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 as well as a physical machine in the INF019. 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; later you will connect your virtual network to the Internet using a machine in the INF 019; and finally, you will help Jon and Arya to fix their networking problems when a common enemy, Joffrey, changes the configuration in their network.

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.
In the two last sections you will be required to work with a persistent live linux machine bootable from a **USB key**. If you decide to work with one of our pre-configured USB keys, you should borrow one from Marguerite Delcourt in exchange for a signature. Once you are done using it **you must give it back in order for your lab to be graded**. In case you loose the USB, we will ask you to pay 30chf.

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 1 VS. LAYER 2 VS. LAYER 3 NETWORKING

The aim of this section is to illustrate the difference between networking devices that work at layer 1, layer 2 and layer 3. For this exercise we only consider IPv4 addressing.

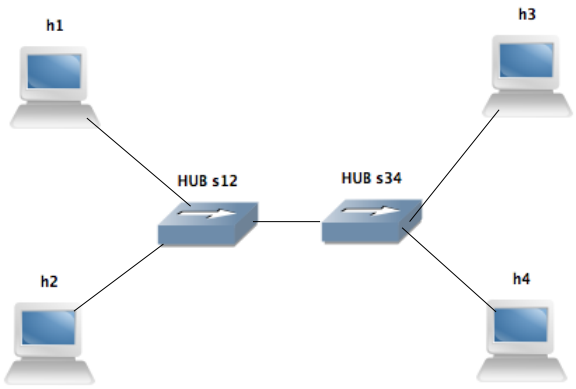

Figure 1: Network configuration with two hubs in the center

2.1 USING HUB AS A NETWORKING DEVICE

A hub is layer-one intermediate system that repeats bits (symbols) received from one of its ports to all other ports. In this section we analyze how it works.

Open a terminal in your VM and run the script `topo1.py`, which should be located in the shared folder on the Desktop. If not, refer to Section 1.2.

```
# python topo1.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.

Run the following two commands to configure `s12` and `s34` as hubs.

```
mininet> sh ovs-ofctl add-flow s12 action=flood
mininet> sh ovs-ofctl add-flow s34 action=flood
```

`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. `h4` should have the IP address 10.0.1.4/24. Check the configuration of the IP addresses for each of the hosts in their respective terminals and correct any wrong configuration.

Q1/ Was there a wrongly configured host? How did you find out?

Solution. *h2 was configured with a wrong subnet mask. ip addr show at h2 or dump on the mininet CLI would help us find out.*

Q2/ Which line of code in the mininet configuration script does this error correspond to?

Solution. *Line 22 in the topo1.py file should be:*
h2 = net.addHost('h2', ip='10.0.0.2/24')
instead of
h2 = net.addHost('h2', ip='10.0.0.2/31')

Q3/ What command did you run to fix the configuration issue?

Solution. *In the h2 terminal: ip addr flush dev h2-eth0; ip addr add 10.0.0.2/24 dev h2-eth0*

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
```

Q4/ What would be another way of doing the same thing?

Solution. *From the h1 terminal, ping 10.0.0.2*

Q5/ Is there any difference between the traffic captured by the four hosts in Wireshark?

Solution. *No, In all machines we can see exactly the same packets*

Q6/ Explain why you see these results.

Solution. *In a hub, the incoming traffic from a port is forwarded to all ports in the hub (except the one it received the packet from) without any type of filtering. A hub is an intermediate system that just amplifies and repeats signals.*

2.2 USING A BRIDGE AS A NETWORKING DEVICE

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

We will first change `s12` to act as a bridge by running the following command.

```
mininet> sh ovs-ofctl add-flow s12 action=normal
```

Note that `s12` and `s34` were originally created as bridges, but we modified them in the previous section to act as hubs by adding flows. You will learn more about flows in Lab 4.

Q7/ If you were to exit Mininet and run `topo1.py` again, what command(s) would you then execute to reach the topology we have now?

Solution. *We need to set s34 as a hub. From the mininet CLI; sh ovs-ofctl add-flow s34 action=flood*

Now, let's test our bridge configuration. Start a Wireshark capture on all four hosts again, and again ping `h2` from `h1`.

Q8/ 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. In `h3` and `h4` we only observed ARP request packets (broadcast)*

Q9/ Explain the results. What is the difference compared to having two hubs?

Solution. *In a bridge, frames are forwarded by searching in the MAC-address table and performing an exact match lookup. If the search finds a match, the frame is forwarded directly to the port given by the MAC-address table. If the frame is not in the table, then the bridge forwards the frame to all ports except the one it received the frame from.*
This is why we only receive broadcast packets on `h3` and `h4` (destination MAC-address of broadcasts is not known), but the actual ICMP traffic we do not see it (the bridge sends traffic directly from `h1` to `h2` and vice versa)

Q10/ Ping from h1 to h4. Observe the traffic captured and explain your findings.

Solution. *We don't see any packets. h1 will use its network mask on h4's IP address and check if they are in the same subnet. As they are not in the same subnet, h1 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.*

Now, focus on the interfaces of `h1, h2` and `h3`. Ping from `h1` to `h3`.

Q11/ Observe the packets captured on `h2` and explain the results.

Solution. *We observe an ARP request for `h3` because `s12` acting as a bridge uses exact matching.*

Q12/ 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.*

2.3 Using a Router as a Networking Device

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 three hosts, a router, and a bridge, as shown in Figure 2.

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

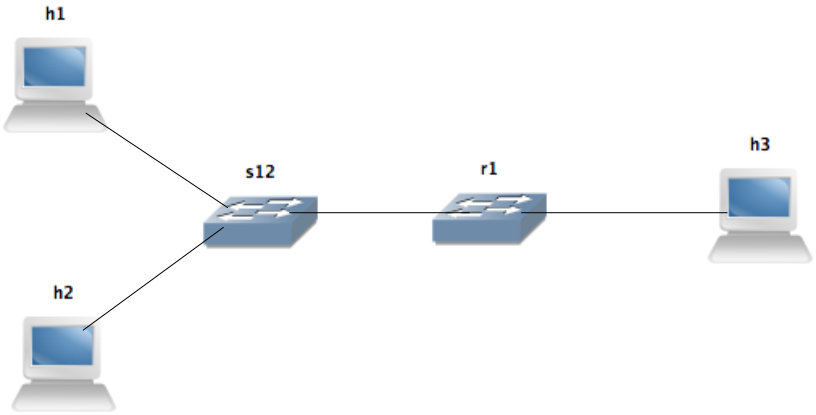

Figure 2: Network configuration with a router and a switch

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. In our case, this also includes the router. A quick way to do this test in Mininet is by running the following command.

```
mininet> pingall
```

Q13/ What is the percentage of dropped packets? What does this correspond to?

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

Q14/ 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.

Q15/ What is the subnet mask used throughout the file?

Solution. *255.255.255.0*

Q16/ 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*

Q17/ 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.

Q18/ What is the command you used at `r1`?

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

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

Q19/ 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`.

Q20/ What are the commands you used to achieve this?

Solution. *ip route del default via 10.0.0.101; ip route add default via 10.0.0.100*

Q21/ 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`.

Q22/ 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.*

Q23/ 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.

Q24/ 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`.

Q25/ 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.*

Q26/ 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 3. `h1` and `h2` are workstations, `r1` is an aggregation router, and `h3` is the perimeter router where we will have our connection to the real world.

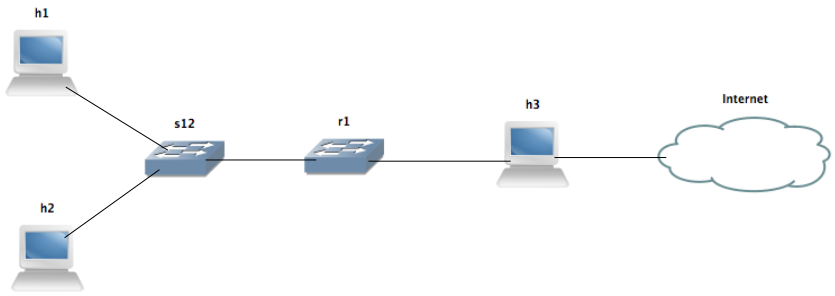

Figure 3: Network configuration with a connection to the real world

Q27/ 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 `h3`'s `h3-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 `h3-eth1` interface of `h3`.
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 `h3`'s `eth1`.
2. Finding a suitable IPv4 address.
3. Setting up the IPv4 address of `h3-eth1` on `h3`.

3.1.1 Bridge between the physical and the virtual interface

For the first step, the process is shown schematically in Figure 4, and we will cover it step by step.

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. Start capturing on each interface and ping `www.google.com` to find out if that interface is being used to access the internet.

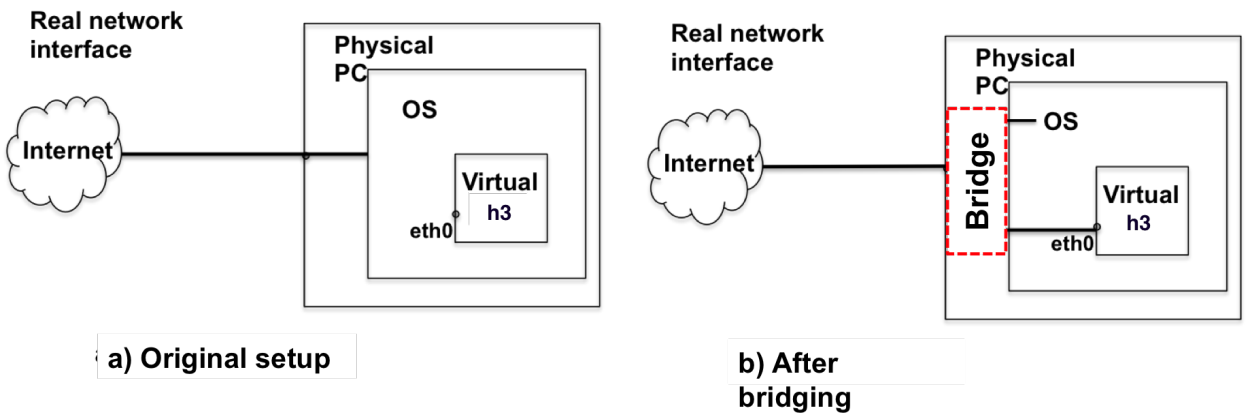

Figure 4: Bridging the network adapter

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

Q28/ What is the IP address of the interface used to access the internet? Do you notice anything special about it?

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.*

Now that we know which interface is used by the VM, the next step is to implement a bridge between it and `h3-eth1`.

In Mininet, we can only implement a bridge with an OVS. To mitigate this, we will add an OVS between `h3` and the internet cloud in Figure 3.

Run the script `topo2_internet.py`. 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 `h3` and run the following command.

```
# dhclient h3-eth1
```

This automatically sets a usable IP address to the `eth1` interface of `h3`, allowing it to access the internet through the bridge we just set up. Test the configuration by pinging Google or EPFL from `h3`.

3.1.2 CHOOSING THE BEST IP ADDRESS FOR H3'S ETH1

The task in this section is to find a suitable IP address for `h3`'s `eth1`. This was basically achieved using the `dhclient` command in the previous section. Here, we will learn to choose an appropriate IP address manually.

Run `ifconfig` and `route -n get default` (in Linux or Mac) or `ipconfig /all` (in Windows) and check your physical IP address, subnet mask and default gateway.

Q29/ Write down your physical IP address, subnet mask and default gateway. According to your network configuration, what is the range of IP addresses that we could use for the virtual adapter?

Solution. *Example output:*
IP Address: `128.179.198.241`
Network Mask: `255.255.255.0`
Default Gateway: `128.179.198.1`
So, in principle we can choose anything in the `128.179.198.0/24` *network except* `128.179.198.1` *and* `128.179.198.241` *assuming there is no other device using an IP address on that segment. Note that in this case* `128.179.198.0` *is the network address and* `128.179.198.255` *is the broadcast address, and therefore cannot be used as valid IP addresses.*

Q30/ How could you find the current non-used IP addresses in your LAN? Is it safe to take any of them?

Solution. *One solution would be by pinging the broadcast IP address, and then check in the ARP table all rows that have a complete IP - MAC address entry. Nonetheless, most network managers would not recommend this approach as it has high consumption of network resources. The only safe IP address you can take is your physical-adapter's IP address, any other IP address is susceptible of being allocated at any given moment by the DHCP service running on the EPFL network or on the network where you are doing the lab.*

Q31/ If you are doing this exercise outside EPFL (e.g. at your home), you will most likely get a private IP address and default gateway (e.g. `192.168.x.x` or `10.x.x.x`). Can you use these private IP addresses on `h3`'s `eth1` to do NAT? Explain why.

Solution. *Yes we can use it, it is called double NAT. The private IP address would behave as a "public" IP address from the Mininet point of view, and NAT would work as expected. In the outside world (from Mininet's perspective) there is another NAT box (e.g. your home ADSL router) that translates the private IP address you received into a valid public IP address in order to reach the internet. This is the reason why the industry is not in a rush towards IPv6, and they prefer to do double, triple or quadruple NATs.*

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.

First let's see what happens when `h1`/`h2` access the Internet with their native IP address. Monitor with Wireshark the interface `eth1` of `h3`. From `h1`/`h2`, ping to Google and its IP address.

```
# ping -c 5 www.google.com
```

```
# ping -c 5 172.217.18.100
```

Q32/ Analyze the packets coming from h1/h2 and explain why you are unable to reach Google.

Solution. *In interface eth0 of h3 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 h3, 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.

Q33/ Propose the `iptables -t nat` command you need to properly configure NAT in h3.

Solution.

```
# iptables -t nat -A POSTROUTING -o h3-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 h3, while capturing eth0 and eth1 traffic on h3 using Wireshark. Explore the difference in the traffic on both cases.

Q34/ When doing `traceroute` from h2, what is the difference in the packets captured on h3'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 h3.*

Q35/ Focus on the `traceroute` from h3. 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.*

Q36/ 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 h3, while capturing the traffic on h3 (both on eth0 and eth1) using Wireshark. Explore the difference in the traffic in both cases.

Q37/ What is the difference in the request ICMP packets captured between packets sent from h1 and packets sent from h3 when capturing on the exit interface of each?

Solution. *The ICMP query ID is different.*

Q38/ 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 HAVING INTERNET ACCESS THROUGH A MACHINE IN INTERNET ENGINEERING WORKSHOP (IEW)/INF019

Up to now, you were accessing the Internet by directly connecting your laptop to a WiFi access point (AP) or Ethernet socket at your home or at EPFL. In this section, you will access the Internet by connecting your laptop to a computer in INF019 using an Ethernet cable. The goal of this section is to give you a feel about the real communication networks. The communication network in IEW is very simple and small, but is probably enough to give you a glance of real network devices and networking between them. You can find the schema of the communication netowrk in IEW here.

4.1 SETTING UP THE MININX MACHINE

In this section we will work with a live linux machine bootable from a USB key that can save the state of the machine. You have three options:

1. Use one of our configured USB keys: it is already set up as a bootable live linux machine with all the right settings for this lab. You need to give it back in order to be graded.
2. Configure your own USB keys using our workstations at the back of INF019: you need a USB 3.0 of at least 32Gb. Plug your USB, click on createkey and unplug your USB. This way you can keep your key after the lab.
3. Configure your own USB yourself: this part is tricky, it can be done at your own risks and it will not be supported by our TAs. Nevertheless, you can have a look at the following french tutorial.

Go to a lab computer from INF019, turn on the USB hub, plug the USB key and turn on the computer. Press Enter to allow the computer to boot on the USB live session: this session works on the machine's RAM and writes the state changes to the USB before shutting down. Change the session password: in a terminal type `passwd`.
In order to shut down, press on the shutdown button and wait for it to be turned off before removing the USB key. This step will save the state of the machine on the USB so that you can continue the lab later on a different machine of INF019 with the configurations you set up. Set up the internet connectivity:

1. Plug an ethernet cable from the computer to an ethernet entry on the shelf above the computer.
2. Check on which interface you are connected: the white adapter has a MAC address which you use to map it to the used interface by looking at `ifconfig -a`.
3. You need to bring this interface up.
4. Give the interface a valid ip address.
5. Test your connectivity: open the web browser and try to access the EPFL website.

Q39/ What is the command you used for steps 3 and 4 of the above procedure?
Solution. *`ip link set dev <interface> up`*

`dhclient <interface>`

4.2 CONNECTING THE LAPTOP TO INTERNET VIA THE MININX MACHINE

To start with, please disable the WiFi interface of your laptop and plug one end of an Ethernet cable into your laptop and the other in the remaining available white interface of the lab computer. Now, configure the connected network interfaces of these two machines (your laptop and the computer from INF019) such that you can ping the lab computer from your laptop and your laptop from the lab computer. To be able to ping these machines with each other, you will first need to assign ip addresses to these two connected interfaces.

Please choose a subnetwork (in private ipv4 address space) that is not yet used in this setting and assign two different ip addresses to these machines.

Q40/ To be able to ping between your laptop and the computer in the lab, please write all the commands you executed on the computer in the lab?

Solution. *The cable is connected on interface* `eth1` *(could be* `eth0` *depending on which USB port you used), we bring up the interface:* `ip link set dev eth1 up` *and we check its ip address (in this case 192.168.200.5/24), we remove it:* `sudo ip addr del 192.168.200.5/24 dev eth1` *and set a new private one we choose:* `sudo ip addr add 10.0.0.2/24 dev eth1`.

Q41/ To be able to ping between your laptop and the computer in the lab, please write all the commands you executed on your laptop?

Solution. *First we find out which interface we are using by looking at* `ifconfig -a`*, in this case it is* `en7` *but it will be different on a different laptop. We give it a valid ip address in the same network as the lab computer's* `eth1` *interface:* `sudo ip addr add 10.0.0.3/24 dev en7`.

Now, as you are able to ping these two machines, the next goal is to be able to ping a machine on the Internet. For this, we will use the other interface of the machine in the lab, which is connected to the Internet. This means, your machine will be connected to the machine in the lab, which, in turn, is connected to the Internet with the other interface.

Now, configure the lab computer and your computer such that you can ping Google DNS server `8.8.8.8`.

Q42/ To do the above, please write all the commands you executed on the lab computer?

Solution. *sudo echo 1 > /proc/sys/net/ipv4/ip_forward*

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

Q43/ To do the above, please write all the commands you executed on your laptop?

Solution. *sudo ip route add default via 10.0.0.2 dev en7*

Once you successfully tested pinging to `8.8.8.8` ip address from your laptop, can you, try pinging `google.com`? If you have not yet specified an address of a DNS server for your laptop, the next thing is to do that. Once done, please, try to ping `google.com` and you should now be able to ping it.

To find the ip address of a DNS server, you can either use the DNS server address used by the lab computer (on the Interface that is connected to the Internet) or you can use Google's public DNS server (8.8.8.8).

Q44/ How did you configure the DNS server on your laptop?

Solution. *Add a DNS server ip address in /etc/resolve.conf.*

The command can be:

sudo echo nameserver 8.8.8.8 > /etc/resolv.conf

If everything is well configured, you should now be able to access the Internet from your laptop connecting through the lab computer. Please visit a web server (`facebook.com`, `google.com`) from your browser to confirm it.

In section 3, you have already connected your mininet hosts to the Internet. Follow the steps from section 3 or if you have saved the commands in a script for section 3, execute them so that you can have the same topology as you had at the end of section 3. You should now be able to access the Internet from hosts `h1`, `h2`, and `h3`.

Q45/ Do the `traceroute` for `epfl.com` from mininet host `h1` and paste the results below. ***Solution.***

5 TROUBLESHOOTING

As the title for this section suggests, in networking things will not always work out as expected. Before starting to work in this section, you will have to execute some executables. These executables will put a number of PCs into a problematic situation where something doesn't work. Your task is to find out what the problem is and propose a solution.

You should not perform any debug command or Wireshark capture in `h3`. Assume `h3` is a router controlled by your Internet service provider (ISP) and you don't have access to it..

5.1 ABOUT GRADING THIS SECTION

The points given for your answer mostly depend on your explanations about how you located the problem!, so describe precisely your steps to locate and diagnose the fault. You should use the scientific method when answering the problem. The methodology you should use is the following:

1. Pose a hypothesis
2. Run experiments to validate the hypothesis
3. If validation is OK exit, else loop (go back to 1. by posing another hypothesis)

In your answer you should write down all steps. Specially, you should also write down all hypotheses that later proved to be wrong. We want to see the path you took to reach your final conclusion!

More specifically for this Lab: What were the commands you executed to get there? Up to which point did the system work as expected? What were the actions that never got executed but were expected? What was the packet that did not reach its destination? Where and why did it get dropped/lost?

5.2 WE WERE HACKED!!!

Jon and Arya are roommates and close friends. Both of them are connected to the Internet through the same home router thus the same ISP. The configuration is the same as described in Figure 5, where `h1` is Jon's computer and `h2` is Arya's computer, `r1` is the home router, shared between Jon and Arya, and `h3` is the ISP's router.

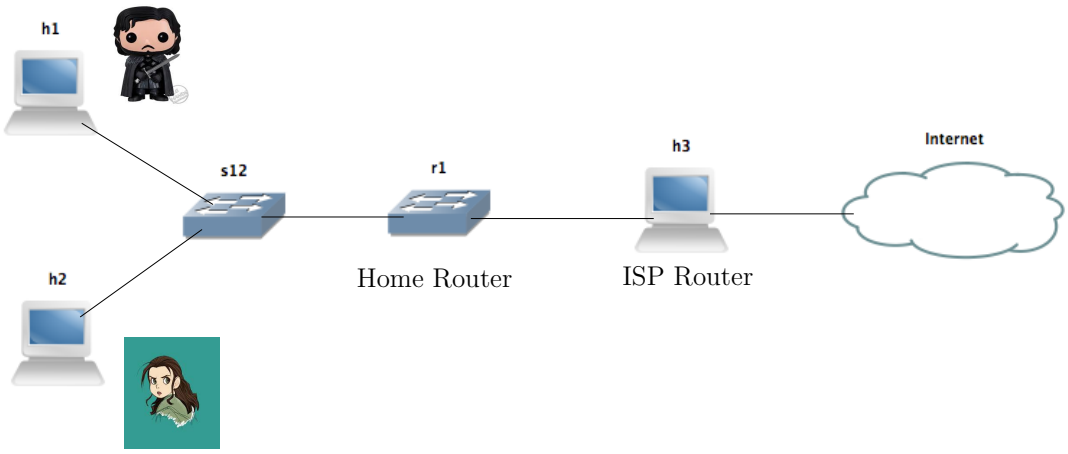

Figure 5: Troubleshooting configuration

Jon and Arya are foreign students at EPFL and they use Facebook a lot to communicate with their relatives in their home town, Winterfell. None of them is an expert in computers and networking. They have a common enemy, Joffrey, who plays jokes and is a Computer Science student and network expert. Joffrey told Jon and Arya that he hacked their computers (`h1` and `h2`) as well their home router (`r1`) and that they would never navigate through Facebook again.

To simulate Joffrey's malicious attack, we prepared an executable for `h1`, `h2` and `r1`. You should download the executables from the course web page on moodle. Unzip and extract the corresponding files:

```
# unzip lab2-execs.zip
```

We recommend you to extract all executables to your shared folder between host and virtual machines.

Let's start with Jon and `h1`. Check that you are on host `h1`. Navigate to the `lab2-execs` folder and execute the executable for `h1` on `h1`. Make sure you **run the executable as a superuser in Linux**.

```
# cd lab2-execs
# sudo ./h1_hack
```

Note that if you stop the simulation in Mininet, then you need to redo the previous section and make sure that `h1` and `h2` have a proper internet connection, then reload the script on `h1`.

Also, if you do not manage to solve one of the questions, you may want to restart Mininet, redo the previous section and make sure that all hosts have an internet connection, and then move on to the next question. This is where using scripts to save the commands you performed to achieve a working configuration would help a great deal. Ideally, solving a question means undoing the changes brought about by the hack, so you can start solving any of the questions in this section from the stable working configuration of Section 3.

Now, your mission is to help Jon find out why he cannot navigate to Facebook anymore. You are only allowed to have Wireshark captures and use debug commands in `h1`.

Open firefox on `h1` by typing `firefox &` in the corresponding terminal.

Q46/ Are there any problems in Jon's PC (`h1`)? Enumerate them (if any) and write down how you find them and how you fix them (including typed commands).

Solution. *We notice that Jon is sending DNS queries but not receiving replies, which means that the nameserver is wrongly configured to go to `1.9.9.2`. Fixing that solves the problem:*
`echo nameserver 8.8.8.8 > /etc/resolv.conf`

After you fix the issue, confirm that Jon can now safely log into Facebook to update his status to "I know nothing about TCP/IP".

Now repeat the same procedure for Arya: execute the `h2_hack` executable on the `h2` terminal. Again you can use Wireshark on `h2` to capture packets.

Q47/ Open Firefox on `h2` and try to navigate to Facebook. Are there any problems in Arya's PC (`h2`)? Enumerate them (if any) and write down how you find them and how you fix them (including typed commands).

Solution. *The problem is that `h2` has a wrongly configured IP address, so we need to change it back to the correct one, and add the default gateway that was removed as a result of flushing the ip address. This can be discovered by observing that no packets are captured on `h2-eth0` and `h3`. To fix the issue:*

```
ip addr del 1.2.1.2/24 dev h2-eth0
ip addr add 10.0.0.2/24 dev h2-eth0
ip route add default via 10.0.0.100
```

While you were solving these issues on `h1` and `h2`, Joffrey had enough time to hack `r1`. To simulate this, execute the `r1_hack` executable on `r1`. Doing this, Joffrey managed to stop both Jon and Arya from accessing Facebook again. Your job is to get both of them Facebook access again. Hurry though, Winter is Coming!

For this question, you may perform debugging and capture packets using Wireshark on `h1`, `h2`, and `r1`. DO NOT TOUCH `h3`!

Q48/ Enumerate the problems (if any) that were stopping Jon (`h1`) from accessing Facebook. How did you find them and fix them?

Solution. *All packets sent from from `h1` to facebook IP addresses through `r1` have a gateway assigned to them which routes them back where they came from. This can be observed by observing Wireshark capture on `r1`. Fixing this allows `h1` to access the internet.*
However, when Jon tries to access Facebook, he is instead rerouted to the SBB website (may not show the webpage on firefox due to security certificates). The problem is some iptables rules on `r1` that change the destination of packets from Jon accessing Facebook to SBB. They can be seen by running `iptables -t nat -v -L -n`, and deleted by running `iptables -t nat -F`.

Q49/ Enumerate the problems (if any) that were stopping Arya (`h2`) from accessing Facebook. How did you find them and fix them?

Solution. *The only issue with Arya is the routing table rules in `r1` that was also an issue with Jon. After clearing this issue by removing these entries in the routing table (one by one), Arya will be able to access the internet (and Facebook), but Jon still had to fix the iptables issue as explained above.*

Q50/ Create a script to unhack `r1` and paste its content below.

Solution. *ip route del 204.15.20.0/22 dev r1-eth0*
ip route del 69.63.176.0/20 dev r1-eth0
ip route del 66.220.144.0/20 dev r1-eth0
ip route del 66.220.144.0/21 dev r1-eth0
ip route del 69.63.184.0/21 dev r1-eth0
ip route del 69.63.176.0/21 dev r1-eth0
ip route del 74.119.76.0/22 dev r1-eth0
ip route del 69.171.255.0/24 dev r1-eth0
ip route del 173.252.64.0/18 dev r1-eth0
ip route del 69.171.224.0/19 dev r1-eth0
ip route del 69.171.224.0/20 dev r1-eth0
ip route del 103.4.96.0/22 dev r1-eth0
ip route del 69.63.176.0/24 dev r1-eth0
ip route del 173.252.64.0/19 dev r1-eth0
ip route del 173.252.70.0/24 dev r1-eth0
ip route del 31.13.64.0/18 dev r1-eth0
ip route del 31.13.24.0/21 dev r1-eth0
ip route del 66.220.152.0/21 dev r1-eth0
ip route del 66.220.159.0/24 dev r1-eth0
ip route del 69.171.239.0/24 dev r1-eth0
ip route del 69.171.240.0/20 dev r1-eth0

ip route del 31.13.64.0/19 dev r1-eth0
ip route del 31.13.64.0/24 dev r1-eth0
ip route del 31.13.65.0/24 dev r1-eth0
ip route del 31.13.67.0/24 dev r1-eth0
ip route del 31.13.68.0/24 dev r1-eth0
ip route del 31.13.69.0/24 dev r1-eth0
ip route del 31.13.70.0/24 dev r1-eth0
ip route del 31.13.71.0/24 dev r1-eth0
ip route del 31.13.72.0/24 dev r1-eth0
ip route del 31.13.73.0/24 dev r1-eth0
ip route del 31.13.74.0/24 dev r1-eth0
ip route del 31.13.75.0/24 dev r1-eth0
ip route del 31.13.76.0/24 dev r1-eth0
ip route del 31.13.77.0/24 dev r1-eth0
ip route del 31.13.96.0/19 dev r1-eth0
ip route del 31.13.66.0/24 dev r1-eth0
ip route del 173.252.96.0/19 dev r1-eth0
ip route del 69.63.178.0/24 dev r1-eth0
ip route del 31.13.78.0/24 dev r1-eth0
ip route del 31.13.79.0/24 dev r1-eth0
ip route del 31.13.80.0/24 dev r1-eth0
ip route del 31.13.82.0/24 dev r1-eth0
ip route del 31.13.83.0/24 dev r1-eth0
ip route del 31.13.84.0/24 dev r1-eth0
ip route del 31.13.85.0/24 dev r1-eth0
ip route del 31.13.86.0/24 dev r1-eth0
ip route del 31.13.87.0/24 dev r1-eth0
ip route del 31.13.88.0/24 dev r1-eth0
ip route del 31.13.89.0/24 dev r1-eth0
ip route del 31.13.90.0/24 dev r1-eth0
ip route del 31.13.91.0/24 dev r1-eth0
ip route del 31.13.92.0/24 dev r1-eth0
ip route del 31.13.93.0/24 dev r1-eth0
ip route del 31.13.94.0/24 dev r1-eth0
ip route del 31.13.95.0/24 dev r1-eth0
ip route del 69.171.253.0/24 dev r1-eth0
ip route del 69.63.186.0/24 dev r1-eth0
ip route del 31.13.81.0/24 dev r1-eth0
ip route del 179.60.192.0/22 dev r1-eth0
ip route del 179.60.192.0/24 dev r1-eth0
ip route del 179.60.193.0/24 dev r1-eth0
ip route del 179.60.194.0/24 dev r1-eth0
ip route del 179.60.195.0/24 dev r1-eth0
ip route del 185.60.216.0/22 dev r1-eth0
ip route del 45.64.40.0/22 dev r1-eth0
ip route del 185.60.216.0/24 dev r1-eth0
ip route del 185.60.217.0/24 dev r1-eth0
ip route del 185.60.218.0/24 dev r1-eth0
ip route del 185.60.219.0/24 dev r1-eth0
ip route del 129.134.0.0/16 dev r1-eth0

ip route del 157.240.0.0/16 dev r1-eth0
ip route del 157.240.8.0/24 dev r1-eth0
ip route del 157.240.0.0/24 dev r1-eth0
ip route del 157.240.1.0/24 dev r1-eth0
ip route del 157.240.2.0/24 dev r1-eth0
ip route del 157.240.3.0/24 dev r1-eth0
ip route del 157.240.4.0/24 dev r1-eth0
ip route del 157.240.5.0/24 dev r1-eth0
ip route del 157.240.6.0/24 dev r1-eth0
ip route del 157.240.7.0/24 dev r1-eth0
ip route del 157.240.9.0/24 dev r1-eth0
ip route del 157.240.10.0/24 dev r1-eth0
ip route del 157.240.16.0/24 dev r1-eth0
ip route del 204.15.20.0/22 dev r1-eth0
ip route del 69.63.176.0/20 dev r1-eth0
ip route del 69.63.176.0/21 dev r1-eth0
ip route del 69.63.184.0/21 dev r1-eth0
ip route del 66.220.144.0/20 dev r1-eth0
ip route del 69.63.176.0/20 dev r1-eth0
iptables -t nat -F