

Name 1:

Name 2:

COMPUTER NETWORKING

LAB EXERCISES (TP) 2

L1 v.s. L2 v.s. L3, NAT AND TROUBLESHOOTING

October 14, 2016

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; later you will help Jon and Arya to fix their networking problems when a common enemy, Joffrey, changes the configuration in their network. Finally you will practice reading TCP-IP headers to collect and correlate information.

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 26th, 23:59:59**

1.2 SETTING UP VIRTUAL MACHINE

In this Lab, you will work with the same virtual machine that you created in Lab 1. Copy the **lab2** 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 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.

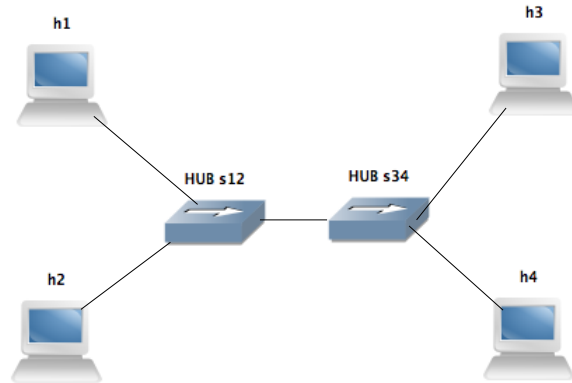


Figure 1: Network configuration with two hubs in the center

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?

[A1]

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

[A2]

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

[A3]

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?

[A4]

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

[A5]

Q6/ Explain why you see these results.

[A6]

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 s34 to act as a bridge by running the following command.

```
mininet> sh ovs-ofctl add-flow s34 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?

[A7]

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`.

[A8]

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

[A9]

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

[A10]

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.

[A11]

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.

[A12]

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.

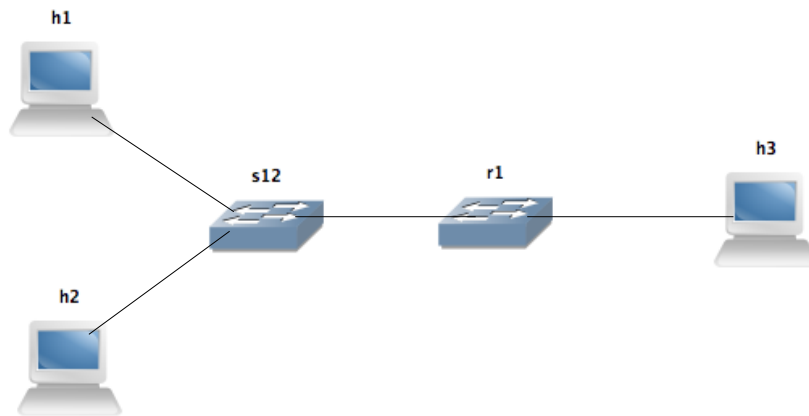


Figure 2: Network configuration with a router and a switch

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

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?

[A13]

Q14/ Which hosts are unable to reach one another?

[A14]

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?

[A15]

Q16/ What are the interfaces and respective IP addresses of the router `r1`?

[A16]

Q17/ Can you spot any misconfigurations in the file?

[A17]

The first issue we notice in the file is that IP forwarding has been disabled in the router `r1`. We will enable it and attempt the reachability test again.



Q18/ What is the command you will use to enable IPv4 forwarding on `r1`?

[A18]

Q19/ What is the percentage of dropped packets now in the reachability test? Which hosts are still unable to ping each other?

[A19]

The next issue in the file is the default gateway of `h2`, which is set to `10.0.0.101`. This IP address does not exist in our network. Delete this configuration and set a proper default gateway through the terminal of `h2`.



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

[A20]

Q21/ What are the results of the reachability test now?

[A21]

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`?

[A22]

Q23/ What is the purpose of such changes?

[A23]

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.

[A24]

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.

[A25]



Q26/ Write down **one single command** that fixes the problem (namely pinging from `h1` and `h2` to `h3`), and specify the PC where you need to apply it.

[A26]

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.

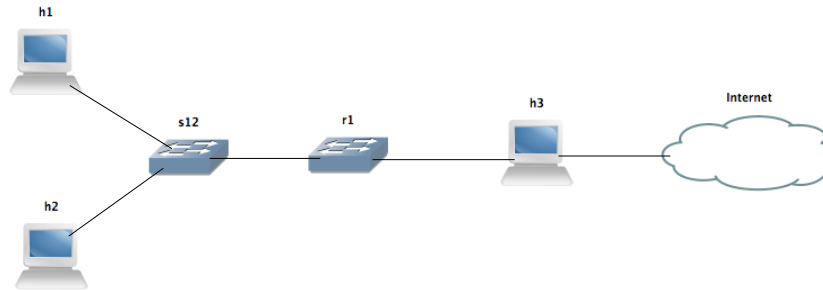


Figure 3: Network configuration with a connection to the real world



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.

[A27]

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.

```

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

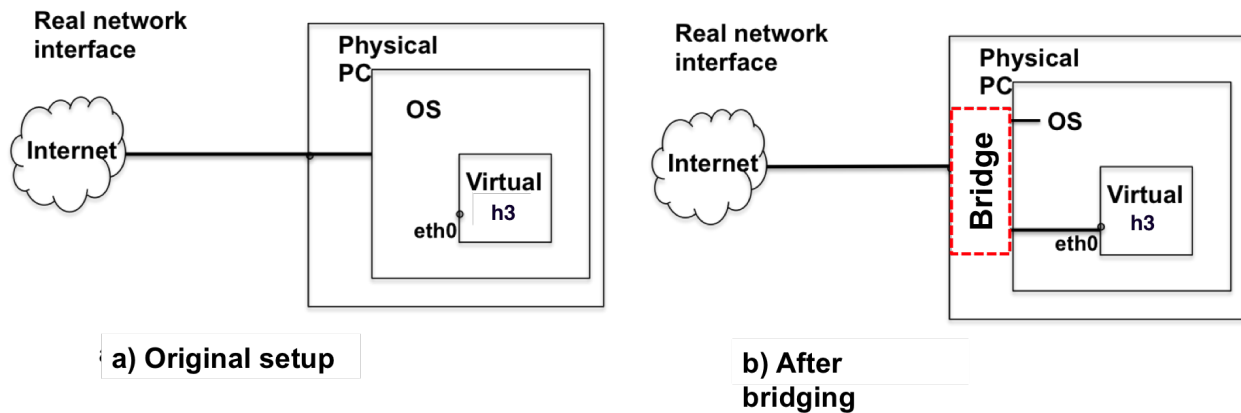



Figure 4: Bridging the network adapter



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



[A28]

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 add an extra OVS that we will use to perform the bridge.

From Mininet, execute the following command.

```
mininet> sh ovs-vsctl add-port s3 ethX
```

Replace `X` with the number of 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 do 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?

[A29]

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

[A30]



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.

[A31]

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 to 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.

[A32]

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

[A33]

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`?

[A34]



Q35/ Focus on the `traceroute` from `h3`. What is the difference in the packets as compared to `h2`?

[A35]



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?

[A36]

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?

[A37]



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?

[A38]

4 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 a script. This script 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..

4.1 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?

4.2 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.

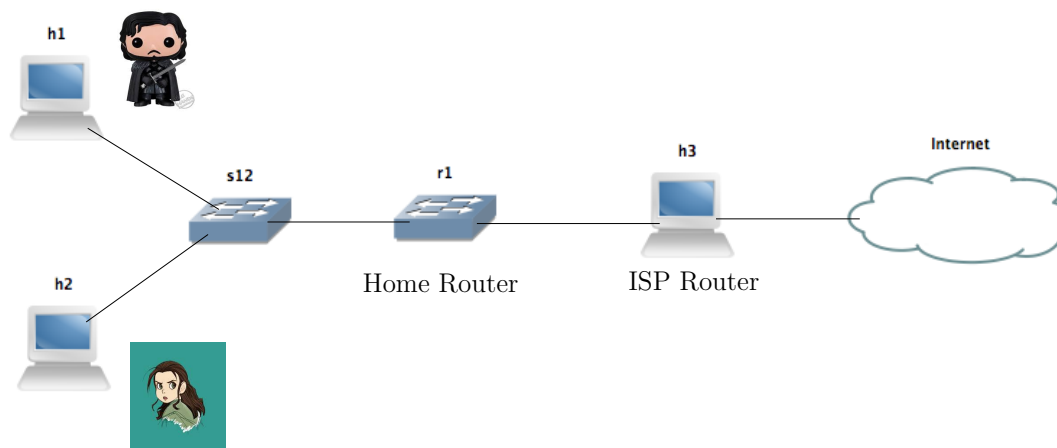


Figure 5: Troubleshooting configuration

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 a script for h1, h2 and r1. You should download the scripts from the course web page on moodle. Unzip and extract the corresponding file:

```
# tar -xzvf lab2-scripts.tar.gz
```

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

Let's start with Jon and h1. Check that on h1 you have a h1.hack directory with the following content:

```
# ls lab2-scripts
h1.hack
```

Navigate to the lab2-scripts folder and execute the script for h1 on h1. Make sure you **run the script as a superuser in Linux**.

```
# cd lab2-scripts
# 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.



Q39/ 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.

[A39]

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` script on the `h2` terminal. Again you can use Wireshark on `h2` to capture packets.



Q40/ 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.

[A40]

While you were solving these issues on `h1` and `h2`, Joffrey had enough time to hack `r1`. To simulate this, execute the `r1.hack` script 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 Wireshark scripts on `h1`, `h2`, and `r1`. DO NOT TOUCH `h3`!



Q41/ Enumerate the problems (if any) that were stopping Jon (`h1`) from accessing Facebook. How did you find them and fix them?

[A41]

Q42/ Enumerate the problems (if any) that were stopping Arya (`h2`) from accessing Facebook. How did you find them and fix them?

[A42]

5 READING IPV4 AND IPV6 PACKET HEADERS

In this section, you will observe IPv4 and IPv6 connections in different contexts. Your task is to answer the questions based only on the traces presented for each connection. You no longer need Mininet for the remainder of this lab. However, you will use your VM for some of the questions.

5.1 IPV4 PACKET HEADERS

Jon is sitting in front of `lrcpc3` workstation and connects to `smartgrid.epfl.ch`. An ethical hacker has read all the frames passing on the network. Here are two packets resulting from this activity:

```

ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 2 arrived at 13:45:24.14
ETHER: Packet size = 74 bytes
ETHER: Destination = 0:0:d:2:ff:e2
ETHER: Source       = 0:0:f3:a4:43:ee
ETHER: Ethertype = 0800
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes
IP: Type of service = 0x00
IP:     xxx. .... = 0 (precedence)
IP:     ...0 .... = normal delay
IP:     .... 0... = normal throughput
IP:     .... .0.. = normal reliability
IP: Total length = 60 bytes
IP: Identification = 2947
IP: Flags = 0x0
IP:     .0.. .... = may fragment
IP:     ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 64 seconds/hops
IP: Protocol = 17
IP: Header checksum = c2ba
IP: Source address = 128.178.156.17
IP: Destination address = 128.178.25.8, IP:   No options
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 1304
UDP: Destination port = 53 (DNS)
UDP: Length = 40
UDP: Checksum = B281
UDP:
DNS: ----- DNS: -----
DNS:
DNS: ""
DNS:

```

```

ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 3 arrived at 13:45:24.85
ETHER: Packet size = 202 bytes
ETHER: Destination = 0:0:c0:b8:c2:8d, Western Digital
ETHER: Source       = 0:0:c:2:78:36, Cisco
ETHER: Ethertype = 0800
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes
IP: Type of service = 0x00
IP:     xxx. .... = 0 (precedence)
IP:     ...0 .... = normal delay
IP:     .... 0... = normal throughput
IP:     .... .0.. = normal reliability
IP: Total length = 188 bytes
IP: Identification = 38579

```



```

IP:   Flags = 0x0
IP:       .0.. .... = may fragment
IP:       ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 58 seconds/hops
IP:   Protocol = 17
IP:   Header checksum = 3d0a
IP:   Source address = 128.178.25.8,
IP:   Destination address = 128.178.156.17,
IP:   No options
IP:
UDP:   ----- UDP Header -----
UDP:
UDP:   Source port = 53
UDP:   Destination port = 1304
UDP:   Length = 168
UDP:   Checksum = 0000
UDP:
DNS:   ----- DNS: -----
DNS:
DNS:   ""
DNS:

```



Q43/ What is the purpose for this packet exchange?

[A43]

Q44/ Within each packet, what accounts for the difference in the *length* field in different headers?

[A44]

Q45/ How many routers would you say are there between Jon's PC and the DNS server?

[A45]

Q46/ Repeat the packet exchange on your VM and observe the packets captured on Wireshark. What fields do you expect to be different?

[A46]

5.2 IPV6 PACKET HEADERS

Arya is sitting in front of lrcpc3 workstation and visits the website `www.ethz.ch`. An ethical hacker has read all the frames passing on the network. Here are the first two packets resulting from this activity:

```

ETHER:   ----- Ether Header -----
ETHER:
ETHER:   Packet 1 arrived at 11:55:22.298
ETHER:   Packet size = 86 bytes
ETHER:   Destination = 33:33:ff:01:00:01
ETHER:   Source = 3c:07:54:3e:ab:f2

```

```

ETHER:  Ethertype = 0x86dd
ETHER:
IP:  ----- IP Header -----
IP:
IP:  Version = 6
IP:  Traffic class = 0x00000000
IP:  .... 0000 00.. .... .... .... .... = Default Differentiated Service Field
IP:  .... .... ..0. .... .... .... .... = No ECN-Capable Transport (ECT)
IP:  .... .... ...0 .... .... .... .... = No ECN-CE
IP:  .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
IP:  Payload length = 32
IP:  NextHeader= 58
IP:  Hop limit= 255
IP:  Source address = 2001:620:618:197:1:80b2:97c0:1
IP:  Destination address = ff02::1:ff01:1
IP:
ICMPv6:  ----- ICMPv6 Header -----
ICMPv6:
ICMPv6:  Type = 135
ICMPv6:  Code=0
ICMPv6:  Checksum = 0xb199 [correct]
ICMPv6:  Reserved = 00000000
ICMPv6:  Target Address=2001:620:618:197:1:80b2:9701:1
ICMPv6:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 11:55:22.306
ETHER:  Packet size = 86 bytes
ETHER:  Destination = 3c:07:54:3e:ab:f2
ETHER:  Source = 00:08:e3:ff:fc:50
ETHER:  Ethertype = 0x86dd
ETHER:
IP:  ----- IP Header -----
IP:
IP:  Version = 6
IP:  Traffic class = 0x000000e0
IP:  .... 1110 00.. .... .... .... .... = Class Sector 7 Differentiated Service Field
IP:  .... .... ..0. .... .... .... .... = No ECN-Capable Transport (ECT)
IP:  .... .... ...0 .... .... .... .... =No ECN-CE
IP:  .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
IP:  Payload length = 32
IP:  NextHeader=58 (ICMPv6)
IP:  Hop limit= 255
IP:  Source address = 2001:620:618:197:1:80b2:9701:1
IP:  Destination address = 2001:620:618:197:1:80b2:97c0:1
IP:
ICMPv6:  ----- ICMPv6 Header -----
ICMPv6:
ICMPv6:  Type = 136
ICMPv6:  Code=0
ICMPv6:  Checksum = 0xe3f8 [correct]
ICMPv6:  Flags=0xe0000000
ICMPv6:  1... .... .... .... .... .... = Router: Set
ICMPv6:  .1.. .... .... .... .... .... = Solicited: Set
ICMPv6:  ..1. .... .... .... .... .... = Override: Set
ICMPv6:  ...0 0000 0000 0000 0000 0000 0000 0000 = Reserved: 0
ICMPv6:  Target Address=2001:620:618:197:1:80b2:9701:1
ICMPv6:

```



Q47/ What are these packets used for in the exchange? What do Types 135 and 136 in the ICMPv6 header represent?

[A47]

Q48/ By observing each layer header separately, how do we know that the packet exchange uses IPv6 in each of them?

[A48]

Shortly afterwards, we observe the following exchange.

```
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 3 arrived at 11:55:23.186
ETHER: Packet size = 91 bytes
ETHER: Destination = 00:08:e3:ff:fc:50
ETHER: Source      = 3c:07:54:3e:ab:f2
ETHER: Ethertype = 0x86dd
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 6
IP: Traffic class = 0x00000000
IP:      .... 0000 00.. .... = Default Differentiated Service Field
IP:      ....      ..0. .... = No ECN-Capable Transport (ECT)
IP:      ....      ...0 .... = No ECN-CE
IP: .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
IP: payload length = 37
IP: Next header = 17 (UDP)
IP: Hop limit = 255
IP: Source address = 2001:620:618:197:1:80b2:97c0:1
IP: Destination address = 2001:620:618:1a6:1:80b2:a66a:1
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 63736
UDP: Destination port = 53 (DNS)
UDP: Length = 37
UDP: Checksum = 0x4cd7
UDP:
DNS: ----- DNS Header -----
DNS:
DNS: Transaction ID=0x2f34
DNS: Flags= 0x100
DNS:  0...      .... = Message is a query
DNS:  .000 0...      .... = Standard query
DNS:      .... ..0. .... = Message not truncated
DNS:      .... ...1 .... = Do query recursively
DNS:      ....      .0.. .... = reserved
DNS:      ....      ...0 .... = Non-authenticated data unacceptable
DNS: Queries
DNS:      ..... Name=www.ethz.ch
DNS:      ..... Record Type= AAAA (IPv6 address)
```

```

DNS:      ..... Class = IN (0x0001)
DNS:

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 4 arrived at 11:55:23:191
ETHER:  Packet size = 331 bytes
ETHER:  Destination = 3c:07:54:3e:ab:f2
ETHER:  Source      = 00:08:e3:ff:fc:50
ETHER:  Ethertype = 086dd
ETHER:
IP:  ----- IP Header -----
IP:
IP:  Version = 6
IP:  Traffic class = 0x00000000
IP:  .... 0000 00.. .... = Default Differentiated Service Field
IP:  .... ..0. .... = No ECN-Capable Transport (ECT)
IP:  .... ...0 .... = No ECN-CE
IP:  .... 0000 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
IP:  Payload length = 277
IP:  Next header = 17 (UDP)
IP:  Hop limit = 60
IP:  Source address = 2001:620:618:1a6:1:80b2:a66a:1
IP:  Destination address = 2001:620:618:197:1:80b2:97c0:1
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 53
UDP:  Destination port = 63736
UDP:  Length = 277
UDP:  Checksum = 0x4a49
UDP:
DNS:  ----- DNS Header -----
DNS:
DNS:  Flags = 0x8180
DNS:  1... .. = Message is a response
DNS:  .000 0... .. = Standard query
DNS:  .... .0.. .... = Server not authoritative
DNS:  .... ..0. .... = Message not truncated
DNS:  .... ...1 .... = Do query recursively
DNS:  .... .... 1... .. = Recursion available
DNS:  .... .... .0.. .... = reserved
DNS:  .... .... ..0 .... = Answer/authority portion not authenticated
DNS:  .... .... ...0 .... = Non-authenticated data unacceptable
DNS:  .... .... .... 0000 = No error
DNS:  Queries
DNS:  ..... Name=www.ethz.ch
DNS:  ..... Record Type= AAAA (IPv6 address)
DNS:  ..... Class = IN
DNS:  Answers
DNS:  ..... Name=www.ethz.ch
DNS:  ..... Record Type= AAAA
DNS:  ..... Class = IN (0x0001)
DNS:  ..... TTL=1 hour
DNS:  ..... Data length= 16
DNS:  ..... Address = 2001:67c:10ec:4380::216
DNS:  Non-authoritative nameservers
DNS:  ..... ""
DNS:  Additional records
DNS:  ..... ""
DNS:

```



Q49/ What does this exchange correspond to? Explain both packets.

[A49]

Q50/ Mention two ways that can be used to recognize a DNS packet.

[A50]

Q51/ What does "non-authoritative nameservers" mean?

[A51]

Q52/ How would you get the IPv6 address of `www.ethz.ch` using `nslookup`? Try it on your VM and report your findings.

[A52]