Linux Lab Activity #1: Create a virtual network using Mininet

# DAT230 Communication Technology I

Department of Electrical Engineering and Computer Science, University of Stavanger

# Author: Aitor Martin Rodriguez ([aitor.martinrodriguez@uis.no](mailto:aitor.martinrodriguez@uis.no))

# Objectives

In this lab, you will create a simple virtual network in a Linux environment, using Ubuntu 22.04 LTS. The goals of this lab are the following:

* Become familiar with the Linux Command Line Interface (CLI).
* Learn how to install packages on Ubuntu OS.
* Learn how to create a virtual network using *mininet.*

# Background / Scenario

In this activity you will learn the basics of GNU/Linux-based OSs and become familiar with the use of the Linux CLI for moving through the file system and operating with files.

You will then use **mininet** to create your own virtual network. **mininet** (<http://mininet.org>) is an application that allows you to create virtual networks in a single Linux machine, creating virtual nodes that act as hosts and switches:

* **Virtual hosts** run real instances of the Linux kernel and run real application code in real-time.
* **Switches** are based on software switch implementations such as Open vSwitch (<https://www.openvswitch.org>).

The **mininet controller** allows to monitor and control every node in the network through a console. It also provides access to the CLIs of each virtual node. In Figure 1, we show a comparison between a real network and a virtual network deployed with mininet.

💡 💡 **TRIVIA** 💡 💡

The mininet controller can also act as a Software Defined Network (SDN) controller, which allows to experiment with protocols such as OpenFlow and programming languages such as P4. You will learn more about SDN in the last weeks of this course.

Even if virtual networks introduce some abstraction from the real world, they are highly scalable and cost-effective**:** they allow to change topology and add more nodes without having to redo any cabling or configuration. This makes mininet a very powerful tools for network application developers, students, and researchers.

**iperf3** (<https://iperf.fr>) is a CLI tool for measuring throughput (data rate) between two end-hosts in an IP network. To run iperf3 tests, an iperf3 server must be started on one end-host and an iperf3 client must establish a connection with the server from the other end-host, as shown in Figure 2. It can be used to measure network metrics such as troughput, packet loss and jitter, and it is compatible with TCP, UDP and other transport layer protocols.

**A diagram of a computer network

Description automatically generated**

Figure 1. Comparison between a real and a virtual network

A diagram of a cloud

Description automatically generated

Figure 2. Example of an iperf3 TCP download measurement

iperf3 has several options which you can experiment with. The fundamental options that you will use in this lab are the following:

| client | | server | |
| --- | --- | --- | --- |
| **-c** <server\_ip> | Enables client mode and attempts to connect to IP | **-s** | Enables server mode |
| **-u** | Enables UDP mode |  | |
| **-b** <rate> | Sets download rate (only in UDP mode) |
| **-R** | Enables reverse mode (data flows from server to client) |
| **-t** <time> | Specify test duration in seconds |
| Common | | | |
| -p <port> | Specify server port (default 5201) | | |

# Requirements

* A computer running a Linux-based OS (VM or native).
* The following Linux packages:
  1. Essential packages: build-essential, net-tools,
  2. Tools for network emulation: mininet, xterm
  3. Traffic generation and performance measurement tools: iperf3

# Instructions

## Become familiar with the Linux CLI.

In this part, you are going to do a few basic exercises to get familiar with the commands shown in the tutorial slides. You will be handed a cheat sheet with all the necessary commands.

### Explore the filesystem.

Hint: use the commands from Slide 37: pwd, cd, ls

* + - 1. Open a terminal (Right Click > Open in Terminal, shortcut: CTRL+ALT+T)
      2. Verify your current directory.

**Q: What is the default starting directory when you open a new terminal?**

* + - 1. Move to the directory /etc and verify its content.

**Q: What does this directory contain? Explore the directory and comment on what protocols or services you recognize.**

### Operate with files.

Hint: use the commands from Slides 37 and 38: cd, mkdir, touch, cat, cp, mv, rm…

Complete the following tasks AND **write down the command(s) you use for each step.**

* + - 1. Move to your home directory and create a new directory with the name **‘kommtek’**.

Command(s): pwd , cd home/boglaptop, mkdir kommtek

* + - 1. Inside the newly created directory, create a new text file called **‘test.txt’**.

Command(s): touch test.txt

* + - 1. Open the file with a CLI text editor (e.g., **nano** or **vi**), write something inside, and save the changes. Werecommend **nano** for beginners, but we encourage you to test **vi** and use whichever is the most comfortable for you.

Instructions for saving the file and exiting the nano editor:

* + - * 1. CTRL+O and then ENTER to save the changes.
        2. CTRL+X to leave the editor.
      1. Verify that the file was saved by printing its contents on the CLI.

Command(s): cat test.txt

* + - 1. Duplicate the text file to a new file called **‘test2.txt’.** Verify that the copy has been created appropriately.

Command(s):

cp test.txt test2.txt

ls

cat test2.txt

* + - 1. Rename the new duplicate file to **‘test3.txt’**.

Command(s):

mv test2.txt test3.txt

* + - 1. Create a new folder inside of **/home/$USERNAME/kommtek** named “**files”** and move the text file **‘test3.txt’** inside.

Command(s):

mkdir /home/boglaptop/komtek/files

mv test3.txt files/

* + - 1. Remove the original file **‘test.txt’**.

Command(s):

rm test.txt

## Deploy an emulated test network with *mininet*

### Verify mininet and xterm installation.

* + - 1. Check mininet version. You should have version 2.3.0.

$ **mn –version**

* + - 1. Verify that xterm is installed.

$ **xterm**

A new CLI window should appear. You can now close it.

* + - 1. Run an iperf3 test towards a public server. You can use the server “**paris.testdebit.info**”, which runs iperf3 servers listening on ports 9200-9240.

$ **iperf3 -c paris.testdebit.info -p 9200 -R**

NOTE: If port 9200 is unavailable, try other ports between 9200 and 9240.

**Q: What information do iperf3 tests provide?**

**Interval, transfer , bitrate**

**Q: Have you performed a similar test using your browser? Which kind of test? Yes , it’s a speedtest.**

### Learn the basics of mininet.

* + - 1. Start a simple mininet session with the default topology.

$ **sudo mn**

* + - 1. Use the following commands from the mininet console to get information about the topology: **nodes, links, net, ports**.

**Q: How many hosts and switches are there in the default topology? How are they connected, and what are the names of their interfaces?**

**2 hosts, 1 switch**

**H1-eth0 s1 eth1**

**H2-eth0 s1 eth2**

* + - 1. Ping h2 from h1 and verify that the ping is successful. Use “-c 5” to only send 5 ICMP Echo requests.

mininet> **h1 ping h2 -c 5**

* + - 1. Use xterm to access the CLIs in the hosts. Two new xterm windows should appear with h1 and h2’s CLIs.

mininet> **xterm h1 h2**

NOTE: You can make the text in the xterm windows bigger with **CTRL+Right click** and change the **VT Fonts size**.

* + - 1. Figure out the interface names and IP addresses assigned to h1 and h2 by mininet, and fill in the table.

NOTE: You can use the Linux **ifconfig** command to view the network interface configuration.

| Host | Interface name | IP address |
| --- | --- | --- |
| h1 | h1-eth0 | 10.0.0.1 |
| h2 | h2-eth0 | 10.0.0.2 |

* + - 1. From h2’s CLI, start a traffic capture using **tcpdump** on the interface you found in the previous step.

h2: $ **tcpdump -i <interface\_name>**

* + - 1. From h1’s CLI (the xterm window opened in Step 2.d), start a ping towards h2. Observe how the ICMP packets are captured on h2 by tcpdump.

**Q: What’s the average round trip time or latency between h1 and h2? 0.1**

**Q: What information does the *tcpdump* packet capture show? ICMP requests and reply**

**Q: Does *tcpdump* show more traffic other than ICMP? Which protocol(s) do you recognize?**

**YES ARP**

* + - 1. Stop the session and clean it.

mininet> **exit**

$ **sudo mn --clean**

## Create a network topology and run basic performance tests

In this part, you will set up the mininet session that you will use during the whole assignment and learn how to run basic performance tests using iperf3.

### Step 1: Create a dumbbell topology and verify connectivity.

1. Start a new mininet session with a dumbbell topology with **2 hosts on each side**, as shown in the figure below. Use the following command:

$ sudo mn --topo=linear,n=2,k=2

NOTE: There should not be any whitespaces between the commas.



Figure 2. A basic dumbbell topology

* + - 1. Figure out the IP addresses assigned to the end hosts and the switch ports they have been connected to. You can use the **ifconfig** commands in each host to view the interface configuration. You can also use the commands **links**, **ports** and **net** on the mininet console to get more information.

| Device | IP address | Switch port number |
| --- | --- | --- |
| h1s1 | 10.0.0.1 | Eth1 |
| h2s1 | 10.0.0.3 | Eth2 |
| h1s2 | 10.0.0.2 | Eth1 |
| h2s2 | 10.0.0.4 | Eth2 |

* + - 1. Verify that the IP addresses you collected match the output of the **dump** command in the mininet console.

mininet> dump

* + - 1. Verify that all nodes can reach each other.

mininet> pingall

### Step 2: Measure raw TCP/UDP performance under ideal link conditions with iperf3.

1. Open the CLIs of all 4 hosts from the mininet console using **xterm**.

mininet> xterm h1s1 h1s2 h2s1 h2s2

1. Start an iperf3 server in h1s2, selecting a port based on your group number, so that:

**<your\_port> =** 5000 + <group\_number>

*e.g., if you are in Group 7, you must use port 5007.*

h1s2: $ iperf3 -s -p <your\_port>

1. Start a 30 second TCP download in h1s1 from h1s2, using iperf3. This allows us to estimate the maximum bandwidth in the link.

h1s1: $ iperf3 -c 10.0.0.2 -p <your\_port> -t 30 -R

REMEMBER**:** the option -R enables reverse data transmission – i.e., from server to client.

**Q: What is the average bitrate measured? Why is it so high?**

**19.6gbits/sec. because they have the same NIC since its virtual.**

1. Start a 30 second UDP download of 20 Mbps in h1s1 from h1s2, using iperf3.

h1s1: $ iperf3 -c 10.0.0.2 -p <your\_port> -t 30 -R -u -b 20M

**Q: How much packet loss have you measured? Why is it so low? 0%**

**Virtual envoirment makes it impossible to lose packets.**