

Name 1:

Name 2:

COM-407: TCP/IP NETWORKING

LAB EXERCISES (TP) 1

INTRODUCTION TO MININET

With Solutions

September 25, 2019

Deadline: October 9, 2019 at 23.55 PM

Abstract

In this TP you deploy a virtual machine that is pre-installed with Mininet, a virtualized environment used to emulate hosts and switches that will be used throughout this course. 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 OVERVIEW

Most of the labs of this course will run in a virtualized environment that can be installed directly on your own computer. With the virtualized environment, you will be able to operate a network with several hosts, routers, and other communication equipment, all in your own machine. This considerably simplifies the operation of the labs and may prove useful outside this course whenever you have to test a communication system.

The virtualized environment is an emulated¹ environment, i.e. the virtual hosts and routers run the same code as real physical hosts and routers; only their hardware is replaced by the virtualized environment.

The virtualized environment is mainly composed of the following items:

- A virtualization software, like **VirtualBox**, provides hardware emulation to support virtual routers and standard PCs. Version 6.0.10 was tested for compatibility with the lab, but versions 6.0.x should all work similarly.
- Next, we need to install the virtual machine in the virtual box. The virtual disk for the virtual machine is available on moodle.
- The virtual machine already has Mininet installed on it, **Mininet** provides emulation of networks and cables; it will be used to create network topologies.

¹In contrast, a *simulated* environment such as ns3 replaces hosts and routers with simplified code.

2 VIRTUALBOX: INSTALLATION GUIDE

Download VirtualBox² and install it on your computer.

Download the compressed virtual HD image `TCP-IP-2019.vdi.zip` (the link is on Moodle).

IMPORTANT FOR WINDOWS USERS Please use Winrar to uncompress the image, as other uncompressing tools will corrupt it. Also, keep the compressed version stored locally to avoid re-downloading the file in case of issues later on.

NOTE The uncompressed image takes roughly 6GB (and may grow if you want to install additional packages). Thus, you will need about 7GB of free space on your hard drive for the TCP-IP labs.

2.1 YOUR FIRST VIRTUAL PC

We will first now create a virtual machine.

1. Launch the VirtualBox software.
2. Create a new virtual machine, set the name to `MininetVM`, the type to `Linux`, choose version `Other 64bit` (and press next).
3. Assign 1GB of RAM (press next).
4. Choose “use an existing virtual hard disk file” and pick the uncompressed file `TCP-IP-2019.vdi`.

After the machine is created, we will add two features to it. First, we will create a shared folder between the guest machine and the host machine. With this folder you can easily backup and transfer configuration scripts, snapshots and any file you might find useful during your labs. Second, we will enable the “copy to clipboard” functionality between your host machine and the guest machine, and between guest machines as well. Note that you must do these changes when the VM is not launched.

1. **For the shared folder:** In virtualbox go to the machine settings: (`Devices`→ `Shared Folders...`→ `Add`). Point the “Folder Path” to a folder of your choice on your host machine, and in the “Name” field write `shared`. Check the box `Auto-mount`. Point the “Mount point” to `/media/lca2/shared`. Check the box `Make permanent`.
2. **For the clipboard copy:** Go to the machine settings in `General`→ `Advanced` and set the shared clipboard to “Bidirectional”.

Lastly, we will attach a NAT to the VM for it to be able to access the internet via the host connection. This should be done by default, but in case it is not, follow the steps below.

1. Go to machine settings (`Network`→ `Adapter 1`).
2. Enable the adapter by checking the box “Enable Network Adapter”.
3. Next, under the field “Attached to”, choose “NAT”.

Congratulations, you have just created a virtual machine!

²You can also use a different virtualization software but we will provide support and instructions only for VirtualBox.

2.2 RUNNING THE VIRTUAL MACHINE

In VirtualBox select `MininetVM` and run it.

Login using the username is `lca2` and password `lca2`. (Password for azerty keyboards on the login page: `lca2`).

The virtual machine should already be connected to the Internet via the host's connection. Open the Firefox web browser and go to a webpage to check that you are indeed connected.

In case you have issues launching your virtual machine, you may need to activate hardware virtualization. To do this you need to access the BIOS menu of your computer. You might need to power off your computer after enabling this option (not just a simple reboot!).

If you have any issues at this point (no Internet connection, cannot launch VM, cannot copy paste, cannot access shared folder) that you cannot resolve yourself, contact a TA for help.

2.3 LINUX CRASH COURSE (OPTIONAL)

This section is meant to provide a brief introduction to Linux commands and best practices. Feel free to skip this section and go directly to Section 2.4, if you are familiar with Linux. Note, however, that these are the basic commands that you will need to be familiar with for all the remaining labs.

2.3.1 LINUX COMMANDS

The Linux distribution of the virtual machine comes with a friendly graphical interface. For configuring network interfaces however, we will use the terminal. Here are a few things you need to know:

Linux is a multi-user system that uses the Extended file system (e.g., `ext2`, `ext3`, `ext4`) to store files. In extfs each file has a unique owner (a user), belongs to a group of users, and has a set of permissions which define access rights to the file (read, write, and/or execute) for the owner, the group, and everyone else.

Each normal user has a home directory, that is referred to by the symbol `~` (tilde). To change directories in the terminal use the command `cd` followed by the name of the directory. This can be a relative name, such as `Documents`, or an absolute name, such as `/etc/init.d` (i.e., beginning with the `/`, which is the root of the filesystem). It can also be the home directory (i.e., `cd ~`). To move up in the tree, i.e., out of a directory, use `cd ..` (two dots). The current directory is always represented by a single dot, so `cd .` does nothing. To display the current directory use `pwd` (print working directory).

Try it yourself: open a terminal³. Use `pwd` to show the current directory. Then `cd` to `~/Tutorial/`. Use the Tab key for auto-complete.

In the terminal, you can list the files in a directory by using the command `ls`. You can add switches to a command. For example, if you want to see detailed attributes of all the files in a directory (including the permissions), you can use the `-l` switch:

```
$ ls -l
```

You should see something like this:

³Tip: The shortcut to open a terminal is `Ctrl+Alt+T`.

```
lca2@lca2:~/Tutorial$ ls -l
total 4
-rw-r-r-- 1 lca2 lca2  0 Aug 18 12:14 emptyFile.txt
-rw-r-r-- 1 lca2 lca2 12 Aug 18 12:16 helloWorld.txt
```

You can output the contents of a file to the terminal by using commands such as `cat` or `less`:

```
$ cat helloWorld.txt
```

You can see above that the permissions of the `helloWorld.txt` file are `-rw-r--r--`, that it belongs to the user `lca2` and the group `lca2`, and that it is 12 bytes long. The permissions string is 10 characters long. The first character is either a dash `-` for regular files, or other letters for special files (a `d` for directories, etc.). The next three characters give the permissions for the owner of the file, in this case `rw-`. This means that the owner has the right to read the file (`r`), to write/modify the file (`w`), but not to execute the file. If the file was executable, in the third position there would be an `x`. The next three characters describe the permissions of the group, and the last three characters the permissions of all the other users in the system. In this case, the file is read-only for the group and for everyone else.

A file that the user `lca2` does not want anyone to see but herself would have permissions `-rw-----`, whereas a file with full rights for everybody would have permissions `-rwxrwxrwx`.

A file's permissions can be changed by using the command `chmod`. You need to specify whose access rights to the file you want to alter: of the user who owns it (`u`), of the group (`g`), or of others (`o`), whether you want to add (+), or remove (-) a right, and which right you mean (`r`, `w`, or `x`).

For example,

```
$ chmod o-r,g+w emptyFile.txt
```

removes the reading right for other users than the owner or the group and adds writing for the group. To change the ownership of the file, use `chown`.

When you issue a command in the terminal, you are in fact running a certain executable file. The command interpreter (or the shell) looks for these executable files in one of the several directories specified in the `PATH` environment variable. To list the contents of this variable, run

```
$ echo $PATH
```

The character `$` before `PATH` indicates that we want to display the contents of the variable `PATH`; without it, the command would simply display the string `PATH`. The directories are separated by semicolons, and they are searched in order. To see which executable you are running, use the command `which` followed by the name of the executable. For example, `which ls` displays `/bin/ls`, the location of the `ls` executable.

Note that the current directory (`.`) is not in the `PATH` for security reasons (a miscreant user might create an executable called `ls` in some directory, which in fact erases the given directory instead of listing it). Therefore, if you really want to execute a file in the current directory, you need to specify the path (the current directory), i.e., to type `./some_script` instead of simply typing `some_script` (the latter results in a "file not found" error).

Normal users cannot alter system configurations files (they do not have permission). For this reason it is safer to use a Linux machine as a normal user, and not as an administrator. This way, you cannot do too much harm.

There is a super-user (administrator) called `root` that has absolute rights (i.e., can do **anything**). In the terminal, the command prompt for a normal user ends with a dollar sign `$`, whereas for the `root` the prompt ends with a hash `#`.

IMPORTANT In these labs, whenever you see the hash `#` sign in front of a command that you are supposed to type, it means that you need `root` access.

There are users called “sudoers” that are allowed to run a single command as `root` (the user `lca2` in our virtual machine is such a user). This is achieved by typing `sudo` followed by the desired command. You will then be prompted for the password of the user.

If you want to run a terminal in `root` mode, type the command `sudo su`. You will then be prompted for the `root` password and you will switch to `root` mode. the password for root is `lca2`.

2.3.2 BEST PRACTICES

In these labs you will often type configuration commands in the terminal, usually one by one, to observe and understand their effects. However, after a reboot, the effects of these commands are usually lost, and you need to type them again, which is cumbersome.

We recommend the following practice:

Keep a text editor open in the virtual machine (for example “Leafpad”, located in Accessories, or `nano` in another terminal). Whenever you type a configuration command in the terminal, paste it in the editor. In Linux it suffices to select a text to copy it in the clipboard. For pasting use the middle mouse button. Otherwise use the standard “right-click” + Copy (but be warned that this might not work in all terminals). The shortcuts for copy and paste on the terminal are `Ctrl+Shift+C` and `Ctrl+Shift+V`, respectively.

Save the resulting file in your home directory (for example as `conf.sh`). When you reboot, you can run all the commands in the file as root by

```
# sh conf.sh
```

or as a regular user via `sudo` by

```
$ sudo sh conf.sh
```

2.3.3 KEYBOARD LAYOUT

The keyboard layout on your virtual machine is `us` by default. The command to change it to swiss one is

```
$ sudo setxkbmap ch
```

However, you will have to type this command every time when you reboot your system. If you want to make this command executed upon booting the system write the script (for example `keyboard.conf.sh`)

```
#!/bin/sh
sudo setxkbmap ch
```

You have to set it executable with

```
$ chmod +x keyboard_conf.sh
```

And place the script in `/etc/profile.d`

2.3.4 ADDITIONAL INFO

There are two main software packages that provide tools for configuring the network: the older, standard `net-tools` (provides `ifconfig`, `route`, `netstat`), and the newer and more powerful `iproute2` (provides `ip`, `ss`). Both are installed on the virtual machine, but we will focus primarily on the second set of tools (here is an angrily argumented viewpoint <http://inai.de/2008/02/19>).

2.4 TEST SOME COMMANDS

Start Wireshark on the virtual machine. To be able to capture packets using wireshark, you need to be a super user. Enter `sudo wireshark` in a new terminal/or a new tab (you will not be able to use this terminal/tab while wireshark is running) to start wireshark.

```
$ ping 10.0.100.200
```

Q1/ Describe and explain your observations in Wireshark

Solution. *There is no host with IP address 10.0.100.200 thus you should only see echo-request with no echo-reply*

To test that the shared folder works, copy three files to the shared folder of your **host** machine and rename them `file.1`, `file.2` and `.file.3` (with a dot at the beginning).

Q2/ On your virtual machine, open a terminal, go to the folder `/media/lca2/shared` and type the command `ls`. What do you observe?

Solution. *Files starting with a dot are not shown by `ls`, unless you append the option `-a` (show all).*

2.5 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 client and server functionality, 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.

2.5.1 TCP CLIENT AND SERVER

In order to launch iperf TCP server use:

```
$ iperf -s
```

In order to launch iperf TCP client use:

```
$ iperf -c iperf_server_ip_address
```

2.5.2 UDP CLIENT AND SERVER

In order to launch iperf UDP server use:

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

In order to launch iperf UDP client use:

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

There are also publicly available iperf servers, in this lab you will use `iperf.he.net`.

Q3/ Start Wireshark on your **host** and **guest** machines. Then connect a TCP iperf client on your guest machine to `iperf.he.net` (`iperf -c iperf.he.net`. **Note: If you cannot connect with this command, try to change the server port to listen on using `-p` option: `iperf -p port_number -c iperf.he.net` with `port_number` equals to 5201, 5202, etc.)** Compare the traffics captured on your host and guest machines. What are the source and destination IP addresses and port numbers of the packets coming from the server? Are there any differences in IP addresses and port numbers between two captures? If you see any differences, explain why.

Solution. *On host machine: the source IP address is 216.218.227.10 (IP address of iperf.he.net) and port number is 5001, the destination address is 192.168.0.14 (IP address of the host machine) and port number is 54057.*

On guest machine: the source IP address is 216.218.227.10 (IP address of iperf.he.net) and port number is 5001, the destination address is 10.0.2.15 (IP address of the guest machine) and port number is 45868. We observe that source IP addresses and port numbers are the same in both cases, whereas destination IP addresses and port numbers are different. This difference is caused by a NAT between the guest and the host network.

3 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. To learn more about the advantages and limitations of Mininet over other emulation platforms, take a look at [this](#) seminar from SIGCOMM 2014.

3.1 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 3.2. 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`.

Q4/ What are the different links in the network?

Solution. *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 with 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 -ls
```

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

Q5/ What command will you use to check the network interfaces of the host `h2`? How many interfaces does host `h2` have? What is/are its/their IP addresses?

Solution. *h2 ifconfig*
h2 has two interfaces h2-eth0 and lo.
Their IP addresses are 10.0.0.2 and 127.0.0.1, respectively.

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

```
mininet> h2 ping h1
```

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.

Q6/ What is the source IP address of the ICMP echo replies sent from h1 to h2?

Solution. *10.0.0.1*

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.

3.2 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 `scripts` folder, copy it to your shared folder on virtual machine. 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 later 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.

Q7/ List down the names of the interfaces that are available for capture.

Solution. `h2-eth0, lo`

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

```
mininet> exit
```

More information on the Python API can be found [here](#).

4 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.

4.1 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, you 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. 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.

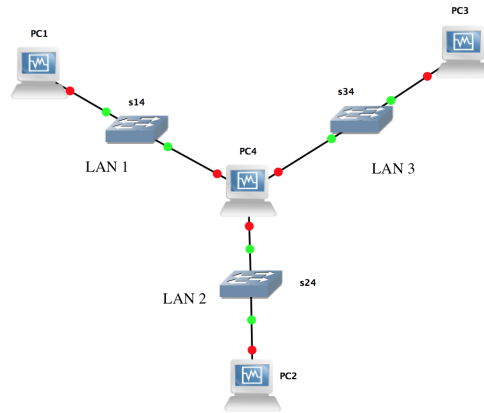


Figure 1: Basic configuration

4.1.1 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.

Q8/ How many IPv4 networks does PC4 belong to? Name the type of the network(s).

Solution. 3, Local area networks (LANs)

Q9/ Write down the IPv4 addresses you will use for the interfaces according to this addressing scheme.

Solution.

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

4.1.2 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.

Q10/ Write down the IPv6 addresses you will use for the interfaces according to this addressing scheme.

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
```

Q11/ Write down the MAC address of the interfaces.

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.

Q12/ Compute the IPv6 link-local addresses for the interfaces below.

Solution. *Answer may vary, but check for consistency with the previous question.*

PC3-eth0 – `fe80::0a00:27ff:fedb:9e20/64`
PC4-eth2 – `fe80::0a00:27ff:feff:5bed/64`

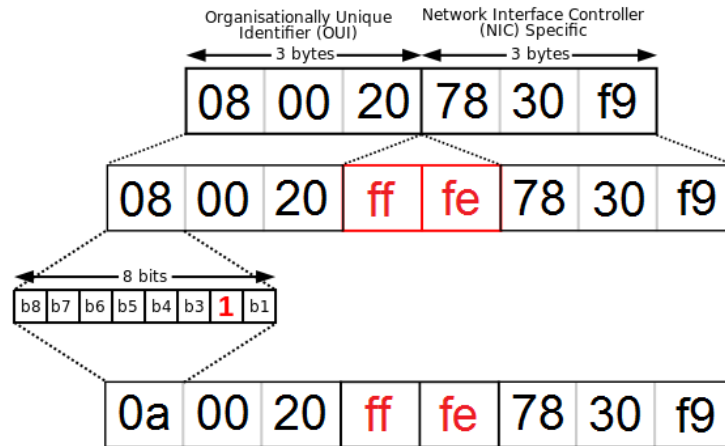


Figure 2: Formation of EUI-64 starting from MAC address of an interface.

4.2 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) – see Section 2.3.1.

To see the network interfaces of any PC, type:

```
$ ip link
```

Bring up the interfaces of PC4. Run in a 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 incoming 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 the PC1:

```
$ ping 10.10.10.4
```

Q13/ Explain what happens.

Solution. *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.*

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
```

Q14/ Describe the traffic you observe with Wireshark.

Solution. *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
```

Q15/ Describe and explain the differences (if any) between the output from the two ping commands.

Solution. *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*

4.3 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.

4.3.1 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 both for 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 PC1 type:

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

Q16/ Write down the routing table entries.

Solution. *ip route: 10.0.0.0/8 dev PC1-eth0 proto kernel scope link src 10.0.0.1
ip -6 route: fe80::/64 dev PC1-eth0 proto kernel metric 256*

To configure your network interfaces⁴ and assign an IPv4 and an IPv6 address to PC1-eth0 interface, open a root terminal (su) and type:

```
# ip addr add 10.10.10.1/24 dev PC1-eth0
# ip -6 addr add fd24:ec43:12ca:c001:10::1/80 dev PC1-eth0
```

Make sure to delete any 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 PC1-eth0
```

Q17/ What are the routing table entries after assigning the new ip addresses?

Solution. *ip route: 10.10.10.0/24 dev PC1-eth0 proto kernel scope link src 10.10.10.1
ip -6 route: fd24:ec43:12ca:c001:10::/80 dev PC1-eth0 proto kernel metric 256.*

Q18/ For the IPv4 addresses assigned to PC1-eth0, identify the host and subnetwork portions. Why is it important for the PC to know this information?

Solution. *With the netmask /24, the subnetwork is 10.10.10 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 PC1, run another ping to PC4's IPv4 interface:

```
$ ping 10.10.10.4
```

Q19/ What do you observe on Wireshark? Explain

Solution. *ARP packets are sent, but still no ICMP since PC4 is not configured.*

Configure the interfaces of PC4 and the one of PC2 and PC3 according to the addressing scheme.

To verify the configuration, type:

```
$ ip addr show
```

Try again pinging PC4 from PC1. It should work now. Do the same for PC2 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:

⁴Tip: 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.


```
$ ip neigh
```

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

4.4 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
```

Q20/ Does it work? Why?

Solution. *No, no route is available. There is no default gateway configured in PC1*

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.

Q21/ Which command(s) you need to fix the problem?. On which PC did you apply them?

Solution. *A route needs to be added on PC3, for example by configuring PC4 as a default gateway.*

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
```

Q22/ Perform the following pings simultaneously: PC2 from PC1, PC3 from PC2, and PC1 from PC3. Comment on the round-trip times that you observe.

Solution. *They are roughly the same in all cases.*

Q23/ Do an iperf test between PC1 and PC2 (PC1 run as server) using TCP and UDP. Report the performance (bandwidth values) you obtain with TCP and UDP. Do the port numbers for server and client are changing if you repeat experiment several times?

Solution. *The bandwidth with TCP is 18 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.*

5 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 4.4. 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).

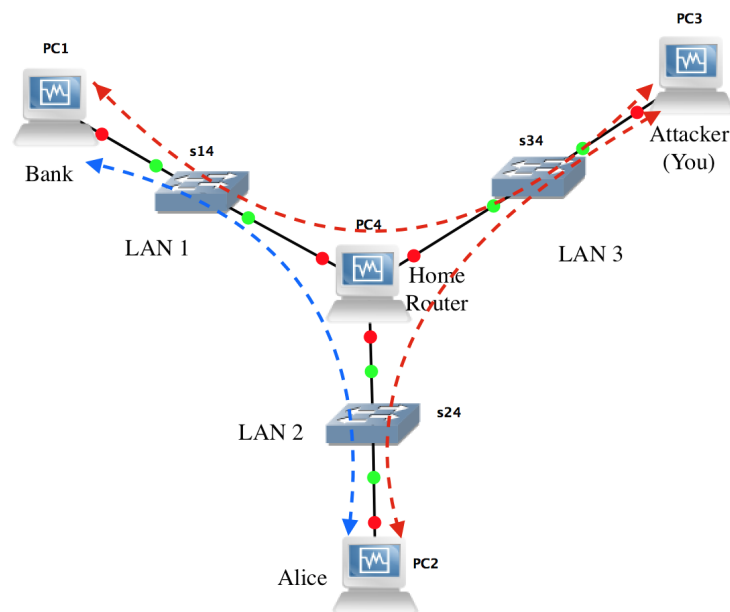


Figure 3: Basic configuration with modified labels

5.1 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.

5.2 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.

Q24/ Do a ping again from Alice to the malicious server and check in wireshark the differences between packets. Write-down any difference in source and destination IP addresses.

Solution. *Before the iptables the source IP address was 10.10.20.2; after iptables the new source IP address is 10.10.30.4. Destination IP address remained intact.*

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

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

Verify that `iptables` is disabled by doing another ping from Alice to 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
```

Q25/ Do one more ping from Alice to the bank and check in wireshark source and destination IP addresses. Based on your observations, what are the use cases for the “-j MASQUERADE” and “-j DNAT” configuration options?

Solution. -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.

5.3 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.

Q26/ Write down the command(s) required to this end. If you need help, check the iptables howto for masquerading (NAT):

(<http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO-6.html>)

Solution.

```
# 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

Q27/ Propose the iptables command(s) that needs to be added to the malicious server in order to complete this task. (Make sure that IP forwarding on PC3 is enabled.)

Solution. 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.

Q28/ Write down the command(s) here

Solution. *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 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.

Q29/ 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 current configuration on PC3 iptables in order to let the traffic come to the program that runs on PC3 host?

Solution. *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.

To conclude the lab just answer the final questions:

Q30/ Can this attack be mounted by just tampering the ip routing table of Alice's home router (PC4)?

Solution. *No. If we use IP routes (tampering the ip routing table), then traffic would be routed based on destination IP address only. In order to carry out the attack we need to modify the source IP address as well which cannot be done by manipulating the ip routing table.*

Q31/ 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?

Solution. *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.

6 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. How do communication latencies (ping or otherwise) change as the number of hosts N increases from 2 to 5 to 10 to 100?

We ask you to analyze up to two scenarios: either one scenario other than the star topology, or two scenarios including the star topology.

What to submit A PDF file containing a small description of each scenario you analyzed, along with the results (plots, tables) you obtained, and a snippet of the source code used to generate the topology. Put this PDF file along with the PDF file of this report in the same zip folder `Lab3_Name1_Name2`, and submit this folder.

Grading The bonus part will be graded separately from the lab, and you will receive a grade between 0-10. This grade will be part of your research exercise grade.