Computer Networks Lab

based on

Mastering Networks - An Internet Lab Manual
by Jörg Liebeherr and Magda Al Zarki

and on

Computer Networking - A Top-Down Approach
by James Kurose and Keith Ross

Adapted for
'Computernetwerken'
by Johan Bergs and Stephen Pauwels

Name 1
Name 2
Group groupID

February 16, 2023

Introduction

Before diving into this computer networks lab course, it is useful to present some conventions first. ***Keep in mind that these conventions are introduced to make all of our lives easier.*** In case of problems, it is easier to have a look at a lab setup if everyone sticks to the same rules.

0.1 Practical Arrangements

Group Practice

The labs are performed in groups of two students. Within the next week, you need to form your groups. To register, subscribe to one of the groups that have been made available on Blackboard.

Part 1. Mininet Virtual Machine

Getting the VM

We provide a Virtual Machine with an Ubuntu variant and IPMininet installed. You can download this image from `https://student.idlab.uantwerpen.be/computernetwerken/`. There are two distinct VMs available: one for running on Intel-based computers ("Computernetwerken-AMD64"), and one for running on Apple silicon ("Computernetwerken-ARM").

If you are using a computer running on Intel architecture, we suggest you use Virtualbox (`https://www.virtualbox.org/`) for running your Virtual Machine. After installing Virtualbox, set up your machine as follows:

1. Download and open the Computernetwerken-AMD64.ova file.
2. You should now see the VirtualBox Import Appliance.
3. Click the Import button, and the Virtual Machine is imported in Virtualbox. This can take a few minutes.

When you are using a machine running on Apple silicon, A VM is provided that has been made with the "UTM" virtualisation software. UTM is freely downloadable from the App Store. If your are using Apple silicon, set up your machine as follows after installing the UTM appliance:

1. Download and open the Computernetwerken-ARM.utm.zip file.

2. Unpack the zip file.
3. You should now see the UTM Appliance.
4. Double-click the Computernetwerken-ARM.utm file to import it into UTM.

Now you should be able to start your virtual machine. The password is `mvkbj1n`. Open a terminal and type `sudo mn` to verify if the machine is indeed installed correctly. If everything is fine, you should see Mininet starting up and creating a basic network.

Sudo

When working in the terminal either use `sudo` before each command to get root privileges.

```
% sudo ip addr add 2001:db8::1/64 dev eth0
```

Or type in `sudo su -` at the beginning of each new terminal session to keep working with root privileges.

You will be asked for a password when using the `sudo` command, the default password on the VM is `mvkbj1n`.

0.2 IPv6 Address Notation

Throughout the course, Internet Protocol (IP) version 6 (IPv6) addresses are used. The following conventions always apply:

- For ***documentation purposes***, IPv6 addresses are used that belong to the *documentation range*, i.e., `2001:db8::/32`. These addresses are ***never*** used on actual interfaces!
- The actual IPv6 addresses that are configured on the PCs and routers, will always belong to the *unique local unicast* range, which is defined as `fc00::/8` and `fd00::/8`.

0.3 Communication

Do not hesitate to seek help! The objective of this course is to get a hands on experience in networking, not in debugging devices or software. If you get stuck, try to find a solution using one of the following channels:

- Send an e-mail: johan.bergs@uantwerpen.be, or andrei.belogaev@uantwerpen.be

All information about this course will be posted on Blackboard. Blackboard will also be used to collect your reports.

0.4 Course Evaluation

This course will be evaluated based on the lab reports that you will need to complete and send in before the given deadlines.

Evaluation Criteria and Deadlines

Evaluation Criteria (for groups of two)

1. Each group member receives the same grade.
2. The final grade is calculated as an average of the grades of the individual lab reports.
3. The lab reports are subject to deadlines. If a lab report is not handed in by 23:59 on the day of the deadline, the grade for that lab report will automatically be 0/20. Consult Blackboard for this year's deadlines.

0.5 Lab Reports

Writing Lab Reports

This course expects you to write your lab report using LaTeX. To make things easier, you will be provided with a template for each lab which already contains the questions you need to fill in. The idea is that you answer the questions in separate files in the `solutions/` folder. You are supposed to send in a compiled PDF file, the `.tex` solution files and your traces.

Download the lab report templates from Blackboard. They will all be bundled in one file `Lab.tar.gz`. Unpack this file and you will see the following structure:

```
/
├── Lab0/
├── Lab1/
│   ├── graphics/ ........ Graphical resources used in lab.
│   ├── solutions/ ....... Contains all your solutions.
│   │   └── L1-1-1.tex ... Answer to first question of the lab.
│   ├── traces/ .......... Place your traces here. Mostly pcap, txt and pdf files.
│   ├── lab1.tex ......... Compile this file to create lab1.pdf.
│   └── main1.tex ........ Body of lab1. Don't edit directly.
├── Lab2/ ................ Same as Lab1/.
├── Lab3/
├── ...
├── acro.text ............ List of acronyms.
├── bibliography.bib ..... Bibliography.
├── footer.tex ........... Don't edit this file.
├── groupid.tex .......... Add your name and group id in this file.
├── header.tex ........... Don't edit this file.
└── labo.tex ............. Concatenation of all labs.
```

Each of the labs needs to be submitted separately, so make sure you do not use any files outside of the directory of the lab.

Using LaTeX on your own computer

Try to compile the `lab0.tex` file with:

```
pdflatex lab0.tex
```

This should work and create the file `lab0.pdf`.

You might get an error message in the following labs that looks like this :

```
! You can't use `macro parameter character #' in vertical mode.
#
Add your group information to groupid.tex and remove this line.
```

This tells you that you need to add your name and group id to the file `groupid.tex`. After you updated the information, remove the line starting with '#' and the lab should compile without errors.

Using Overleaf.com to write LaTeX

You can also upload all your source files to `Overleaf.com`, this is an online tool that allows you to write LaTeXand generate pdf files. Create a new project for every lab, and upload all files to this project (make sure you keep the directories as they are). You can share an overleaf project with the student you are working with.

In Menu, you can choose to download the source code and the generated the pdf, both should be uploaded to Blackboard.

Answering questions

You can then begin to answer the questions in the lab. These will look like this:

Exercise 1:

L0-1-1 What is the answer to life the universe and everything? /1

Notice that the blue box with the question number is clickable. When you click the question, your default text editor should open and display the file `solutions/L0-1-1.tex` (not in Overleaf.com).

Depending on the program you use to view PDF documents, you might need to change some security settings to get this to work. For example on Linux you can disable apparmor for evince (default PDF viewer) with the following commands:

```
sudo ln -s /etc/apparmor.d/usr.bin.evince /etc/apparmor.d/disable/usr.bin.evince
sudo /etc/init.d/apparmor restart
```

❗This disables all security limitations imposed by apparmor on evince. Use these commands at your own risk.

Sometimes you will be asked to include data, or write a python script that specifies the topology for the following exercise. You should save all files in the `traces/` directory with the name given in the exercise. If you do this correctly, you can then use the `\includetrace{}` command to include the data from a file in the traces directory.

For example: if you want to include the file `traces/L0-1-2.txt` in question 2 of exercise 1 in lab 0, all you need to type is `\includetrace{txt}` . The prefix "`traces/L0-1-2.`" is automatically added. Give it a try: edit `L0-1-2.txt` so that it contains some text, and include it.

L0-1-2 Include `L0-1-2.txt` below. /1

Of course not all traces need to be included in the lab. Often you will be asked to refer to traces or link them. Sometimes this has even been done for you. In this case you should use the `\linktrace{}` command. This allows you to link to trace files much like questions link to their answers. The linktrace command is a little bit smarter than includetrace. It will only link to trace files when they are present. If this is the case, the link will be blue and you should be able to click it. If not, the text is red and not clickable. Here is an example:

L0-1-3 Refer to `L0-1-3.pcap` and `L0-1-3.txt`. /1

As you can see, the file `L0-1-3.txt` is present, but `L0-1-3.pcap` is not.

When you encounter a LaTeXerror that you cannot work around, use the verbatim environment.

```
\begin{verbatim}

\end{verbatim}
```

Installing LaTeX

Having issues installing LaTeXon your own laptop? Check http://www.latex-project.org/get/.

Answering Questions

When answering a question in your lab report, keep the following in mind:

- Read the question carefully. If you are asked to compare two things, then do so. Do not describe one and forget about the other. Comparing also means that you do not describe both items separately, but that you describe the similarities and differences between them.
- Most answers can be short, but make sure you include all necessary information.

- Only answer the question, otherwise you may lose points even when you include the correct answer. If you are asked to give a Media Access Control (MAC) address, for example, only give the MAC address and do not give the IPv6 address as well.
- If you are asked to give examples by giving packet IDs from specific packets in a trace file, then do so. Take extra care that you are referring to the correct trace file, especially if you made several traces for the same exercise.
- Give scientific answers. For example, you should write “The ping in this setup is five times faster than in the previous setup”. Do not write “There is a huge speed difference.” or “In this scenario, the ping is way faster.”

0.6 Lab 0: Introduction

In this first lab, you will acquaint yourself with the tools we are going to use this semester.

Part 1. Basics of Wireshark

Wireshark is a network protocol analyser with a graphical user interface. Using Wireshark, you can interactively capture and examine network traffic, view summaries and get detailed information for each packet. Wireshark can be downloaded from `https://www.wireshark.org`.

Exercise 2: Running Wireshark

This exercise walks you through the steps of capturing and saving network traffic with Wireshark. The exercise is conducted on your own computer.

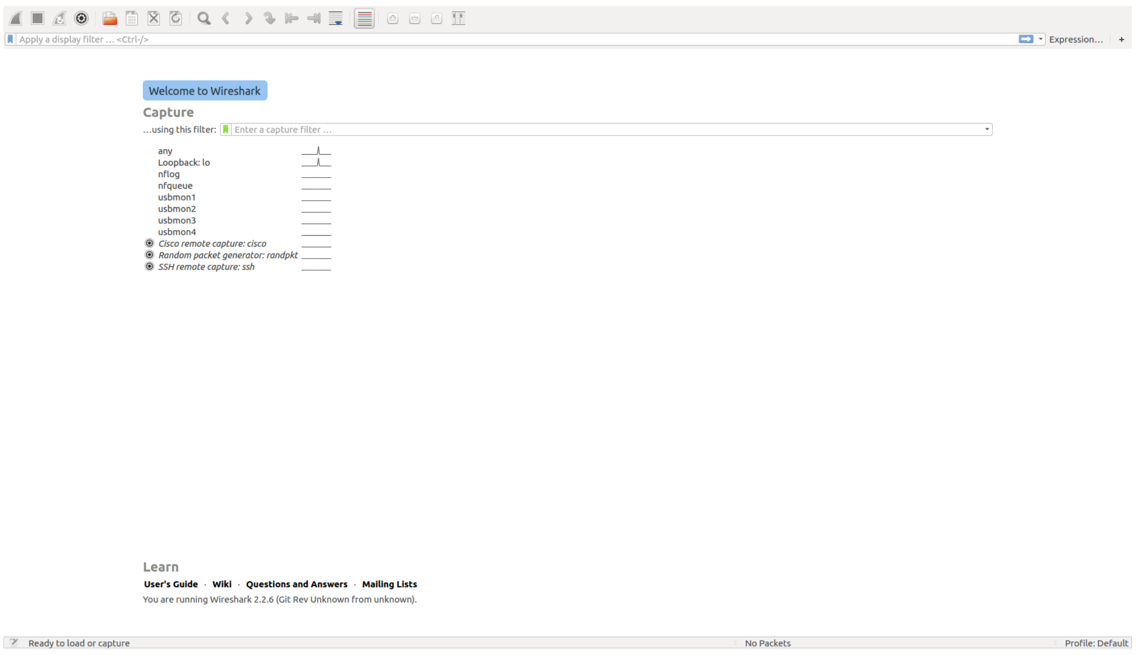

Apply a display filter ... <Ctrl-/>
Expression...
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
any
Loopback: lo
nflog
nfqueue
usbmon1
usbmon2
usbmon3
usbmon4
Cisco remote capture: cisco
Random packet generator: randpkt
SSH remote capture: ssh
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
You are running Wireshark 2.2.6 (Git Rev Unknown from unknown).
Ready to load or capture
No Packets
Profile: Default

Figure 1: Wireshark Main Window.

1. Starting Wireshark: You can start Wireshark from a terminal with the command `wireshark` or from the desktop.

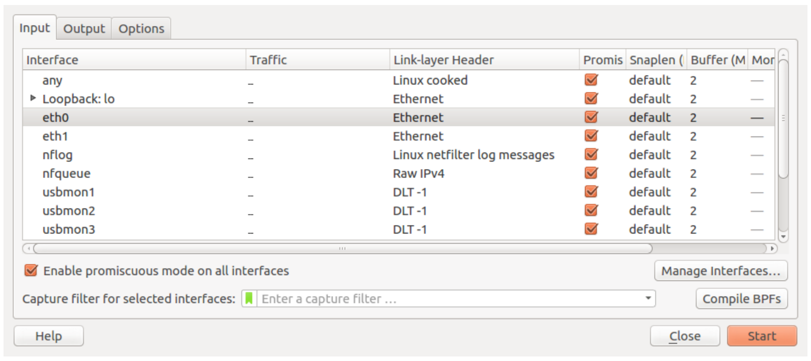
   This displays the Wireshark main window on your desktop as shown in figure 1.

2. Selecting the capture options: use the instructions in figure 2 to set the options of Wireshark in preparation for capturing traffic. Use the same options in other labs, whenever Wireshark is started.

Selecting capture preferences in Wireshark

(a) From the main window, select “Capture:Options”.

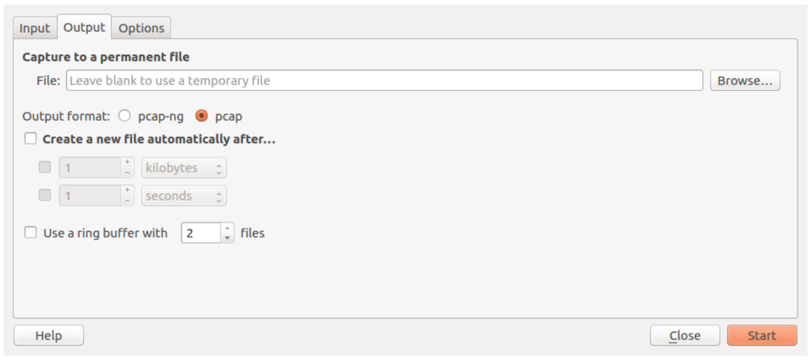
(b) This displays the following “Capture Preferences” window:


Input | Output | Options

Interface	Traffic	Link-layer Header	Promis	Snaplen (	Buffer (M	Mor
any	_	Linux cooked	✓	default	2	—
▸ Loopback: lo	_	Ethernet	✓	default	2	—
eth0	_	Ethernet	✓	default	2	—
eth1	_	Ethernet	✓	default	2	—
nflog	_	Linux netfilter log messages	✓	default	2	—
nfqueue	_	Raw IPv4	✓	default	2	—
usbmon1	_	DLT -1	✓	default	2	—
usbmon2	_	DLT -1	✓	default	2	—
usbmon3	_	DLT -1	✓	default	2	—

[✓] Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: Enter a capture filter ...


Input | Output | Options

Capture to a permanent file

File: Leave blank to use a temporary file

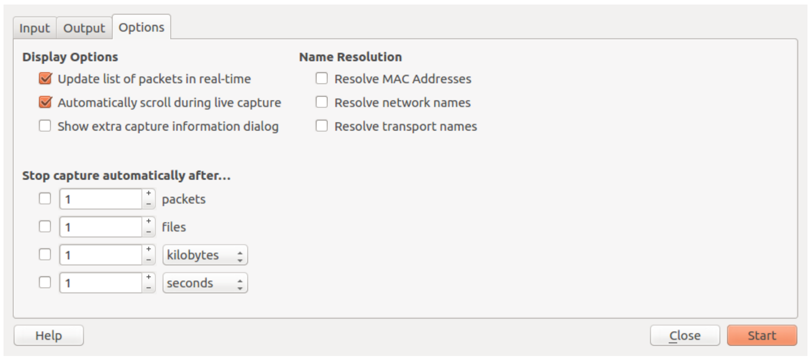
Output format: ○ pcap-ng ● pcap

[ ] **Create a new file automatically after...**

[ ] 1 kilobytes

[ ] 1 seconds

[ ] Use a ring buffer with 2 files


Input | Output | Options

Display Options

[✓] Update list of packets in real-time

[✓] Automatically scroll during live capture

[ ] Show extra capture information dialog

Name Resolution

[ ] Resolve MAC Addresses

[ ] Resolve network names

[ ] Resolve transport names

Stop capture automatically after...

[ ] 1 packets

[ ] 1 files

[ ] 1 kilobytes

[ ] 1 seconds

- Select `eth0` in “Interface”.
- Select “Use promiscuous mode on all interfaces”.
- Select “Update list of packets in real time”.
- Select “Automatically scroll during live capture”.
- Unselect “Resolve MAC addresses”.
- Unselect “Resolve network-layer names”.
- Unselect “Resolve transport-layer name”.

Figure 2: General capture settings for Wireshark

3. Starting the traffic capture: Start the packet capture by clicking “Start” in the “Capture Options” window.

4. Generating traffic: in a separate window, execute a ping to www.google.be. Limit the amount of pings to 5. This is done by executing the following command:

   ```
   % ping -c 5 www.google.be
   ```

   Observe the output in Wireshark’s main window. Click and highlight a captured packet in the Wireshark window, and view the headers of the captured traffic.

5. Stopping the traffic capture: click “Stop” in the window “Ethernet Capture”.

6. Saving captured traffic: save the results of the captured traffic as libpcap file called `L0-2-1.pcap`.

   If you select “Save” in the “File” menu, the captured data is saved in the format of a libpcap file. This format can be interpreted by both tcpdump and Wireshark. Measurements saved in libpcap format can be analysed at a later time.

   Always include binary libpcap traces with your report. Without the trace files, it is impossible to verify the validity of your answers. Keep in mind that, when answering questions that refer to a trace file, you will get no points if the trace file is not included, even if the answer is the correct one!

L0-2-1 Using the captured ping packets, describe which fields are present in the Internet Control Message Protocol (ICMP) header. What is the amount of data present in the packet? How much bytes are required (in total) to actually transmit this data? /2

Part 2. Creating virtual networks with IPMininet

To be able to perform exercises on various network topologies, we use virtual networks created with IPMininet. This tool allows us to write a python script that describes the full topology, after which we can use this topology to run our tests. IPMininet is an extension of Mininet which allows to use both IP version 4 (IPv4) and IPv6 addresses.

Exercise 3: Creating a Mininet topology

When we want to create our own network topology, we define this network in a Python script which we can load into Mininet. This is done by using the API provided by Mininet. This script has the following structure and can be found in `solutions/mininet.py`:

```
from ipmininet.iptopo import IPTopo
from ipmininet.ipnet import IPNet
from ipmininet.cli import IPCLI

class MyTopo( IPTopo ):

    def build(self, *args, **kwargs):

       # In this method you create all host, switches, links, ...

        super().build(*args, **kwargs)

# Create a network using the topology you just created and run IPMininet
net = IPNet(topo=MyTopo(), allocate_IPs=False)

try:
    net.start()
    IPCLI(net)
finally:
    net.stop()
```

solutions/mininet.py

We can now add hosts, switches and routers by using the following methods:

- `self.addHost(hostname)`
- `self.addSwitch(switchname)`
- `self.addRouter(routername)`

This will create the hosts, switches and routers. The next step is to link them together to form the correct topology, we can do this using `self.addLink(node1, node2)`. For every link you define, a network interface is created on the connecting nodes. Now we can set the IP addresses (both IPv4 and IPv6) of these interfaces using the `addParams` method.

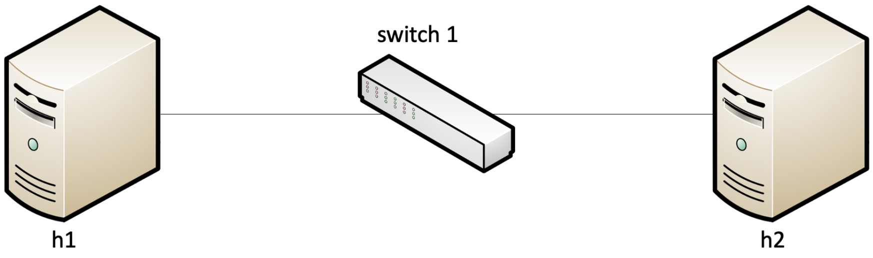


Figure 3: Our first network topology, using 2 hosts and 1 switch

Host	IPv6 address of eth0	IPv4 address of eth0
h1	`fc00:0:0:1::1/64`	`192.168.1.1/24`
h2	`fc00:0:0:1::2/64`	`192.168.1.2/24`

Table 1: IP addresses for Exercise 3

To implement the topology we now have to add a few lines to the `mininet.py` script:

```
from ipmininet.iptopo import IPTopo
from ipmininet.ipnet import IPNet
from ipmininet.cli import IPCLI

class MyTopo( IPTopo ):

    def build(self, *args, **kwargs):
        h1 = self.addHost("h1")
        h2 = self.addHost("h2")
        s1 = self.addSwitch("s1", stp=False)

        ls1h1 = self.addLink(s1, h1)
        # Set the IP addresses of host h1
        ls1h1[h1].addParams(ip=("fc00:0:0:1::1/64", "192.168.1.1/24"))

        ls1h2 = self.addLink(s1, h2)
        # Set the IP addresses of host h2
        ls1h2[h2].addParams(ip=("fc00:0:0:1::2/64", "192.168.1.2/24"))

        super().build(*args, **kwargs)

# Create a network using the topology you just created and run IPMininet
net = IPNet(topo=MyTopo(), allocate_IPs=False)

try:
    net.start()
    IPCLI(net)
finally:
    net.stop()
```

solutions/L0–3–1.py

To run this network we need to run our Python script. The script will start the Command Line Interface for mininet from which we can interact with the different hosts and components of the network.

Before you run your assignments on these networks, you first have to make sure that the configuration is correct. Mininet has some built in commands that makes it easier for you to check all connections between nodes. Try the following command:

```
pingall
```

You see that this command issues a ping command from all nodes to all nodes and displays which nodes are able to communicate which each other.

You can also run commands from one of the hosts, you do this by first entering the host name and then the command you want to perform on this host. If we want to use the ping command on host **h1** to check if we can reach host **h2**, you can use the following command:

```
h1 ping -c 3 h2
```

This command sends 3 ping requests to host **h2** and displays the response it receives. Instead of using the hostname we can also use the IP address of host **h2**:

```
h1 ping -c 3 192.168.1.2
```

You can also check the network interfaces of the different hosts by using `ifconfig`. Use this command to get an overview of the settings of the network interfaces of host **h2**. Copy the generated output to `L0-3-1.txt`.

Exercise 4: Creating a more complex Mininet topology

Look at the topology in Figure 4 and write a Python script that runs this topology in Mininet. The IP addresses to use are listed in Table 2. Save the file as `L0-4-1.py`. Use `pingall` to test if all hosts can reach each other. You can assign an IP address to the interface of a router like you do for hosts.

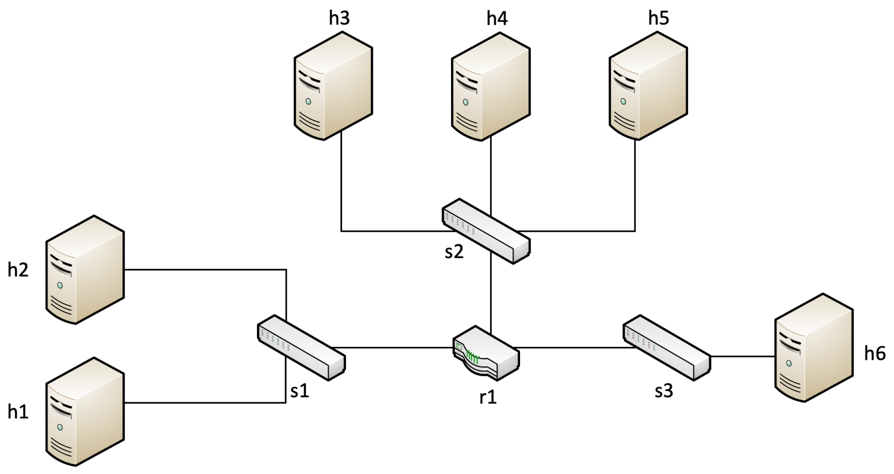


Figure 4: Our second network topology, using 6 hosts, 3 switches and 1 router

Exercise 5: Capture Mininet packets using Wireshark

Our virtual network also produces packets that we can capture using Wireshark, just as we did before. To capture the Mininet packets on a host, you have to follow the following steps:

1. Start your Python script
2. If you want to look at the packets on host h1, you use the command `xterm h1`. This opens a new terminal which is running exclusively on host h1.

Host	IPv6 address of eth0
h1	`fc00:0:0:1::1/64`
h2	`fc00:0:0:1::2/64`
h3	`fc00:0:0:2::1/64`
h4	`fc00:0:0:2::2/64`
h5	`fc00:0:0:2::3/64`
h6	`fc00:0:0:3::1/64`
r1 - eth0	`fc00:0:0:1::10/64`
r1 - eth1	`fc00:0:0:2::10/64`
r1 - eth2	`fc00:0:0:3::10/64`

Table 2: IPv6 addresses for Exercise 5. The interfaces of the router are numbered from left to right.

3. From this new terminal, start Wireshark.

When you have to select the interface to capture the packets from, you will see extra interfaces that correspond with the interfaces configured on your Mininet host (h1, in this case).

In this exercise we will run a http server on one of the hosts and get basic files from it using Hypertext Transfer Protocol (HTTP).

1. Start the Mininet CLI using the network we created in the previous part.
2. Python has a build-in http server we can use. To start a server on host **h2**, use the following command:

   ```
   h2 python3 -m http.server --bind :: &
   ```

3. Start Wireshark, and capture packets from the eth0 interface of host **h1**
4. Use `wget` to download index file from server, use the following command:

   ```
   h1 wget http://h2:8000
   ```

5. Save the HTTP packets that were captured by Wireshark in `L0-5-1.pcap`

Part 3. Basics of tcpdump

Tcpdump allows you to capture traffic on a network and display the packet headers of the captured traffic. The `tcpdump` command can be used to identify network problems or to monitor network activities.

Exercise 6: Simple tcpdump exercise

Use the `tcpdump` command to observe the network traffic that is generated by issuing `ping6` commands.

If you use the `tee` or `tail` commands to simultaneously view and save the output from `tcpdump`, you need to use the `-l` option of `tcpdump`. For example:

```
% tcpdump -n -l > filename \& tail -f filename
% tcpdump -n -l | tee filename
```

! *It may be necessary to hit `Ctrl-c` to terminate the tcpdump session. In some situations, it may be best to simply redirect the output of tcpdump straight to a file (e.g., `tcpdump >filename`) and view it afterwards with the `less` command or a text editor.*

1. Start tcpdump on **h1** so that it monitors all packets that contain the IPv6 address of **h2**, by typing:

   ```
   h1 tcpdump -n host fc00:0:0:1::2 > tcpoutput.txt &
   ```

2. Send 2 pings from **h1** to **h2**
3. Stop tcpdump:

   ```
   % h1 killall tcpdump
   ```

4. Observe the output of tcpdump, which is captured in tcpoutput.txt. Rename this outputfile to `L0-6-1.txt`.

L0-6-1 Include the saved output in your lab report. Explain the meaning of each field in the captured data. /1

Exercise 7: Another tcpdump traffic capture

1. On **h1**, start capturing packets using the `tcpdump -i h1-eth0 -n` command. Don't forget to capture the output in a file.
2. Issue a ping (limited to 2 pings) to the non-existing IPv6 address `fc00:1234::1`.
3. Issue a ping to the special address `ff02::1`.

4. Save the outputs of ping and tcpdump to `L0-7-1.ping.txt` and `L0-7-1.tcpdump.txt`, respectively.

L0-7-1 Include the saved output in your lab report and interpret the results. What happens when you ping `fc00:1234::1`? Did you capture any packets? Why or why not? How many of the Linux PCs responded to the ping to `ff02::1`? Why is this? /2

Acronyms

ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPv6	IP version 6
IPv4	IP version 4
MAC	Media Access Control
HTTP	Hypertext Transfer Protocol