests and replies.

Now ping the link-local IPv6 address of `PC4-eth2` interface (the one that is in LAN 3) from a terminal on PC1.

```
$ ping6 <link_local_address_of_PC4_eth2> -I PC1-eth0
```

• Question 4 in Lab 1 - Part 3 (2nd part)

Pinging PC4-eth2 from PC1-eth0 does not work because they are on different subnets/LANs.
Pinging PC4-eth0 from PC1-eth0 works because they are on the same subnet/LAN.
When pinging PC4-eth2, PC1-eth0 does not receive any Neighbor Adertisement as response because PC4 receives the Neighbor solicitation packets on interface PC4-eth0 which is on a different subnet as the interface PC4-eth2.

IPv6 link-local addresses cannot traverse routers thus PC4's `eth2` link-local address is unreachable from PC1. Only devices that are in the same LAN, like PC4's `eth0` link-local address, will be reachable

3.3 CONFIGURING AND TESTING OF THE LAN

In the previous question you observed that there exists only IPv6 connectivity within the LAN. This is a consequence of IPv6 Stateless Address Autoconfiguration. However, these addresses are not globally routable. To obtain full IPv6 connectivity and IPv4 connectivity some additional steps are required.

3.3.1 LAN CONFIGURATION

The first step in order to achieve full connectivity from your machine is to configure its network interface and to assign an IP address to the interface. This is true for both IPv4 and IPv6 as link-local IPv6 addresses are not routable on the Internet.

To visualize the IPv4 and IPv6 routing tables of the machine, on PC2 type:

```
$ ip route
$ ip -6 route
```

Q11/ Answer **Questions 5 to 8 in Lab 1 - Part 3 (2nd part)** on Moodle

Solution. *• Question 5 in Lab 1 - Part 3 (2nd part)*
Complete the observed routing table entries on PC2 for :
ip route: 10.0.0.0/8 dev PC2-eth0 proto kernel scope link src 10.0.0.2
ip -6 route: fe80::/64 dev PC2-eth0 proto kernel metric 256

To configure your network interfaces and assign an IPv4 and an IPv6 address to `PC2-eth0` interface, open a `root` terminal (`su`) and type:

```
# ip addr add 10.10.20.2/24 dev PC2-eth0
# ip -6 addr add fd24:ec43:12ca:c001:20::2/80 dev PC2-eth0
```

Make sure to delete previously existing IP addresses. You may check for those and delete them with commands similar to the following:

```
# ip addr show
# ip addr del <existing_ip_address> dev PC2-eth0
# ip -6 addr del <existing_ip_address> dev PC2-eth0
```

Tip: To avoid typing this again the next time you boot the PCs save the commands in a script file that you can execute later on. Make use of the `shared` folder and the copy-paste functionality.

• Question 6 in Lab 1 - Part 3 (2nd part)
What are the routing table entries after assigning the new IP addresses (v4 and v6) to PC2 ? :
ip route: 10.10.20.0/24 dev PC2-eth0 proto kernel scope link src 10.10.20.2
ip -6 route: fd24:ec43:12ca:c001:20::/80 dev PC2-eth0 proto kernel metric 256.

• Question 7 in Lab 1 - Part 3 (2nd part)
For the IPv4 address assigned to PC2-eth0 : What length, in bits, has the subnet mask in which PC2-eth0 belongs to ?

With the netmask `/24`, the subnetwork is `10.10.20` and the host is represented with the last octet. This information is useful to know if a destination is on the same LAN or not (whether to communicate directly or through the default gateway).

From PC2, run another `ping` to PC4's IPv4 interface:

```
$ ping 10.10.20.4
```

• Question 8 in Lab 1 - Part 3 (2nd part)
What traffic do you observe on wireshark when you ping 10.10.20.4 (PC4-eth1) from PC2-eth0 ?
ARP packets are sent (PC2-eth0 sends an ARP broadcast), but still no ICMP since PC4 is not configured. The ping fails. PC4 will receive the ARP requests from PC2 but won't respond to them (PC4-eth1 receives an ARP broadcast but does not respond to it).

Configure the interfaces of PC4 and the one of PC1 and PC3 according to the addressing scheme.

To verify the configuration, type:

```
$ ip addr show
```

Try again pinging PC4 from PC2. It should work now. Do the same for PC1 and PC3 to verify connectivity to PC4.

You can see the mapping between IPv4 addresses and MAC addresses. Take a look at the ARP table of the PC1 workstation:

```
$ ip neigh
```

At this point you should have both IPv4 and IPv6 connectivity within each LAN.

3.4 ROUTING PACKETS

Try to reach any of the `PC4-eth1` or `PC4-eth2` interfaces of PC4 from PC1 (i.e., anyone that is not directly connected to PC1):

For IPv4:

```
$ ping 10.10.20.4
$ ping 10.10.30.4
```

For IPv6:

```
$ ping fd24:ec43:12ca:c001:20::4
$ ping fd24:ec43:12ca:c001:30::4
```

Q12/ Answer **Question 9 in Lab 1 - Part 3 (2nd part)** on Moodle

Solution. *What do you observe ?*
From PC1-eth0 you can reach/ping PC4-eth0 but not PC4-eth1. This is because PC1 does not have any default gateway configured. No route is available.

On PC1, add an IPv4 default route (i.e., "configure default gateway") and do the same for IPv6:

```
# ip route add default via 10.10.10.4
# ip -6 route add default via fd24:ec43:12ca:c001:10::4
```

Take the `eth2` interface of PC4 and test again if it is reachable from PC1 (both in IPv4 and IPv6).

In PC4 start a `Wireshark` capture of `PC4-eth2`interface to observe the traffic of the link between the machines belonging to LAN3.

From PC1, try to ping PC3, and observe the traffic on PC4:

```
$ ping 10.10.30.3
```

Frustratingly, it does not work. The ICMP messages do not get sent out by PC4. By default, PC4 does not forward IP traffic from one LAN to another. You need in addition to enable IP forwarding on PC4, which allows PC4 to work as router, by typing the following command (if you use copy-paste from this document, most likely the underscore "_" character will not be copied properly):

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Retry to ping PC3 from PC1. It still does not work, but now you know enough to fix it. *Hint:* Check again the traffic on the two links with Wireshark and see which packets do not get sent.

Q13/ Answer **Question 10 to 12 in Lab 1 - Part 3 (2nd part)** on Moodle

Solution. • *Question 10 in Lab 1 - Part 3 (2nd part)*
Which commands did you use to fix the problem ?
A route needs to be added on PC3, for example by configuring PC4 as a default gateway : ip route add default via 10.10.30.4
ip -6 route add default via fd24:ec43:12ca:c001:30::4
We apply those commands to PC3

Now, configure your network so as to be able to ping PC1, PC2 and PC3 between each other both in IPv4 and IPv6. For that, you will need to enable IPv6 forwarding using:

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

• *Question 11 in Lab 1 - Part 3 (2nd part)*
Perform the following pings simultaneously: PC2 from PC1, PC3 from PC2, and PC1 from PC3. What do

you observe about the round-trip times ?
They are roughly the same in all cases.

• *Question 12 in Lab 1 - Part 3 (2nd part)*
Do an iperf test between PC1 and PC2 (PC1 run as server and PC2 as a client) using TCP and UDP. Report the performance (bandwidth values) you obtain with TCP and UDP :
The bandwidth with TCP is 27 *Gbits/sec, with UDP is* 1.05 *Mbits/sec. The port number for server remains the same for TCP and UDP (equals to* 5001*), for the client it changes every time you launch.*

4 CREATING A MAN-IN-THE-MIDDLE ATTACK WITH IP TABLES

In this section you will have an introduction to `iptables`, which will be useful in the next labs. We will work in IPv4 only, and we will use the same working configuration of section 3.4. For your convenience, this configuration can be created directly by running the python script `lanConfig.py`. We will use a man-in-the-middle (MITM) attack example to illustrate iptables, so let's re-label PCs accordingly to their new function (check fig. 3): PC2 is Alice, the naive internet user who is trying to access her bank account; PC1 plays the role of the bank server; PC4 is Alice's home router, which will be hacked to our malicious purposes; and finally PC3 is the attacker's (your) private server where you will redirect Alice's bank transactions and steal her money.

Let's assume that a successful `ping` between two devices is equivalent to a successful money transfer between them. Your goal (as a hacker) is to make Alice and her bank believe they have a successful and point-to-point connection (blue line in fig. 3) while in reality communication goes all the way to your malicious server (PC3) whenever there is a transaction between Alice and the bank (red line in fig. 3). The rule is we can only make configuration changes in Alice's home router and in our own malicious server (PC4 and PC3 respectively).

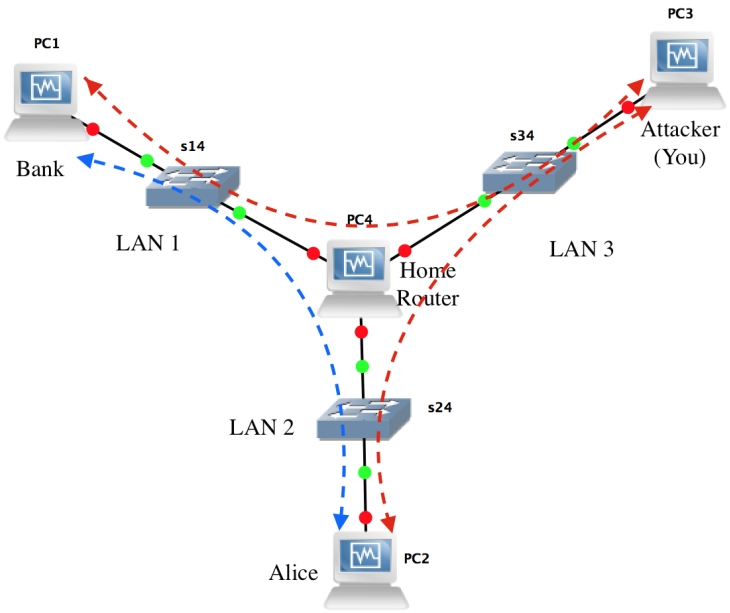

Figure 3: Basic configuration with modified labels

4.1 SETTING UP THE CONFIGURATION

In order for our attack to work properly, we will disable *Reverse path filtering* functionality on Alice's home router. Reverse path filtering (RPF) is a security mechanism where whenever the machine (PC4 in our case) receives a packet, it will first check whether the source IP address of the received packet is reachable through the interface it came in. If it is reachable it will accept the packet; if it is not it will drop it. For more information regarding RPF, here is a link explained by Sarath Pillai (`http://www.slashroot.in/linux-kernel-rpfilter-settings-reverse-path-filtering`):

To disable RPF for IPv4 in all Alice's home router's interfaces, run the script `mitmConfig.sh` script in the folder `/media/lca2/shared/Lab1/scripts` on PC4 terminal.

Finally, we will need to enable IPv4 forwarding on the malicious server (PC3). Do this with the command learnt from previous sections.

4.2 IP TABLES

The Linux kernel contains a packet filter framework called `netfilter` which enables a Linux machine to masquerade source or destination IP addresses. The command used to do this is called `iptables -t nat`, and it manages the table that contains rules regarding address masquerading. This table has two important types of rules:

(i) `PREROUTING`: responsible for packets that just arrived at the network interface, implying rules *before* any routing decision has been made.
(ii) `POSTROUTING`: responsible for packets with a recipient *outside* the linux machine, implying rules before the packet leaves through the network interface (after routing rules).

In this section of the lab, we will learn how to use both rules in order to carry out our MITM attack.

Start a wireshark capture of interface `PC3-eth0` in PC3. Do a ping from Alice's PC (PC2) to the malicious server (PC3). Execute the following command on Alice's home router (PC4):

```
# iptables -t nat -A POSTROUTING -o PC4-eth2 -j MASQUERADE
```

In this command,

- `iptables -t nat` is the command that modifies the masquerade table of the `netfilter`.
- `-A POSTROUTING` says to append a rule to the `POSTROUTING` rules (`-A` stands for `Append`).
- `-o PC4-eth2` states that this rule is valid for packets that leave on interface `PC4-eth2` (`-o` stands for `output`).
- `-j MASQUERADE` states the action that should take place is to "masquerade", i.e., replace source ip address.

Q14/ Answer **Question 1 in Lab 1 - Part 4** on Moodle

Solution. *Do a ping again from Alice to the malicious server and check in wireshark the differences between packets.*
What source and destination IP addresses do you observe for the ping response before executing the iptables command:
Source IP: `10.10.30.3`*, Destination IP:* `10.10.20.2`
after the iptables command :
Source IP: `10.10.30.3`*, Destination IP:* `10.10.30.4`

If you want to remove any configuration from the `iptables` or if you want to flush the mapping or "masquerading" table type both of the following commands:

```
# iptables -F
# iptables -t nat -F
```

Verify that iptables is disabled by doing another ping from Alice to the malicious server.

Execute the following command on Alice's home router (PC4):

```
# iptables -t nat -A PREROUTING -d 10.10.10.1 -j DNAT --to-destination
10.10.30.3
```

Q15/ Answer **Question 2 in Lab 1 - Part 4** on Moodle

Solution. *Now ping from Alice to the bank(PC1) and check in wireshark the source and destination IP addresses of the packets leaving Alice's computer(PC2). Source IP:* `10.10.20.2`*, Destination IP:* `10.10.10.1`
-j MASQUERADE *is used when we want to modify the source IP address if the masquerading IP address belongs to the host machine, i.e. it's* `eth0` *IP address.*
-j DNAT *is used to masquerade the destination IP address.*

Flush again the iptables with the commands provided in this section.

4.3 Configuring the MITM attack

Let's hack Alice's home router. First let's send the traffic targeted to the bank to go to the malicious server. Since Alice may have other internet activity with high bandwidth consumption (e.g. online gaming), and since Alice may have classmates working with her, we want to select only the traffic we are interested in (*ICMP* packets), and coming only from Alice's PC IP address (`10.10.20.2`). Keep doing the wireshark capture on the malicious server.

Q16/ Answer **Questions 3 to 7 in Lab 1 - Part 4** on Moodle

Solution. • *Question 3 in Lab 1 - Part 4*
Write down the command required to this end. If you need help, check the iptables link for masquerade: `https://www.netfilter.org/documentation/HOWTO/NAT-HOWTO-6.html`

```
# iptables -t nat -A PREROUTING -p icmp -s 10.10.20.2 -d 10.10.10.1 -j
DNAT --to-destination 10.10.30.3
```

Check with wireshark that you have successfully redirected the Bank's traffic from Alice's home router to your private (and malicious) server. Now we need to configure such server (PC3) in order to receive the traffic from Alice's home router, masquerade it, and send it to the bank. To this end, we need to replace the destination IP address of the packet with the IP address of the bank server `10.10.10.1`*, and modify the source IP address with your server's IP address* `10.10.30.3`*.*

Start a new wireshark capture on malicious server's `PC3-eth0` *interface and on Bank's* `PC1-eth0` *interface*

• *Question 4 in Lab 1 - Part 4*
Complete the iptables commands that need to be added to the malicious server in order to complete this task. (Make sure that IP forwarding on PC3 is enabled.)
Since we need to change the destination IP address, we need to do a PREROUTING *rule with the* -j DNAT*; and for the source IP address changing we need to do a* POSTROUTING *rule with the* -j MASQUERADE

```
# iptables -t nat -A PREROUTING -p icmp -s 10.10.20.2 -d 10.10.30.3 -j
DNAT --to-destination 10.10.10.1
```

```
# iptables -t nat -A POSTROUTING -o PC3-eth0 -s 10.10.20.2 -d
10.10.10.1 -j MASQUERADE
```

On the malicious server (PC3) you should see packets from `10.10.20.2` to `10.10.30.3` (and viceversa), and from `10.10.30.3` to `10.10.10.1` (and viceversa).

Additionally, on Bank's server (PC1) you should see packets coming from `10.10.30.3`. Since this is something that the bank could detect as malicious (e.g. it has an access-control list allowing connections only from its customers' IP addresses), propose the iptables command that you need to configure on Alice's home router (PC4), which is required to masquerade the malicious server with Alice's IP address `10.10.20.2`.

• *Question 5 in Lab 1 - Part 4*
Complete the iptable command that you need to configure on Alice's home router (PC4), which is required to masquerade the malicious server with Alice's IP address 10.10.20.2:
We need to use the SNAT *option as opposed to* MASQUERADE *because the source IP address we want to hide does not correspond to PC4's `eth0` IP address but to a different IP address.*

```
# iptables -t nat -A POSTROUTING -o PC4-eth0 -s 10.10.30.3 -d
10.10.10.1 -j SNAT --to-source 10.10.20.2
```

That's it, you have successfully mounted a man-in-the-middle attack. However, as you can see, in the current attack, the malicious server bounces the traffic back to the bank and the host PC3 does not receive it.

• *Question 6 in Lab 1 - Part 4*
Now, let us assume that there is a program at PC3 that modifies the traffic before bouncing it back to the bank. What should you change in the current configuration on PC3 iptables in order to redirect the traffic to the program that runs on PC3 host, more precisly what command should you add or remove ?
In order to allow the traffic to reach the program on PC3, we need to remove the rule

```
# iptables -t nat -D PREROUTING -p icmp -s 10.10.20.2 -d 10.10.30.3 -j
DNAT --to-destination 10.10.10.1
```

Thus, the traffic will reach the program on PC3, and then the program modifies the traffic and sends it to the bank.

• *Question 7 in Lab 1 - Part 4*
In our lab environment, is there any way (other than checking Alice's empty bank account) that Alice could notice that she is under attack?
Yes, if we compare pings before and after mounting the attack, we will see a difference in the RTT and the TTL values. When the attack is mounted, the traffic follows the path PC2→PC4→PC3→ PC4→PC1→PC4→PC3→PC4→PC2 thus delay is added to the RTT every time the packets gets in and out each device. Moreover since PC3 is also acting as a router (remember we enabled IPv4 forwarding), then the TTL is decremented by two to account for the two times the packet traverse PC3: one when it originally comes from PC2 and is routed to PC1, and the second when it comes back from PC1 and is routed

back to PC2.
In our lab, since we know the topology and we have very few hops in the network we can see a substantial difference in RTT and TTL with and without the attack. In a real environment the traffic between two points may be routed differently at different hours in one day, depending on the ISP's load balancing policies; consequently, it may traverse different intermediate routers thus having distinct RTT and TTL. Therefore, we cannot tell if we are under a MITM attack by just looking at RTT and TTL.
In reality, we can never be sure that a MITM attack is not present. Encryption, authentication and certificates are used to provide confidentiality, integrity and authenticity in presence of MITM attacks. For example, if Alice signs and encrypts her messages to the bank using a signature and encryption mechanism with a secret key, the malicious server who does not know the secret key can see the messages but cannot decrypt nor forge them.

5 MININET: ANALYZING LATENCY VERSUS NUMBER OF NODES (BONUS)

The following is a bonus section, and is completely optional. It will give you a chance to apply some of what you learned in this lab about Mininet, and it will allow you to gain a better understanding of how Mininet works, in addition to gaining more experience using it, which will be helpful for future labs.

We will be testing how the performance of Mininet scales with the number of nodes. Specifically, as the number of hosts, routers, and switches increases, how is the latency of the communication between these nodes affected?

In order to answer this question, you will need to create several scenarios, each with a Mininet topology in which the number of nodes, and the number of links between them, vary. Of course, this depends on your VM setup, and how much resources (memory, CPU) you assigned to the VM.

One example scenario is a star topology: a router in the center with N interfaces, each interface connected to a switch that is connected to a host. In such a scenario, each host is in a different LAN, and each pair of hosts is separated by the router.

We ask you to analyze two scenarios: star topology, as previously described and a binary tree topology, in which all leaf nodes are hosts and the rest of the nodes are routers.

Q17/ Answer **Lab 1 - Part 5** on Moodle

Solution. *To evaluate both topologies with different number of host we use the* `pingall` *command*

In a star topology, increasing the number of nodes doesn't affect the latency if only two nodes are communicating between them. Thus, the latency is only impacted by the amount of nodes that generate traffic, as adding new nodes does not change the average distance between nodes.
When all nodes communicate, we observe network congestion and thus higher latency as all the traffic is handled by a single router.

In a binary tree topology, increasing the number of nodes affects the average latency, independently of the number of nodes communicating.

Therefore, the average latency change is mainly due to the network shape change when adding new nodes which increases the average end-to-end latency. On the other hand, adding nodes will also add multiple routers to the topology. This will result in better network load balancing.

Suppose you have a tree topology with depth d, half of the nodes, on one side of the tree, will communicate

*with the other half of the nodes having a latency of 2d * single link latency.*

Grading The bonus part will be graded separately from the lab. This grade will be part of your research exercise grade.