COM-407: TCP/IP NETWORKING

LAB EXERCISES (TP) 2
L2 V.S. L3, NAT, PHYSICAL CONNECTION, AND TROUBLESHOOTING
With Solutions

October 20th, 2022
Deadline: November 2nd, 2022 at 23.55 PM

Abstract

In this lab you will work with the virtual environment introduced in Lab 1. First you will see the different behaviors of networking devices that work on layer 2 and layer 3; then you will configure your virtual network to be able to access the Internet; and finally you will connect one physical machine to another one and use its Internet connection.

1 PREPARING THE LAB

1.1 LAB REPORT

Answer questions on Moodle.

The deadline is Wednesday, November 2nd, 23:55

1.2 SET UP

Copy the **lab2 resources** folder from Moodle into the shared folder of your VM before starting the lab.

2 LAYER 2 VS. LAYER 3 NETWORKING

The aim of this section is to illustrate the difference between networking devices that work at layer 2 and layer 3.

2.1 USING A SWITCH AS A NETWORKING DEVICE

ANALYZE and ANSWER: For this part, answer the quiz **Lab2 - Part 1.1** on Moodle.

A switch is a MAC-layer device which expands a LAN by making forwarding decision based on destination

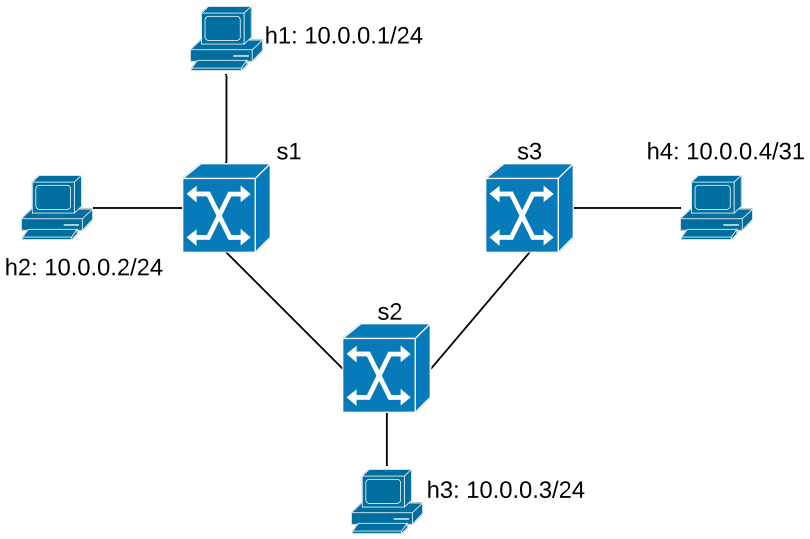

Figure 1: Loop-free network configuration with three switches

MAC-address. In this section you will learn how they work.

Open a terminal in your VM and run the script `topo1.py` as root (*password: lca2*), which should be located in the shared folder on the Desktop. If not, refer to Section 1.2.

```
# sudo python topo1.py
```

This will create the network described in Figure 1, and redirect you to the Mininet Command Line. Additionally, one terminal will appear for each of the four hosts. The four new terminals will be labeled (`h1, h2, h3, h4`) for convenience. `h1, h2` and `h3` should be configured with the `10.0.0.0/24` subnet with the fourth byte of their IP address being `1, 2` and `3`, respectively. Also, `h4` should have the IP address 10.0.0.4 with the subnet mask of `255.255.255.254`. Additionally, every host is automatically assigned an IPv6 address.

Warning! In the questions that follow, be careful when you copy-paste your answer from the mininet terminal to moodle. There might be some hidden characters and your answer will be marked wrong.

Q1/ Answer **Lab2 - Part 1.1** on Moodle.

Solution.

Part 1.1 Question 1:

There is only one LAN as there are only bridges and no router.

Part 1.1 Question 2:

They might be different for every machine, but all the hosts should have 64 bits prefix starting with `fe80`.
`h1-eth0`: $fe80::74d9:b8ff:fef8:73e0$
`h2-eth0`: $fe80::94d4:aeff:fe79:3320$
`h3-eth0`: $fe80::289b:33ff:fec6:4847$
`h4-eth0`: $fe80::0493:8bff:feab:01bb$.

These are link-local addresses, i.e. addresses that can be used for communicating in the same LAN only.

Now, let's test our configuration. Start Wireshark on all four hosts and capture on all the eth0 interfaces.

```
# wireshark &
```

Try to ping from each host to the others using its IPv6 address by executing the following command:

```
ping6 -I <interface name of host> <IPv6 address of destination>
```

Part 1.1 Question 3:

All hosts receive ping-reply. Since each host is using its link-local IPv6 address and all hosts are in the same LAN, it can send ping-request and receive ping-reply from other hosts.

Now, from terminal of `h1`, ping `h2` using its IPv4 address:

```
# ping <IPv4 address of h2>
```

Part 1.1 Question 4:

On `h1` and `h2` we are able to see ICMP echo-request, ICMP echo-reply, ARP requests and ARP replies (if any exists). In `h3` and `h4` we only observed ARP request packets (broadcast), if any exists.

Now, ping from `h1` to `h3` using IPv4.

Part 1.1 Question 5:

Traffic is the same, same ethernet header, same source/destination MAC-addresses. The Ethernet bridge does not affect any source/destination MAC-address, it is transparent to the MAC and IP layers.

Part 1.1 Question 6:

We don't see any packets. h4 will use its network mask on h1's IP address and check if they are in the same subnet. As they are not in the same subnet, h4 will attempt to contact its default gateway to send the packet, and if no default gateway is configured (which is this case), it will not send any packet at all.

Fix the configuration issue with host `h4`. You might have to flush the interface before :

 # ip addr flush dev h4-eth0

***Part 1.1 Question 7:**.*

`h4` subnet is not the same subnet as the other hosts are. You have to change the subnet mask in `h4`:

```
ip addr flush dev h4-eth0
ip addr add 10.0.0.4/24 dev h4-eth0
```

Part 1.1 Question 8:

IPV6 hosts usually self-allocate a link-local address automatically. This allows alls hosts that are in the same LAN to communicate using IPv6 without any configuration. In contrast, for the linux distribution we are using, no such thing happens with IPv4 (in contrast, link-local IPv4 addresses may be automatically allocated by Windows systems).

Exit Mininet and clean up the topology before going to next subsection:

```
mininet> exit
# mn -c
```

2.2 CONFIGURE A SWITCH TO HANDLE LOOPS

ANALYZE and ANSWER: For this part, answer the quiz **Lab2 - Part 1.2** on Moodle.

The goal of this subsection is to configure a LAN with loops. Similarly to the previous subsection, there are four hosts connected through three switches. The switches are forming a loop.

Run `topo2.py`. It creates the topology depicted in Figure 2.

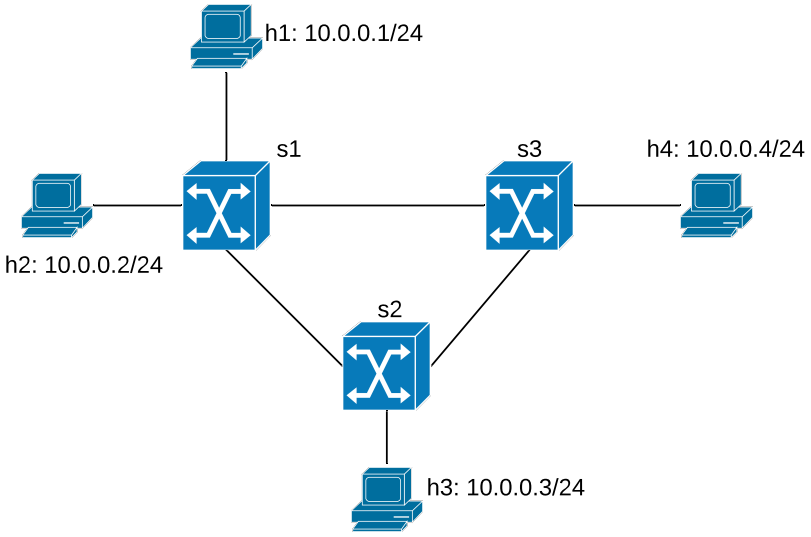

Figure 2: Network configuration with switches forming a loop

Now, perform a reachability test in Mininet using IPv4. A reachability test is a test to determine which hosts can 'reach' one another. This is performed by having each host ping all other hosts using its IPv4 address. A quick way to do this test in Mininet is by running the following command:

```
mininet> pingall
```

Q2/ Answer **Lab2 - Part 1.2** on Moodle.

Solution. ***Part 1.2 Question 1:***

The broadcasting of packets does not work due to loop in the network; this creates some inconsistency in the forwarding table but also create the possibility of an arp request to loop forever. For example, `h2` *wants to send a packet to* `h4` *for the first time. The forwarding table on* `s1` *does not have any information about this MAC address of* `h4` *in its forwarding table. It sends a broadcast packet to the LAN, searching for the owner of this MAC address (*`h4`*). The packet to* `h4` *traverses through* `s3` *and also through* `s2` *(and then* `s3`*).* `h4` *responds to both messages and send them back from their own path and in return the forwarding tables*

of the switches are updated. The switch `s1` receives both responses and has to update its forwarding table with two different value. This may cause inconsistency in the LAN.

Try to ping `h4` from host `h1` using their IPv6 address.

Part 1.2 Question 2:

The ping command does not work. Similarly to IPv4 addressing, using IPv6 addresses has the same issues in the network with loop: the broadcast issue and the inconsistency in the forwarding tables.

The standard solution to this problem is to enable the Spanning Tree Protocol (STP) at every switch.

Part 1.2 Question 3:

It breaks the loops in the LAN by forcing the active topology to be a tree; here it disables one of the interface.

The following command enables STP at the switch `s1`.

```
mininet > sh ovs-vsctl set bridge s1 stp-enable=true
```

Enable STP for all other switches in the network, wait for a minute and then perform a reachability test again and verify the connectivity of all hosts.

Let's check how STP affects the network of Figure 2. First, we open a terminal from Mininet:

```
mininet > xterm s1
```

Then, open a Wireshark from the terminal. You should be able to see all the interfaces for every switch in the network (not only switch `s1`). You can see the volume of traffic beside each interface.

To identify which interface corresponds to each link, you can get the status of links by typing:

```
# ip link show
```

on switch s1.

Execute the ping command for the following pairs of hosts.

- *From `h1` to `h3`*
- *From `h3` to `h4`*
- *From `h2` to `h4`*

Part 1.2 Question 4:

One link is disabled as a result of running STP. Here, the link between switches `s1` and `s2` is disabled (this might differ for each person).

Part 1.2 Question 5:

No. The shortest path between `h1` and `h3` is: $h1 \rightarrow s1 \rightarrow s2 \rightarrow h3$ (might be different for each person; but at least one of the pairs should not follow the shortest path). However, the link between the switches `s1` and `s2` is disabled due to STP and `h1` to `h3` does not follow the shortest path.

Shut down one of the active links between two switches but NOT the link s1-s3, namely `si` and `sj`, using the following command Mininet:

```
mininet > link <si> <sj> down
```

Now take a break and come back in `5` *minutes.*

Use the Ping command to check the connectivity of the hosts.

Part 1.2 Question 6:

The link that was disabled by STP is now enabled. It takes some time for STP to detect a broken link; as soon as discovery of no connectivity in the network, STP updates the forwarding tables of the switches again. Therefore 2 links are enabled and 1 link is disabled. The new path from h1 to h4 goes through h1-s1-s3-h4.

Part 1.2 Question 7:

- *From* `h1` *to* `h3` *:* $h1 \rightarrow s1 \rightarrow s2 \rightarrow h3$
- *From* `h3` *to* `h4` *:* $h3 \rightarrow s2 \rightarrow s3 \rightarrow h4$
- *From* `h2` *to* `h4` *:* $h2 \rightarrow s1 \rightarrow s2 \rightarrow s3 \rightarrow h4$

Exit Mininet and clean up the topology before going to next subsection:

```
mininet> exit
# mn -c
```

2.3 USING A ROUTER AS A NETWORKING DEVICE

ANALYZE and ANSWER: For this part, answer the quiz **Lab2 - Part 1.3** on Moodle.

We have already configured a router in Lab 1, but we did not address how it worked. In this section we learn

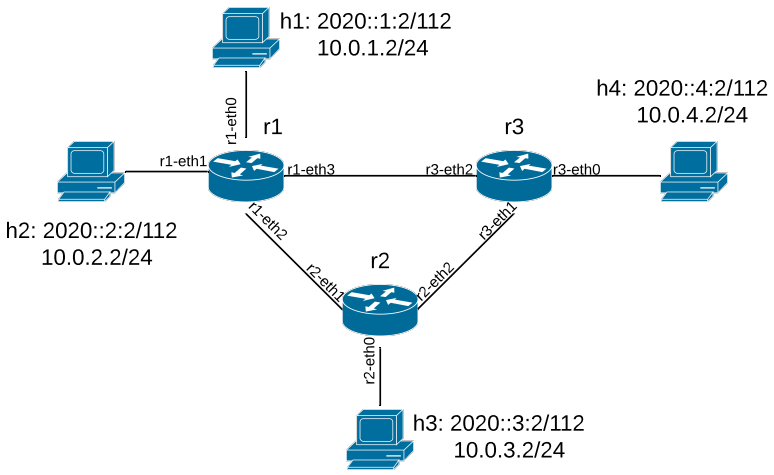

Figure 3: Network configuration with routers

about the process of routing a packet. To do so, run the script `topo3.py`. It creates the network topology with four hosts and three routers as shown in Figure 3.

Q3/ Answer **Lab2 - Part 1.3** on Moodle.

Solution.

Part 1.3 Question 1:

None of the hosts can ping each other.

We will now attempt to fix the problem. First, open the `topo3.py` *script and inspect it.*

Part 1.3 Question 2:

r1-eth0 → IPv4: `10.0.1.1`*; IPv6:* `2020::1:1`
r1-eth1 → IPv4: `10.0.2.1`*; IPv6:* `2020::2:1`
r1-eth2 → IPv4: `10.0.5.1`*; IPv6:* `2020::5:1`
r1-eth3 → IPv4: `10.0.7.1`*; IPv6:* `2020::7:1`

Part 1.3 Question 3:

We noticed that ip forwarding is disabled at `r1`*; the default gateway of* `h2` *is wrong: it is set to* `10.5.0.100` *for IPv4 and* `2020::10:2` *for IPv6, but these IP addresses do not exist in our network; and routing rules are not set in* `r2`*.*

Solve the issue at `r1`*.*

Part 1.3 Question 4:

We enable on `r1` *IPv4 forwarding:*

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

and IPv6 forwarding:

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

Try again to ping hosts from each other with IPv4 and IPv6 addresses.

Part 1.3 Question 5:

Only `h1` *can ping* `h4` *and vice versa.*

Now solve the issue concerning `h2`*.*

Part 1.3 Question 6:

For IPv4:

```
ip route add default via 10.0.2.1
```

For IPv6:

```
ip -6 route add default via 2020::2:1
```

Part 1.3 Question 7:

Hosts cannot ping `h3` *and vice versa.*

You can check the routing table on router `r2` *using the following command for IPv4 and IPv6, respectively:*

```
ip route show
ip -6 route show
```

Part 1.3 Question 8:

`r2` cannot reach any network it is not connected to, because the routing table is not set properly.

Part 1.3 Question 9:

A possible configuration is the following:

```
ip route add 10.0.1.0/24 via 10.0.5.1
ip route add 10.0.2.0/24 via 10.0.5.1
ip route add 10.0.4.0/24 via 10.0.6.2
ip route add 10.0.7.0/24 via 10.0.6.2
```

Note that we could configure different routes as well.

Part 1.3 Question 10:

The routing table for IPv6 addresses is not set properly on `r2`.

```
ip -6 route add 2020::1:0/112 via 2020::5:1
ip -6 route add 2020::2:0/112 via 2020::5:1
ip -6 route add 2020::4:0/112 via 2020::6:2
ip -6 route add 2020::7:0/112 via 2020::6:2
```

Ping again each host from another one using both IPv4 and IPv6 addresses, and confirm that your fix solves the problem.

Based on your observations, conclude this section by comparing switches and routers in a network.

Part 1.3 Question 11:

In a network with switches only, all hosts receive link-local IPv6 addresses automatically therefore can communicate with each other without configuration. With IPv4, all hosts should be configured with the same IPv4 subnet mask and prefix (except if IPv4 auto-configuration is happening).

In contrast, when using routers, each of its interface is connected to a separate IPv4/v6 network, i.e., subnet prefixes are different, and hosts can take whatever IPv4 and IPv6 addresses within the valid range. In this case, the hosts still receive link-local IPv6 addresses, but they cannot use them to communicate with another host in another subnet.

Part 1.3 Question 12:

The switches set the forwarding tables by learning. In a network without loops, no configuration is needed to connect the hosts. If the network has loops of switches, then by enabling STP in all of switches the forwarding tables are created . However, in a network of routers, the routing tables must be set manually (this can be done automatic by running protocols like OSPF (you will do it in the next labs).

Part 1.3 Question 13:

When there is a loop in the network of switches, one (or several) link is disabled to break the loop. The side-effects of this action is increasing the load on the other active links in the network and paths are not the shortest. However, in the network of routers, all links are used and the traffic is balanced. Moreover, all paths can be the shortest, e.g., if you run OSPF.

Now, exit Mininet and clean up the topology before going to next section:

```
mininet> exit
# mn -c
```

3 Connecting Virtual Environment to the Real World Using Network Address Translation (NAT)

ANALYZE and ANSWER: For this part, answer the quiz **Lab2 - Part 2** on Moodle.

Please read carefully! In Lab0, we asked you to connect the VM to the host machine using a "Network Adapter" of type "**NAT Network**". This was done to enable IPV6 connectivity inside the VM. However, we have noticed that this part of the lab does not work correctly with this type of adapter. For this part, please make sure you configure a "Network Adapter" of type "**NAT**". Please refer back to Lab0 to check how to change the adapter. In case IPV6 connectivity is needed in a future lab, we will remind you to change the network adapter again.

In this section we will use what we learned from Lab1 about manipulating the `iptables` filter. The purpose of the section is to connect an isolated virtual network that we have deployed so far, to the Internet.

Look at the Figure 4. The NAT in the box "Physical Machine" is the one created by VirtualBox. It connects the network interface of "LCA2 VM" to the physical interface of your laptop (**Note that in the network setting of the VM, there should be one Network Adapter which is set to "NAT"**).

As soon as you turn on the VM, remove the IP configuration of the interface connected to the NAT, as it is going to be used by Mininet. Get the list of interfaces in the VM and use the following command to flush the interface of the VM connected to NAT:

```
# sudo ip addr flush dev <interface name of VM connected to NAT>
```

Remember that the root password is `lca2`. Run the script `topo4.py`. This creates the network described in the box "Network in Mininet" shown in Figure 4. In this network, `h1` and `h2` are hosts, `r1` is also a host but configured to act as a perimeter router where we will have our connection to the real world. The goal of the switch `s3` is connecting `r1-eth1` to the network interface of the LCA2 VM. However, we know that LCA2 VM interface is used by the virtual machine itself. Therefore, we add a port to `s3` and connect it the network interface of LCA2 VM.

Q4/ Answer **Lab2 - Part 2** on Moodle

Solution.

3.1 Connecting the perimeter router to the Internet

The goal of this section is to connect `r1` to the Internet. To perform the bridging between physical and virtual network-adapters, execute the following command from Mininet terminal to connect the interface of VM to the switch `s3`.

```
mininet> sh ovs-vsctl add-port s3 <interface name of VM connected to
NAT>
```

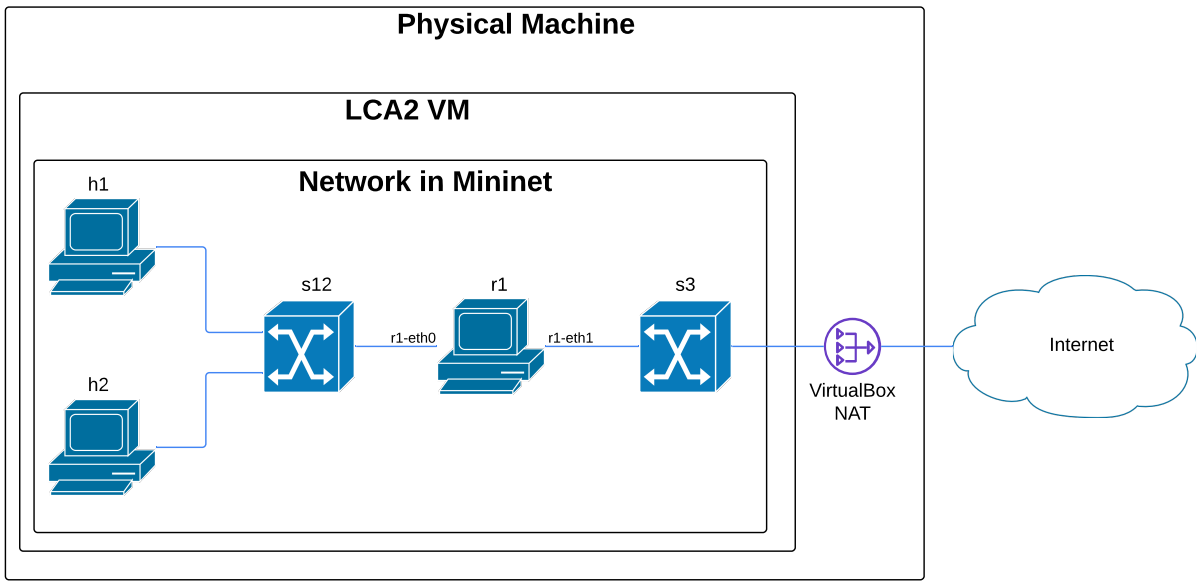

Figure 4: Network configuration with a connection to the real world

Replace `<interfacename>` *with the interface that accesses the Internet in your VM.*

The next step is to assign an appropriate IP address to the `r1-eth1` *interface of* `r1`*. The address should be in the same subnet as* `<interface name of VM connected to NAT>` *because both are connected by a switch and are thus in the same subnet. In a physical network, address allocation can be done manually or using a DHCP server. The same holds in the VM, as VirtualBox also provides a DHCP server. We use DHCP in order to avoid conflict with IP addresses that the VM might have allocated to other interfaces in the same subnet as the interface of the VM connected to the VirtualBox NAT. You can ask the DHCP server of VirtualBox to provide a valid IP address by using the following command in the terminal of* `r1`*:*

```
# dhclient r1-eth1
```

This automatically sets a usable IPv4 address to the `eth1` *interface of* `r1`*, allowing it to access the internet through the bridge we just set up. Test the configuration by pinging* `8.8.8.8`*.*

Part 2 Question 1:

The IPv4 address is 10.0.2.16. The IPv4 address is a private one (the network behind the NAT of VirtualBox). The IPv4 address of the DHCP server of VirtualBox is 10.0.2.2. When running the `dhclient` *command, you can run Wireshark and see the destination address of the packet sent to DHCP server.*

3.2 Providing Internet access to Mininet hosts

The goal of this subsection is to provide Internet access to hosts `h1` *and* `h2`*, via* `r1`*.*

Part 2 Question 2:

We need to allocate addresses to `h1` *and* `h2`*. Whatever addresses we end up allocating, we need that the network prefix used by* `h1` *and* `h2` *is visible to the rest of the virtual network, we should make VirtualBox aware of it. This may be hard to do and we prefer to use a NAT inside* `r1`*.*

In order to give Internet access to `h1` *and* `h2` *we will configure* `r1` *as a NAT. Indeed, the situation is the*

same as if `r1` would be connected to an ADSL modem at home: `r1` receives a single IP address from its provider (here: VirtualBox) and we want to use it to connect more devices (here: `h1` and `h2`).

Part 2 Question 3:

```
# iptables -t nat -A POSTROUTING -o r1-eth1 -j MASQUERADE
```

Test from `h1` and `h2` that you have Internet connectivity by pinging `8.8.8.8`. Next, let's explore in detail the result of our configuration.

Do `traceroute` to `8.8.8.8` from `h2` and then from `r1`, while capturing `eth0` and `eth1` traffic on `r1` using Wireshark. Explore the difference in the traffic on both cases.

Part 2 Question 4:

The source IP address is modified according to the NAT rule, replacing the IP address of `h2` with the IP address configured on `eth1` in `r1`.

Part 2 Question 5:

Source and destination ports are different. Source and destination IP addresses are the same.

Part 2 Question 6:

A symmetric NAT could use the following fields to identify the local IP: source IP address and port (of the server) and destination port (of the local machine)

Do `ping` to Google from `h1` and `r1`, while capturing the traffic on `r1` (both on `eth0` and `eth1`) using Wireshark. Explore the difference in the traffic in both cases.

Part 2 Question 7:

The ICMP query ID is different.

Part 2 Question 8:

ICMP query replies are forwarded back to `h1` based on the QueryID taken from the ICMP packet header. This is handled according to ***RFC 5508***

RESEARCH EXERCISES (OPTIONAL)

4 POINT-TO-POINT WIRED CONNECTION OF TWO PHYSICAL MACHINES

ANALYZE and ANSWER: For this part, answer the quiz **Lab2 - Bonus - Part 1** on Moodle.

In this section, you will connect two physical machines via an Ethernet cable. The goal of this section is to give you a feel about the communication between physical machines.

To accomplish this section, you are required to have access over two physical machines, e.g. your laptop and your friend's, and an Ethernet cable. If you need an Ethernet cable or a USB-to-Ethernet adapter, you can borrow one INF015.

Figure 5: Point-to-point wired connection of two physical machines

Q5/ Answer **Lab2 - Bonus - Part 1** on Moodle.

Solution.

4.1 SETTING UP THE CONNECTION

The goal of this subsection is to make a point-to-point connection between two physical machines, namely `M1` *and* `M2`*, via cable. To avoid any complication in the process, please turn off the Wi-Fi connection of the machines (or set it to flight mode). Now, physically connect the two machines by plugging in one port of an Ethernet cable to* `M1` *and the other port to* `M2`*.*

Get the list of interfaces by running `ifconfig` *in MacOS and Linux, and* `ipconfig` *in Windows. Teredo is the interface for tunneling IPv6 in IPv4 as mentioned in the "Network Layer" lecture.*

To find out their IP addresses, use the commands seen in the previous labs (e.g. with Linux and MacOS, `ip addr show`*; with Windows, use* `ipconfig`*).*

Bonus Part 1 Question 1:

The IPv6 addresses are link-local addresses (auto-configured). For IPV4, it depends on the system. Some systems auto-configure link-local addresses `169.254.x.x`*, some others configure private addresses, yet some others don't configure any address.*

If you are using Windows, turn off the Windows Firewall now, as it may block traffic between interfaces, for security reasons.

Bonus Part 1 Question 2:

Yes. They are able to ping each other using IPv6 addresses. Since two computers are in the same LAN and link-local IPv6 addresses are autoconfigured, they can ping each other.

Bonus Part 1 Question 3:

It depends on the system. If link-local IPv4 addresses were auto- configured, this should work. Otherwise, if private addresses are configured they should not be able to receive ping-reply because they are not in the same subnet.

We now set up a private IPv4 network between `M1` and `M2`, using private but routable addresses.

Bonus Part 1 Question 4:

The link local addresses are not routable, so we cannot use them if they were auto-allocated. We create a network with the IPv4 range, e.g., `10.2.2.0/24`.

at `M1`:

In Linux:

```
ip addr flush dev <M1 interface>
ip addr add 10.2.2.2/24 dev <M1 interface>
```

In Mac, use System Preference/Network to set up IP address manually.

In Windows, you need to go to Settings and change the IPv4 properties (address and network mask) of the corresponding interfaces.

at `M2`:

In Linux:

```
ip addr flush dev <M2 interface>
ip addr add 10.2.2.3/24 dev <M2 interface>
```

In MacOS, use System Preference/Network to set up IP address manually. A limited version of `ip` command is available for MacOS by installing the `iproute2mac` package:

With Windows, you need to go to Settings and change the IPv4 properties (address and network mask) of the corresponding interfaces.

Verify the connection of the machines by executing the ping command again with IPv4 on `M1` and `M2`.

4.2 MEASURING THE BANDWIDTH OF THE COMMUNICATION LINK

So far, you have built a point-2-point IPv4/v6 network between two physical machines via an Ethernet cable. The goal of this subsection is to find out the practical bandwidth of the Ethernet cable.

In Lab1, you worked with `iperf` and how to measure the physical bandwidth of a communication link. In this subsection, you want to measure the bandwidth of the Ethernet cable connecting `M1` and `M2`.

Run `iperf` as server in `M2`.

Bonus Part 1 Question 5:

```
iperf -s
```

The IP address is the one used by the interface of `M2` and the port is written in the terminal when the server runs.

Now run the `iperf` client in `M1`.

Bonus Part 1 Question 6:

```
iperf -c <IP of the server> -p <port number of the server>
```

Bonus Part 1 Question 7:

It should be higher than 80*Mbps (for* 100*Mbps link) or* 950*Mbps (for* 1*Gbps link).*

4.3 SHARING INTERNET ACCESS

ANALYZE and ANSWER: For this part, answer either the quiz **Lab2 - Bonus - Part 2 for Linux** or **Lab2 - Bonus - Part 2 for Windows/Mac**, depending on your operating system, on Moodle.

The goal of this subsection is to allow `M1` to access the Internet via `M2`. This is similar to "tethering", when you share a mobile phone's internet access with other devices that do not have Internet access.

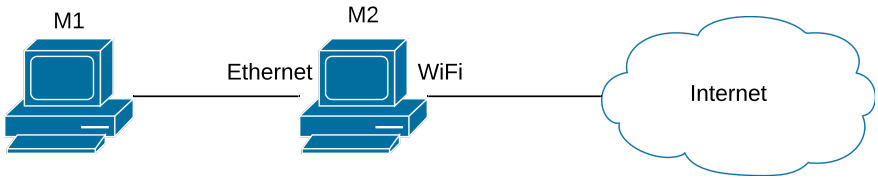

Figure 6: Sharing Internet with a friend!

Assume the configuration is as in Figure 6. We want to connect `M1` via `M2` to the Internet. We could setup `M2` as a bridge, a router, or a NAT.

The process depends on the OS that M2 runs. Please complete the quiz on moodle that corresponds to your case (Linux or Windows/Mac).

Q6/ Answer **Lab2 - Bonus - Part 2 for Linux** or **Lab2 - Bonus - Part 2 for Windows/Mac**, depending on your operating system, on Moodle.

Solution.

Bonus Part 2 Question 1:

- *Bridge: this requires the IP address of M1 to be a public address and be in the same subnet as the WiFi IP address of M2. This requires that the internet service provider gives us several IP addresses, not just one. This may be sometimes possible with IPv6 but usually not with IPv4.*
- *Router: this requires that we setup a subnet prefix for the Ethernet between M1 and M2 and that this prefix be a set of public addresses in the same subnet as the WiFi IP address of M2. Again, this may be sometimes possible with IPv6 but usually not with IPv4.*
- *NAT: this does not place any constraint on the addresses allocated to the Ethernet segment and requires only one public address for M2. This works with IPv4 and IPv6.*

Turn on the Wi-Fi interface of the physical machine `M2` and check its connectivity by pinging `google.com`. Note that `M1` still does not have Internet access.

4.3.1 Setting Up M2 as a NAT

If `M2` is running Linux, then it can be setup as a NAT. If this is your case, continue with this section; otherwise jump to Section 4.3.2.

Bonus Part 2 for Linux Question 2:

- *Install the iptables package if it is not present; then do*

```
sudo echo 1 > /proc/sys/net/ipv4/ip_forward
sudo iptables -t nat -A POSTROUTING -o <VM Wi-Fi interface> -j
MASQUERADE
```

Now you may go directly to Section 4.3.3.

4.3.2 Setting Up the LCA2 VM in M2 as a NAT

If `M2` cannot be natively configured as a NAT, we can use a VM inside `M2` and configure it as a NAT, since we know how to do this. This involves two steps:

1. *Connecting the VM to the Ethernet port*
2. *Setting up the VM as a NAT*

Connecting the VM to the Ethernet port *As usual, the VM is connected to the Internet (i.e to the WiFi interface) via a NAT. We could do the same to connect the VM to the Ethernet adapter, but since we have full control of all IP addresses allocated in this Ethernet LAN, we can do a simpler solution, i.e., use a bridge.*

Set two network adapters (a NAT and a Bridge) in host `M2`. To do so, you have to open "Settings" of the VM, then select "Network", go to tab "Adapter 1" and set it to "NAT" and tab "Adapter 2" and set it to

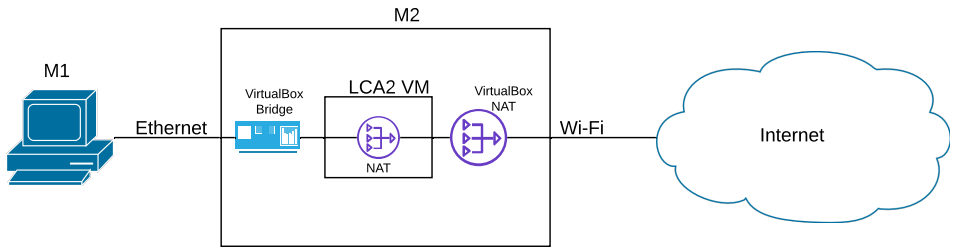

Figure 7: Sharing Internet with NAT in the VM

"Bridged adapter"; the name should be the Ethernet interface of your physical machine as it is used to connect `M2` *to* `M1`.

Now power on the LCA2 vitual machine. Open a terminal in the VM and ping `epfl.ch`.

Bonus Part 2 for Windows/Mac Question 2:

Open Wireshark. The interface that has traffic on is connected to NAT.

Use the following command on the VM in `M2` *separately to find out their IP addresses:*

```
ip addr show
```

It shows four adapters: LoopBack (LO), teredo, one connected to NAT (the one that already has and IPv4 address), and the other is connected to Bridge.

We need to check the connectivity of the VM in `M2` *to Internet via NAT (we already had it in Section 3) and set up its connection to* `M1` *via bridge.*

The first step to achieve this goal is to assign a suitable IPv4 address to the bridge interface of the VM.

Bonus Part 2 for Windows/Mac Question 3:

The IP address should be in the same subnet as `M1` *as they are in the same LAN.*

Bonus Part 2 for Windows/Mac Question 4:

```
sudo ip addr flush dev <bridge interface in VM of M2>
sudo ip addr add 10.2.2.4/24 dev <bridge interface in VM of M2>
```

Now, ping `M1` *from the VM and verify the connectivity of VM and* `M1` *through the bridge.*

SETTING UP LCA2-VM AS A NAT *So far, we have connected one interface of the VM in* `M2` *to the Internet via the VirtualBox NAT and WiFi connection, and the other interface of VM to* `M1` *via the VirtualBox bridge and Ethernet cable. The final step is to set up the LCA2-VM to work as a NAT.*

Bonus Part 2 for Windows/Mac Question 5:

First, enable IP forwarding for IPv4:

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

If permission is denied, try the following:

```
sudo sysctl net.ipv4.ip_forward=1
```

Then:

```
sudo iptables -t nat -A POSTROUTING -o <VM Wi-Fi interface> -j
MASQUERADE
```

4.3.3 FINAL STEPS!

Bonus Part 2 Question 6 for Windows/Mac or Question 3 for linux:

You need to configure a default gateway at M1 with the IP address of the bridge in the VM of M2. For example if M1 runs linux, you will type:

```
ip route add default via <IP address the bridge in the VM of M2>
```

Verify the Internet connection of `M1` by pinging `8.8.8.8` as the Google DNS server.

Assume a router `r` is the next hop of `M2`. The host `M1` starts to ping `8.8.8.8`.

Bonus Part 2 Question 7 for Windows/Mac or Question 4 for linux:

It is the IP address of the Wi-Fi interface of `M2`. As on `M2`, NAT is enabled, all the incoming traffic from `M1` is passed though the NAT in `M2`. According NAT, all the machines behind it are non-visible from the Internet and are seen `M2` as the sender/receiver of traffic.

One application of such configuration is to share your own Internet access with other people who are not connected to the Internet. Suppose a friend of you visits Switzerland and would like to have Internet access; however, the cost of Roaming is too much for him/her. Therefore, you would like to do him/her a favor and share your own Internet with him/her. This can be done by the practical experience you have obtained in this section.

Disclaimer: You may think of sharing your EPFL Internet connection using your GASPAR credentials and give Internet access to your friend. We would like to warn you that this generous behavior is unfortunately forbidden.

Note: If you borrowed any Ethernet cable or USB-to-Ethernet adapter, please return them to INF015; as otherwise, your submission will not be graded.

Note: If you are using a Windows machine, turn on the Windows Firewall again.