Name 1:
Name 2:

Computer Networking

Lab Exercises (TP) 2
L1 v.s. L2 v.s. L3, NAT, Physical Connection, and Troubleshooting
With Solutions

October 12, 2018

Abstract

In this Lab you will work with the virtual environment introduced in Lab 1. First you will see the different behavior of networking devices that work on layer 1, 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 share its Internet connection.

1 Preparing the Lab

1.1 Lab report

Type your answers in this document. We recommend you use Adobe Reader XI to open this PDF. When you finish, save the report and upload it on Moodle. Don't forget to write your names on the first page of the report. **The deadline is Wednesday, October 24, 23:59:59**

1.2 Set up

In this Lab, you will work with the same virtual machine that you created in Lab 1. Copy the **lab2 resources** folder from Moodle into the shared folder of your VM before starting the lab.

1.3 Using scripts

As a general advice, use scripts to save your work for each section, especially Section 3. This is useful for 1) saving time and not repeating the same commands each time you restart Mininet, and 2) reviewing and debugging your work in case you run into issues.

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.

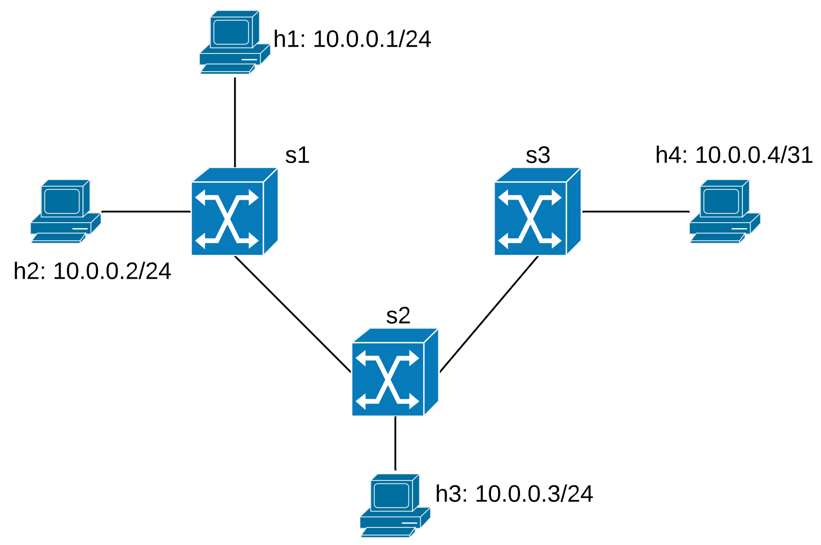

Figure 1: Loop-free network configuration with three switches

2.1 Using a switch as a networking device

A switch is a MAC-layer device which expands a LAN by making forwarding decision based on destination MAC-address. In this section you will learn how they work.

Open a terminal in your VM and run the script `topo1-line.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-line.py
```

This will create the network described in Figure 1, and redirect you to the mininet CLI. 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 located on 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 10.0.0.4/31. Additionally, every host is automatically assigned an IPv6.

Now, let's test our configuration. Start Wireshark on all four hosts. It will be hard to keep track of which Wireshark window corresponds to which host. One way to do so would be to start Wireshark on the hosts in order, i.e. `h1`, then `h2`, then `h3`, and finally on `h4`. This way the Wireshark windows will be in this same order in the taskbar. Start capturing on all the eth0 interfaces.

```
# wireshark &
```

From `h1`, ping `h2`.

```
mininet> h1 ping h2
```

Q1/ Describe the different types of packets observed on `h1, h2, h3` and `h4`.

Solution. *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.*

Q2/ Ping from `h4` to `h1` using IPv4. Observe the traffic captured and explain your findings.

Solution. *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.*

Q3/ What are the IPv6 addresses `h1` and `h4`?

Solution.
`h1`:
`h1-eth0`: $fe80::5caf:87ff:feb5:c04e$
`h4`:
`h4-eth0`: $fe80::c486:45ff:fef5:501c$

Try to ping from `h4` to `h1` using IPv6 using the following command:

```
ping6 -I <interface name of h4> <IPv6 address of h1>
```

Q4/ Is there any difference in the results of ping from `h4` to `h1` when using IPv6 and IPv4? Explain.

Solution. *Yes. `h4` is able to ping `h1` using IPv6 but not with IPv4. An IPv6 is automatically assigned to each host based on their MAC address. Since all the hosts are in the same LAN, `h4` can ping `h1` and vice versa. `h4` cannot ping `h1` with IPv4 as their do not have the same subnet.*

Q5/ Fix the configuration issue with host `h4`. What commands did you execute?

Solution. *`h4` 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
```

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

Q6/ Compare the packets sent by `h1` to the ones received by `h3`, specifically at source/destination MAC-addresses. Explain the similarities and differences, if any.

Solution. *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.*

2.2 Configure a switch to handle loops

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 `topo1-loop.py`. It creates the topology depicted in Figure 2.

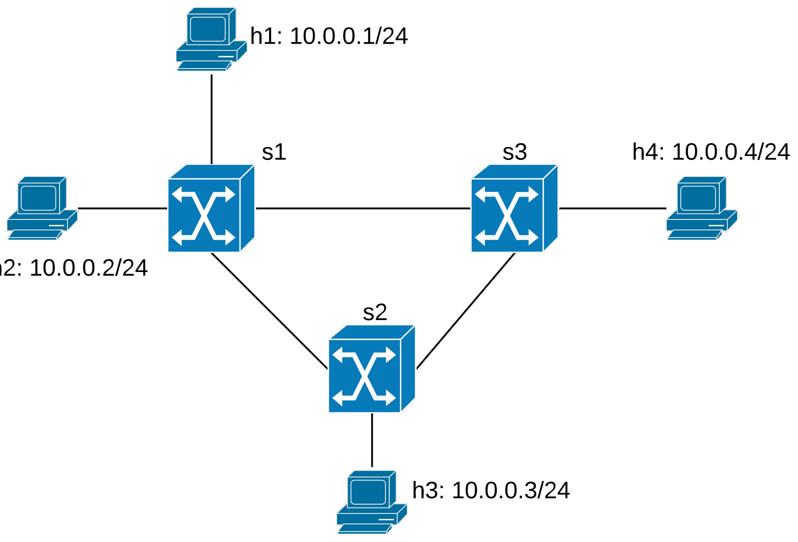

Figure 2: Network configuration with a router and a switch

Now, perform a reachability test in Mininet. 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. A quick way to do this test in Mininet is by running the following command:

```
mininet> pingall
```

Q7/ What is the percentage of dropped packets? Why does it happen?

Solution. *100%. Because the broadcasting of packets does not work as the network has a loop.*

Now ping `h4` from host `h1` using their IPv6.

Q8/ What is the percentage of dropped packets when using IPv6? Why does it happen?

Solution. *100%. Similarly to IPv4, because the broadcasting of packets does not work as the network has a loop. Using IPv6 does not help to avoid broadcast issue in networks with loops.*

Q9/ What is the solution to avoid the packet drops? What are the commands that should be executed?

Solution. *We have to avoid broadcast issue that is happening in networks with loops. The solution to do that is to enable Spanning Tree Protocol (STP) in all switches.*

In Mininet:

```
mininet > sh ovs-vsctl set bridge s1 stp-enable=true
mininet > sh ovs-vsctl set bridge s2 stp-enable=true
mininet > sh ovs-vsctl set bridge s3 stp-enable=true
```

Perform a reachability test again and check the connectivity of all hosts.

Now we want to observe the traffic in the switches using Wireshark. First, we open a terminal from Mininet:

```
mininet > xterm s1
```

Then, open Wireshark from the terminal. You should be able to see all the interfaces for every switch in the network (not only switch `s1`).

Execute the ping command for the following pairs of hosts.

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

Q10/ Write down the path of the packets for each pair of hosts.

Solution. *One link is disable as a result of running STP. Here, the link between switches `s1` and `s2` is disabled.*

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

Q11/ Are the hosts following the minimum hop count to send their packets to the destinations? Explain.

Solution. *No. The minimum hop count path between `h1` and `h3` is:* $h1 \rightarrow s1 \rightarrow s2 \rightarrow h3$*. However, the link between the switches `s1` and `s2` is disabled due to STP. The same phenomenon happens between `h3` and `h2`.*

2.3 Using a Router as a networking device

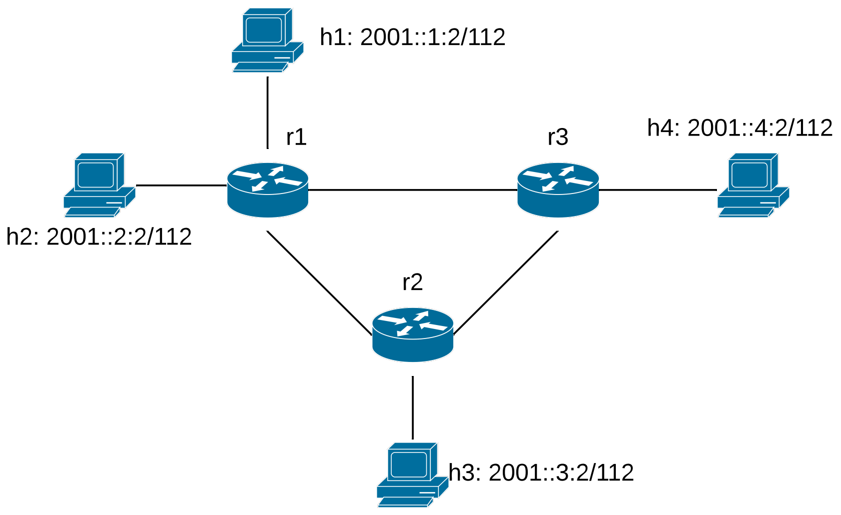

Figure 3: Network configuration with a router and a switch

We have already configured a router in Lab 1, but we did not address how it worked. In this section we learn about the process of routing a packet.

We'll begin by exiting Mininet, cleaning up the previous topology, and running `topo2.py`. The new topology consists of four hosts and three routers as shown in Figure 3.

```
mininet> exit
# mn -c
# python topo2.py
```

Perform a reachability test in Mininet using `pingall`.

Q12/ What is the percentage of dropped packets? What does this correspond to?

Solution. *33%, one third of the links are broken*

Q13/ Which hosts are unable to reach one another?

Solution. *h1 and h3; h2 and h3*

We will now attempt to fix the problem. First, open the `topo2.py` script and inspect it.

Q14/ What is the subnet mask used throughout the file?

Solution. *255.255.255.0*

Q15/ What are the interfaces and respective IP addresses of the router `r1`?

Solution. *r1-eth0: 10.0.0.100; r1-eth1: 10.0.1.100*

Q16/ Can you spot any misconfigurations in the file?

Solution. *We noticed that ip forwarding is disabled at `r1` and that the default gateway of `h2` is wrong: it is set to `10.0.0.101` but this IP address does not exist in our network.*

Solve the issue at `r1` and attempt the reachability test again.

Q17/ What is the command you used at `r1`?

Solution. *We enable IPv4 forwarding on `r1`*

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

Q18/ What is the percentage of dropped packets now in the reachability test? Which hosts are still unable to ping each other?

Solution. *16%; h2 and h3*

Now solve the issue concerning `h2`.

Q19/ What are the commands you used in `h2` to achieve this?

Solution.

```
ip route del default via 10.0.0.101
ip route add default via 10.0.0.100
```

Q20/ What are the results of the reachability test now?

Solution. *0% dropped packets. Our network is now properly configured.*

Now we learn about routing between two interfaces. Monitor the traffic in Wireshark of both interfaces of `r1`. From `h1`, ping `h3`.

Q21/ What changes are done to IP packets when they are routed between `r1`'s `r1-eth0` and `r1-eth1`?

Solution. *We see that the original source MAC-address of `h1` is replaced with `r1`'s `r1-eth0` MAC-address and destination MAC-address of `r1`'s `r1-eth1` is replaced with `h3`'s `eth0`. Also we see that TTL is decremented by 1. To get MAC-address info of a device, run the following command in its terminal:*

```
ifconfig -a
```

Q22/ What is the purpose of such changes?

Solution. *The reason is to adjust the IP packet to the new LAN where it is being routed to. When routing from `eth0` to `eth1` (or vice versa), `r1` removes the MAC header, next it reads the IP header, then it makes forwarding decisions based on the destination IP address, and finally it inserts a new MAC header which has the source/destination MAC-addresses compatible with the scope of the LAN between `r1` and `h3`. The reason for TTL is to avoid loops in the network.*

2.4 Routing with multiple hops

In this section we want to see what happens when we introduce a second intermediate routing device to the network. Let's start by setting the default gateway of `r1` to `h3`'s IP address, then on `h3` remove the default gateway and enable IPv4 forwarding.

Q23/ Type in the commands you need to do this.

Solution. *For `h3`*

```
# ip route del default via 10.0.1.100
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

For `r1`

```
#ip route add default via 10.0.1.3
```

Monitor `eth0` of `h3` and `r1`. Try pinging from `h1` to `h3` and `h2` to `h3`.

Q24/ Based on Wireshark captures, explain why it does not work.

Solution. *From `r1` we see ICMP echo-requests sent to `h3`. On `h3` we don't see any ICMP echo-reply. `h3` does not know how to reach `h1`'s IP address, which means that there is no route configured in `h3` to return ip packets to `h1` and `h2`. By default, if a network device does not know how to route a packet, it drops it.*

Q25/ Write down **one single command** that fixes the problem, namely pinging from `h1` and `h2` to `h3` while keeping internet access from `h3`. Specify the PC where you need to apply it.

Solution. *One* ***wrong*** *solution would be to return the default route on `h3`, and point to `r1`, but this would disable any future Internet access on `h3`, thus even though it would solve the problem for now, it will not work in the future section.*

A second approach is to add static routes in `h3` in order to reach `h1` and `h2`:

```
#ip route add 10.0.0.0/24 via 10.0.1.100
```

Ping again from `h1` and `h2` to `h3`, and confirm that your fix solves the problem before moving to the next section.

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

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

We will work in the network described in Figure 4. `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.

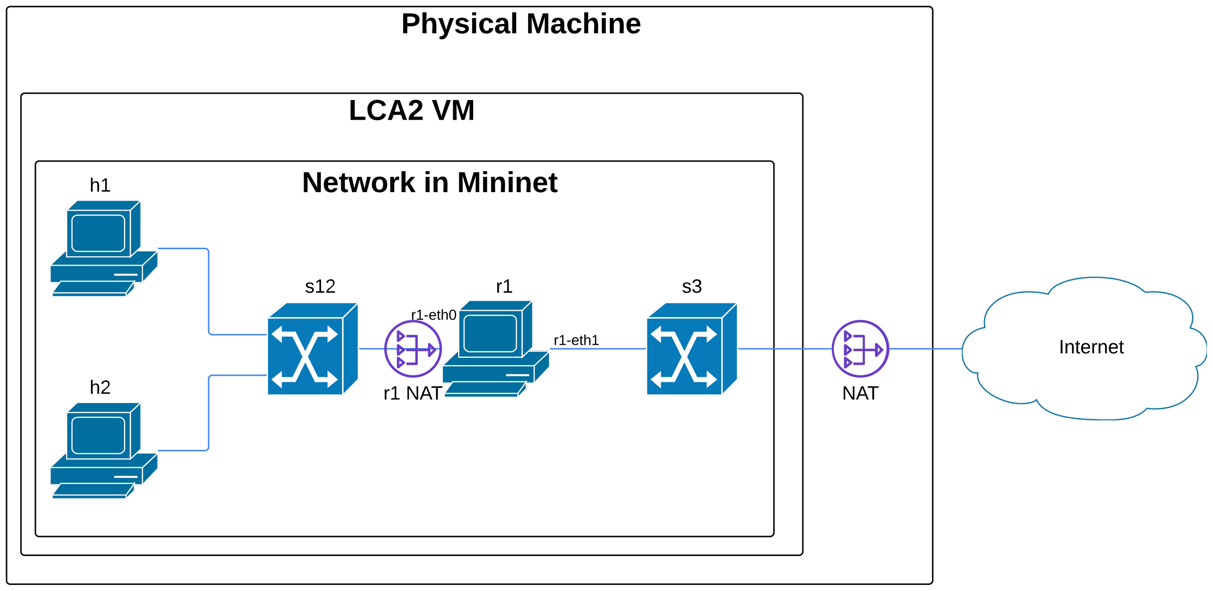

Figure 4: Network configuration with a connection to the real world

Q26/ We have one real connection (IP address) to the real world, but we have two clients (`h1` and `h2`) that require access to the internet. Which solution would you use to tackle this problem, and explain how would it solve the problem.

Solution. *We use network-address translations (NAT). NAT would create separate mappings by using separate TCP/UDP port of the real IP address of `r1`'s `r1-eth1` for each of `h1`'s and `h2`'s connections.*

There are two main steps to connect your virtual environment to the real Internet:

1. We require a real IP address on `r1-eth1` interface of `r1`.
2. We need to masquerade the traffic coming from `h1` and `h2`.

3.1 Borrowing an IPv4 Address

As we said before, we need an *existing* (real) IPv4 address that can be used to connect to the Internet. The purpose of this section is to obtain such valid IPv4 address. There are three steps for that:

1. Create a bridge between a real interface in your host machine, and `r1`'s `eth1`.
2. Finding a suitable IPv4 address.
3. Setting up the IPv4 address of `r1-eth1` on `r1`.

To perform the bridging between physical and virtual network-adapters, we first need to find out which interface our VM is using to access the internet.

Exit Mininet, clean up the topology, and run wireshark on the VM with root access. Start capturing on each interface and ping `www.google.com` and `www.epfl.ch` to find out if that interface is being used to access the internet.

```
mininet> exit
# mn -c
# sudo wireshark &
# ping www.google.com
# ping www.epfl.ch
```

Q27/ What is the IP address of the interface used to access the internet? Do you notice anything special about it? Take note of the IP addresses of Google and EPFL.

Solution. *Depends on the person. It is a private address given by the NAT of the VM, and not the public address given by the bridged adapter. The interface itself is important for the next few questions.*

www.google.com: 172.217.168.78 (it might be different)

www.epfl.ch: 128.178.222.108

Now that we know which interface is used by the VM, the next step is to implement a bridge between it and `r1-eth1`.

In Mininet, we can only implement a bridge with an OVS. To mitigate this, an OVS, i.e., `s3`, is added between `r1` and the internet cloud in Figure 4.

Run the script `topo2_internet.py` as root. This will create the same topology as before (in addition to all the extra configurations that you have performed, and addition of an extra OVS that we will use to perform the bridge).

From Mininet, execute the following command.

```
mininet> sh ovs-vsctl add-port s3 <interfacename>
```

Replace `<interfacename>` with the interface that accesses the Internet in your VM. Remember that the sudo password for the VMs is lca2. Now head over to the terminal of `r1` and run the following command.

```
# dhclient r1-eth1
```

This automatically sets a usable IP 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 Google or EPFL from `r1` using their IP addresses.

3.2 NAT Configuration

We worked with the command used to configure NAT in Lab1, `iptables -t nat`, which manages the table that contains rules regarding address translations. In this section, we will analyze how NAT works for different types of packets on router `r1`.

First let's see what happens when `h1`/`h2` access the Internet with their native IP address. Monitor with Wireshark the interface `eth1` of `r1`. From `h1`/`h2`, ping to Google with its IP address.

```
# ping -c 5 172.217.168.78
```

Q28/ Analyze the packets coming from `h1`/`h2` and explain why you are unable to reach Google.

Solution. *In interface `eth0` of `r1` we see packets with a private IP address in the source field. Most likely the first hop router in the ISP network will drop the packet as it is sourced by a private IP address.*
In the case you are doing the exercise from home and you have a private IP address in `eth0` of `r1`, it is likely that the private IP address you use for Mininet environment is already allocated by your ISP, therefore your ISP will route it according to its own rules which do not cover your virtual environment.

Q29/ Propose the `iptables -t nat` command you need to properly configure NAT in `r1`.

Solution.

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

Test from `h1` and `h2` and you have Internet connectivity by pinging Google and test that you have successfully configured your router to do NAT.

Next, let's explore how NAT works!!.

Do `traceroute` to Google 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.

Q30/ When doing `traceroute` from `h2`, what is the difference in the packets captured on `r1`'s `eth0` and `eth1`?

Solution. *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`.*

Q31/ Focus on the `traceroute` from `r1`. What is the difference in the packets as compared to `h2`?

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

Q32/ Which field in the UDP packet is used to identify the (local) source IP address of `h2` in order to properly forward incoming ICMP replies back to it?

Solution. *This is done via UDP port, and NAT device keeps a track on the private source IP/port and the correspondent public IP/port.*

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.

Q33/ What is the difference in the request ICMP packets captured between packets sent from `h1` and packets sent from `r1` when capturing on the exit interface of each?

Solution. *The ICMP query ID is different.*

Q34/ Conclude how the incoming ICMP replies are forwarded back to `h1` when doing `ping` from `h1`. In particular, which field in the request/reply ICMP packets was used to identify the (local) source IP address?

Solution. *ICMP query replies are forwarded back to `h1` based on the QueryID taken from the ICMP packet header. If we issue the `iptables -t nat -L -v -n` command, we will see this IP address to Query ID mapping. This is handled according to* ***RFC 5508***

4 Point-to-Point Wired Connection of two Physical Machines

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 real communication networks.

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 do not have enough machines or Ethernet cable, you may use the ones in Internet Engineering Workshop (IEW)/INF019.

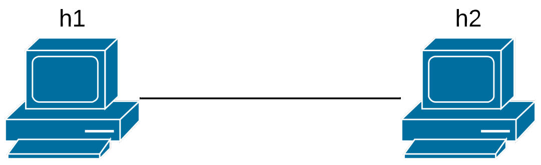

Figure 5: Point-to-point wired connection of two physical machines

4.1 Setting up the Connection

The goal of this subsection is to make a point-to-point connection between two physical machines, namely `h1` and `h2`, via cable. To avoid any complication in the process, please turn off the Wi-Fi connection of the machines (or set to flight mode). Now, physically connect the two machines by plugging in one port of an Ethernet cable to `h1` and the other port to `h2`.

Q35/ What are the interfaces in `h1` and `h2` that are connected through Ethernet cable? What are their corresponding IP addresses?

Solution. *Open Wireshark and start pinging from `h1` to `h2` and vice versa. The used interfaces can be found by checking the traffic on each interface.*

Use the following command on `h1` and `h2` separately to find out their IP addresses:

```
ip addr show
```

Now, to check connectivity, ping `h2` from `h1` and vice versa (**note that before execution of the command, you should turn off the windows firewall if you are using Windows OS**).

Q36/ Explain why you do not receive ping-reply in `h1` and `h2`. What are the commands you need to execute in order to solve this issue?

Solution. *Because they are not in the same subnet, or they do not have valid IP addresses.*

You have to set up a private network between `h1` and `h2`. An example is: `10.2.2.0/24`.

at `h1`:

```
ip addr flush dev <h1 interface>
ip addr add 10.2.2.2/24 dev <h1 interface>
```

at h2:

```
ip addr flush dev <h2 interface>
ip addr add 10.2.2.3/24 dev <h2 interface>
```

Verify the connection of the machines by executing ping command again on the hosts `h1` and `h2`.

4.2 Measuring the Bandwidth of the Communication Link

So far, you have built a point-2-point 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 `h1` and `h2`.

Run `iperf` as server in `h2`.

Q37/ What are the commands you used in `h2` to run it as an `iperf` server? What are the IP address and port of the server?

Solution.

```
iperf -s
```

The IP address is the one used by the interface of `h2` and the port is written in the terminal when the server runs.

Now run the `iperf` client in `h1`.

Q38/ What are the commands you used in `h1` to run it as an `iperf` client?

Solution.

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

Q39/ What is the practical bandwidth of the Ethernet cable?

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

4.3 Sharing Internet among your Friends!

The goal of this subsection is to learn how to share your own Internet access with others who do not. Now, turn on the Wi-Fi interface of the physical machine `h2` and check its connectivity by pinging `google.com`. Note that `h1` still does not have Internet access; you want to make it possible! Power on the LCA2 virtual machine with two network adapters (a NAT and a Bridge) in host `h2`. Open a terminal in the VM and ping `epfl.ch`.

Q40/ Find out which interface is connected to the NAT and which one is connected to the Bridge.

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

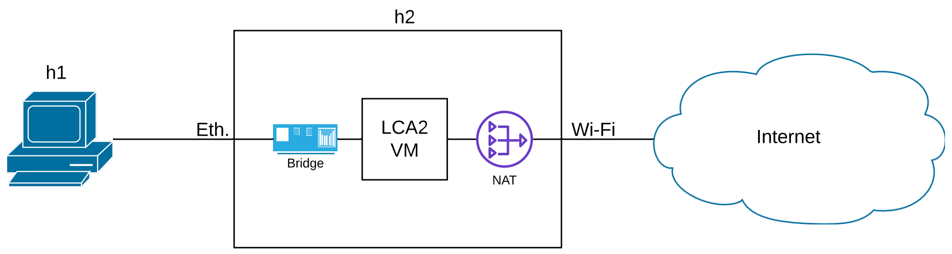

Figure 6: Sharing Internet with a friend!

Use the following command on the VM in `h2` *separately to find out their IP addresses:*

```
ip addr show
```

It shows three adapters: LoopBack (LO), one connected to NAT, and the other is connected to Bridge.

To make the VM interface connected to the bridge be visible by host `h1`, we need to assign an IP address to it.

Q41/ What is a suitable IP address for the bridge interface in the VM? what is the command to be executed in the VM to assign the IP address to this interface?

Solution. *The IP address should be in the same subnet as hosts* `h1` *and* `h2`*.*
At the terminal of VM in `h2`*:*

```
ip addr flush dev <bridge interface in VM of h2>
ip addr add 10.2.2.4/24 dev <bridge interface in VM of h2>
```

Now, verify the connectivity of the bridge interface of VM and `h1` by pinging each other. After connectivity of VM to host `h1` through bridge and Internet through NAT, it is the time for th e final step: route the `h1` traffic via the VM.

Q42/ What are the commands you need to execute at `h1` and VM in `h2` to give `h1` Internet access?

Solution. *At* `h1`*:*

```
ip route add default via <IP address the bride in the VM of h2>
```

At VM in `h2`*:*

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

Verify the Internet connection of `h1` by pinging `google.com`.

Assume a router `r` is the next hop of `h2`. The host `h1` starts to ping `google.com`.

Q43/ If Wireshark is running on `r`, what would be the source IP address of the ICMP packets send by `h1`? Explain the reason.

Solution. *It is the IP address of the Wi-Fi interface of `h2`. As on `h1`, NAT is enabled, all the incoming traffic from `h2` is passed though the NAT in `h2`. According NAT, all the machines behind the NAT are transparent from Internet and it is seen as `h2` is 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 Internet. Suppose a friend of you visits Switzerland and would like to have Internet access; however, the cost of Roaming is too much. 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 and you are responsible for the traffic down/uploaded by your friend.