Name 1:
Name 2:

COMPUTER NETWORKING

LAB EXERCISES (TP) 1
INTRODUCTION TO GNS3 AND VIRTUALIZED ENVIRONMENT
With Solutions

September 25, 2015

Abstract

In this TP you will deploy on your own machine the virtual environment that will be used for most of the labs in this course, and you will perform basic network configuration tasks.

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 PCs, 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.

We will use the standard nomenclature: we will call the physical machine running the environment (i.e., your machine) *the host* and the virtual machines (i.e., the emulated Linux machines) *the guests*.

The virtualized environment is an emulated[1] environment, i.e. the virtual PCs and routers run the same code as real physical PCs and routers; only their hardware is replaced by the virtualized environment.

The virtualized environment is mainly composed of the following items:

- GNS3 provides emulation of networks and cables; it will be used to create and simulate network topologies. It provides a graphical user interface to design and configure virtual networks. (`http://www.gns3.net/`). Make sure to download version `1.3.10` since this version was tested for compatibility with the lab.
- A virtualization software, like VirtualBox, provides hardware emulation to support virtual routers and standard PCs. Make sure to download version `5.0.2` as this version was tested for compatibility with the lab.

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

- Last, we need to install real PC or router operating systems into the virtual boxes. In this lab we will use SliTaz (http://www.slitaz.org/en/), a lightweight Linux distribution. The virtual machine is available on moodle.

2 VIRTUALBOX: INSTALLATION GUIDE

Download VirtualBox[2] and install it on your computer. We strongly recommend you to use version `5.0.2` since it is the one used to test all TPs

Download the compressed virtual HD image `SlitazVM.vdi.gz` (the link is on Moodle).

NOTE Slitaz is a relatively small distribution in terms of required disk space. Still, the uncompressed image takes roughly 1.5GB (and may grow if you want to install additional packages). Since we will run several machines for the labs, you will clone it several times. Thus, you will need about 6.5GB of free space on your hard drive for the TCP-IP labs.

2.1 YOUR FIRST VIRTUAL PC

We will first create one virtual machine. We will then clone this machine.

1. Launch the VirtualBox software.
2. Create a new virtual machine, set the name to `PC1`, the type to Linux, choose version Other 32bit (and press next).
3. Assign 384Mo of RAM (press next).
4. Choose "use an existing virtual hard disk file" and pick the uncompressed file `SlitazVM.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.

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`[3]. Check the box `Auto-mount`.
2. **For the clipboard copy**: Go to the machine settings in `General`→ `Advanced` and set the shared clipboard to "Bidirectional".

Congratulations, you have just created a virtual machine! Before starting it, let us create three clones named PC2, PC3 and PC4. For each clone:

1. On the first VM Right-click→ Clone
2. Select reinit MAC address in all interfaces
3. Select complete (full) clone

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

[3] There is a link on the Desktop of the VM that doesn't work otherwise, so only change the name if you know what you are doing.

2.2 RUNNING THE VIRTUAL MACHINE

In VirtualBox select `PC1` and run it.

Login as `tux` with password `tux`.

`PC1` should already be connected to the Internet via the host's connection. Open the Midori browser and go to a webpage to check that you are indeed connected. If it is not the case, 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.

2.3.1 LINUX COMMANDS

The Slitaz distribution comes with a friendly graphical interface. For configurating 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/tcpip`, 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 terminal. Use `pwd` to show the current directory. Then `cd` to `~/Images/ASCII`. 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:

```
tux@slitaz:~/Images/ASCII$ ls -l
total 16
-rw-r--r--    1 tux      users          226 Aug 24 12:13 gnu-linux.txt
-rw-r--r--    1 tux      users          141 Aug 24 12:13 gnu.txt
```

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

```
$ cat gnu-linux.txt
```

You can see above that the permissions of the `gnu.txt` file are `-rw-r--r--`, that it belongs to the user `tux` and the group `users`, and that it is `141` 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 `tux` 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 tux.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 `$` 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 `tux` 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 `su`. You will then be prompted for the `root` password and you will switch to `root` mode.

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 Applications/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).

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 ADDITIONNAL 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

In PC1 start Wireshark and run

```
$ ping 10.0.100.200
```

Q1/ Describe and explain your observations in Whireshark

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 `~/Desktop/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).*

After this activity, complete the following tasks:

- **shut down** PC1.
- For each guest machine, go to Machine → Settings → Network. Make sure that for "Adapters" 1,2 and 3 you have the adapter enabled (checkbox) and in "attached to" the option *not attached* is chosen.

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 INTERFACE BETWEEN VIRTUAL MACHINES AND GNS3

Download and install GNS3 from the Internet. We strongly recommend to download version `1.3.10` since it is the one used to test all TPs. We will first configure GNS3 to interface with the virtual PCs you just created.

1. Open GNS3, start a new project.
2. Go to the menu `edit`, then `preferences`. In the left side click on `VirtualBox` and then `VirtualBox VMs`.
3. For each PC:
   - Click on the `new` button
   - Choose one of the virtual machines `PCx` and click on `finish`
   - Select the VM you just created and click on `edit`
   - In the `Network` tab change the number of adapters to `3`.
4. At the end click `Apply` and then `OK`

3.2 WIRING AND ADDRESSING SCHEME

In GNS3, create a configuration with four virtual PCs. Connect PC1 and PC2, and PC3 to PC4 like in Figure 1.

Press the start button in GNS3. This will start the four virtual machines.

3.2.1 RESETTING THE LABEL OF INTERFACES (`ETH0`, `ETH1`, `ETH2`)

All TPs assume that the labeling of the interfaces of your virtual PCs is `eth0`, `eth1` and `eth2`. If this is not the case, then you need to do a correction. To verify this condition we can use the `ip link` command to check the label of the ethernet interfaces:

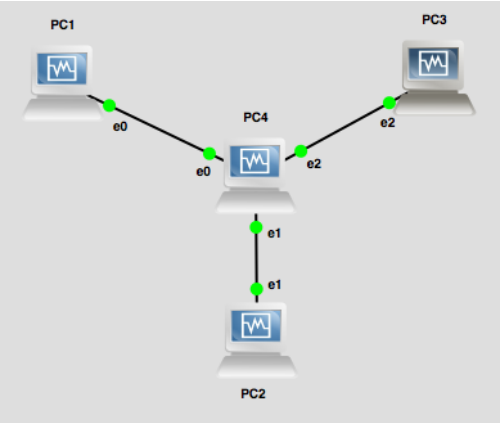

Figure 1: Basic configuration

```
# ip link
3: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
mode DEFAULT qlen 1000
link/ether 08:00:27:64:94:24 brd ff:ff:ff:ff:ff:ff
4: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
link/ether 08:00:27:e7:d6:16 brd ff:ff:ff:ff:ff:ff
5: eth4: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
link/ether 08:00:27:7b:1d:01 brd ff:ff:ff:ff:ff:ff
```

To set interface labeling to default (`eth0`, `eth1` and `eth2`), delete the following file on the affected PC, and then just reboot it.

```
# rm /etc/udev/rules.d/70-persistent-net.rules
# reboot
```

After coming back from the reboot, confirm with the same command (`ip link`) that ethernet-interface labeling is OK before proceeding any further.

3.2.2 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 `10.10.14.0/24`. The host identifier (the fourth byte) should be `1` to indicate the PC1 and and `4` for PC4.
- For the local network connecting PC4 to PC2 (LAN2), we will use the IPv4 network address prefix `10.10.24.0/24`. The host identifier (the fourth byte) should be `2` to indicate the PC2 and and `4` for PC4.
- For the local network connecting PC4 to PC3 (LAN3), we will use the IPv4 network address prefix `10.10.34.0/24`. The host identifier (the fourth byte) should be `3` to indicate the PC3 and and `4` for PC4.

Q3/ How many IPv4 networks does PC4 belong to? Name the type of the network(s).

Solution. *3 × LAN.*

Q4/ Write down the IPv4 addresses you will use for the interfaces according to this addressing scheme.

Solution.

PC1(eth0): `10.10.14.1`
PC2(eth1): `10.10.24.2`
PC3(eth2): `10.10.34.3`
PC4(eth0): `10.10.14.4`
PC4(eth1): `10.10.24.4`
PC4(eth2): `10.10.34.4`

3.2.3 IPv6 ADDRESSING

Several IPv6 addresses can be obtained by each interface. In the virtual GNS3 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:c001:14::/80` and addresses `fd24:ec43:12ca:c001:14::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:c001:24::/80` and the addresses `fd24:ec43:12ca:c001:24::X`, with `X` equal to `2` for PC2 and `4` for PC4.
- For LAN 3, we will use the IPv6 subnet `fd24:ec43:12ca:c001:34::/80` and the addresses `fd24:ec43:12ca:c001:34::X`, with `X` equal to `3` for PC3 and `4` for PC4.

Q5/ Write down the IPv6 addresses you will use for the interfaces according to this addressing scheme.

Solution.

PC1(eth0): `fd24:ec43:12ca:c001:14::1`
PC2(eth1): `fd24:ec43:12ca:c001:24::2`
PC3(eth2): `fd24:ec43:12ca:c001:34::3`
PC4(eth0): `fd24:ec43:12ca:c001:14::4`
PC4(eth1): `fd24:ec43:12ca:c001:24::4`
PC4(eth2): `fd24:ec43:12ca:c001:34::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 `eth1` interface, look at the `link/ether` field after issuing in a terminal the following command:

```
$ ip link show eth1
```

Q6/ Write down the MAC address of the interfaces.

Solution.

PC1(eth0): `08:00:27:32:76:ae`
PC2(eth1): `08:00:27:ba:76:45`
PC3(eth2): `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`

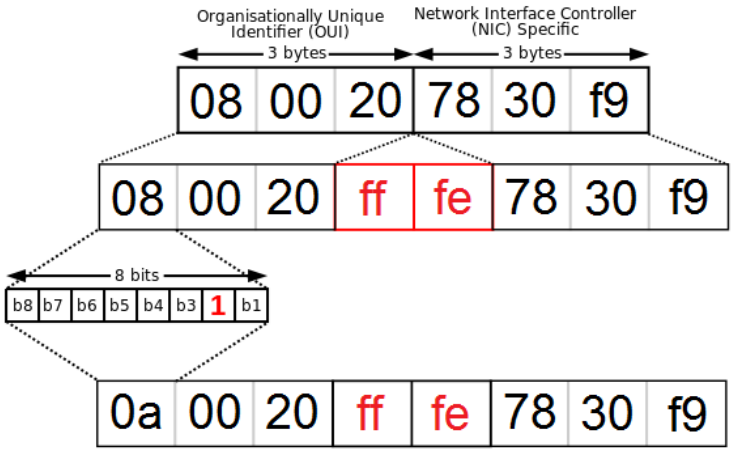

Figure 2: Formation of EUI-64 starting from MAC address of an interface.

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 setting 7th bit to 1 and inserting `ff:fe` in the middle. An illustration is given in Figure 2 adapted from the Wikipedia page dedicated to IPv6 addresses.

Q7/ Compute the IPv6 link-local addresses for the interfaces below.

Solution.

PC3 $fe80::0a:00:27:ff:fe:db:9e:20/80$
PC4(eth2) $fe80::0a:00:27:ff:fe:ff:5b:ed/80$

3.3 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 eth0 up
# ip link set eth1 up
# ip link set 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 `Whireshark` capture on all interfaces of PC4.

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 10.10.14.4
```

Q8/ Explain what happens.

Solution. *`ping` exits with the following error message: "connect: Network is unreachable". There are no routing rules for reaching the destinations and no packets are sent out on the link.*

Test IPv6 connectivity by running a *ping* to the link-local IPv6 address of PC4's `eth0` interface (the one that is connected to PC1) from a terminal on PC1.

```
$ ping <link_local_address_of_PC4_eth0>
```

Q9/ 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's `eth2` interface (the one that is in LAN 3) from a terminal on PC1.

```
$ ping <link_local_address_of_PC4_eth2>
```

Q10/ 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*

3.4 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.

3.4.1 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
```

Q11/ Write down and explain the routing table entries.

To configure your network interfaces[4] and assign an IPv4 and an IPv6 address to PC1's `eth0` interface, open a `root` terminal (`su`) and type:

```
# ip addr add 10.10.14.1/24 dev eth0
# ip -6 addr add fd24:ec43:12ca:c001:14::1/80 dev eth0
```

Q12/ Compare the routing table after assigning ip addresses with the one you got from last question. What are the new modifications used for?

Solution. *We can see new entries for the configured subnets. Notice there is no default gateway set, which means that we reach any host in this subnet directly (after an ARP/NS request).*

Q13/ For the IP addresses assigned to 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 `10.10.14` 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 10.10.14.4
```

Q14/ What do you observe? Has the analyzer captured anything?

Solution. *ARP packets are sent, but 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
```

[4] 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.

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:

```
$ arp -a
```

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

3.5 ROUTING PACKETS

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

For IPv4:

```
$ ping 10.10.24.4
$ ping 10.10.34.4
```

For IPv6:

```
$ ping fd24:ec43:12ca:c001:24::4
$ ping fd24:ec43:12ca:c001:34::4
```

Q15/ 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 10.10.14.4
# ip -6 route add default via fd24:ec43:12ca:c001:14::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 `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 10.10.34.3
```

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 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.

Q16/ 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
```

Q17/ Open a terminal on PC1, PC2 and PC3. Send pings from PC1 to PC2 and PC3; from PC2 to PC1 and PC3 and from PC3 to PC1 and PC2. Display the three virtual machine windows on screen and comment on the round-trip times that you observe.

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

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 3.5. 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 reallity 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).

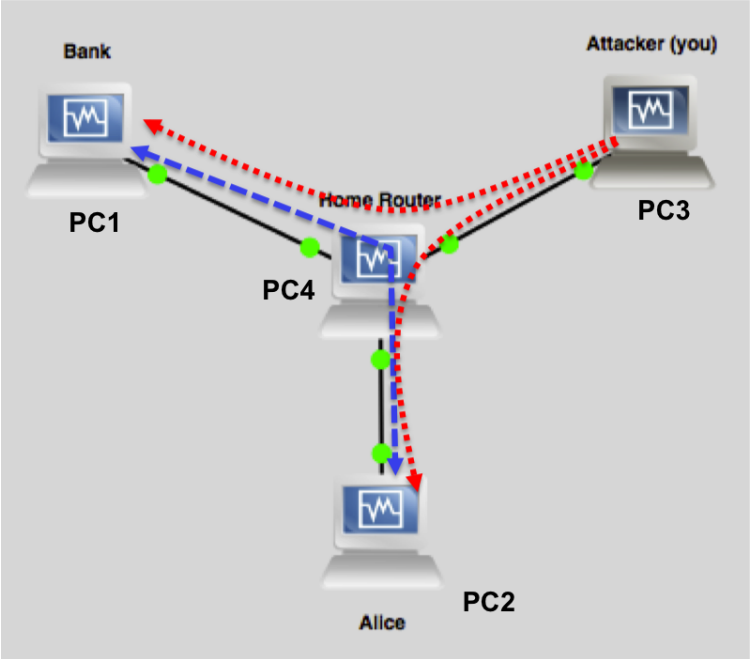

Figure 3: Basic configuration with modified labels

4.1 MINOR CONFIGURATION CHANGES REQUIRED

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) recieves 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 following commands as a superuser (mind the underscore!):

```
# echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
# echo 0 > /proc/sys/net/ipv4/conf/eth0/rp_filter
# echo 0 > /proc/sys/net/ipv4/conf/eth1/rp_filter
# echo 0 > /proc/sys/net/ipv4/conf/eth2/rp_filter
```

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 `eth2` 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 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 eth2` states that this rule is valid for packets that leave on interface `eth2` (`-o` stands for `output`).
- `-j MASQUERADE` states that the action that should take place is to "masquerade" or replace source ip address.

Q18/ 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 `10.10.24.2`; after iptables the new source IP address is `10.10.34.4`. 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 10.10.14.1 -j DNAT --to-destination
10.10.34.3
```

Q19/ 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.

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 (`10.10.24.2`). Keep the wireshark capture on the malicious server.

Q20/ 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 10.10.24.2 -d 10.10.14.1 -j
DNAT --to-destination 10.10.34.3
```

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 `10.10.14.1`, and modify the source IP address with your server's IP address `10.10.34.3`.

Start a new wireshark capture on malicious server's `eth2` interface and on Bank's `eth0` interface

Q21/ Propose the iptables command(s) that needs to be added to the malicious server in order to complete this task.

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 10.10.24.2 -d 10.10.34.3 -j
DNAT --to-destination 10.10.14.1
# iptables -t nat -A POSTROUTING -o eth2 -s 10.10.24.2 -d 10.10.14.1 -j
MASQUERADE
```

On the malicious server (PC3) you should see packets from `10.10.24.2` to `10.10.34.3` (and viceversa), and from `10.10.34.3` to `10.10.14.1` (and viceversa).

Additionally, on Bank's server (PC1) you should see packets coming from `10.10.34.3`. 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 `10.10.24.2`.

Q22/ 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 eth0 -s 10.10.34.3 -d 10.10.14.1 -j
SNAT --to 10.10.24.2
```

That's it, you have successfully mounted a man-in-the-middle attack. To conclude the lab just answer the final questions:

Q23/ 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.*

Q24/ 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 substancial 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.

```
% % %  BEFORE
tux@slitaz:~$ ping 10.10.14.1
PING 10.10.14.1 (10.10.14.1): 56 data bytes
```

```
64 bytes from 10.10.14.1: seq=0 ttl=63 time=0.589 ms
64 bytes from 10.10.14.1: seq=1 ttl=63 time=0.708 ms
64 bytes from 10.10.14.1: seq=2 ttl=63 time=0.692 ms
64 bytes from 10.10.14.1: seq=3 ttl=63 time=0.726 ms
64 bytes from 10.10.14.1: seq=4 ttl=63 time=0.801 ms
64 bytes from 10.10.14.1: seq=5 ttl=63 time=0.717 ms
64 bytes from 10.10.14.1: seq=6 ttl=63 time=0.764 ms
64 bytes from 10.10.14.1: seq=7 ttl=63 time=0.679 ms
64 bytes from 10.10.14.1: seq=8 ttl=63 time=0.800 ms
^C
--- 10.10.14.1 ping statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 0.589/0.719/0.801 ms
tux@slitaz:~$
tux@slitaz:~$
tux@slitaz:~$
% % %   AFTER
tux@slitaz:~$ ping 10.10.14.1
PING 10.10.14.1 (10.10.14.1): 56 data bytes
64 bytes from 10.10.14.1: seq=0 ttl=61 time=1.389 ms
64 bytes from 10.10.14.1: seq=1 ttl=61 time=1.324 ms
64 bytes from 10.10.14.1: seq=2 ttl=61 time=1.364 ms
64 bytes from 10.10.14.1: seq=3 ttl=61 time=1.359 ms
64 bytes from 10.10.14.1: seq=4 ttl=61 time=1.388 ms
64 bytes from 10.10.14.1: seq=5 ttl=61 time=1.249 ms
^C
--- 10.10.14.1 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 1.249/1.345/1.389 ms
```