# Lab Exercise 3 (part B) Network Performance

## Objectives:

- Learn how to measure network performance in Mininet
- Gain insights into examining delay, loss, and throughput for a given network topology

## Marks:

This exercise forms the second part of lab exercise 3. You should have already completed the first part in the previous lab (Part A). Both parts together will comprise 15 marks. Only selected exercises will be marked. However, students must submit answers for all exercises.

## Deadline:

Before your scheduled lab next week. So you get one week to work on this lab. For example, if you go to the Monday 18:00 lab, then your submission is due at 17:59 on the following Monday. You can submit as many times as you wish before the deadline. A later submission will override the earlier submission, so make sure you submit the correct file. Do not leave until the last moment to submit, as there may be technical or communications error and you will not have time to rectify it.

## Late Penalty:

Late penalty will be applied as follows:

- 1 day after deadline: 20% reduction
- 2 days after deadline: 40% reduction
- 3 days after deadline: 60% reduction
- 4 or more days late: NOT accepted

Note that the above penalty is applied to your final mark. For example, if you submit your lab work 2 days late and your score on the lab is 8, then your final mark will be 8-3.2 (40% penalty) = 4.8.

## Submission Instructions:

Submit a PDF document **lab3b.pdf** with answers to all questions for all exercises. Include all supporting documents such as topology files and text files conducting measurements (Exercise 3 and 4). Create a tar archive of all files called **lab3b.tar**. Submit the archive using give. Click on the submission link at the top of the page. Max file size for submission is 3MB.

## Original Work Only:

You are strongly encouraged to discuss the questions with other students in your lab. However, each student must submit his or her own work. You may refer to the reference material and also conduct your own research to answer the questions.

Notation: In the examples below, we have used the `$` sign to represent Linux commands that should be typed at the shell prompt, `mininet>` to show Mininet commands that should be typed at Mininet's CLI (command line interface), and `#` to show Linux commands that are typed at a root shell prompt. The actual prompt may look quite different on your computer (e.g. it may contain the computer's hostname, or your username, or the current directory name). The commands that **you** are supposed to type are in **this bold font**.

**NOTE: PLEASE REMEMBER TO MOUNT YOUR CSE HOME DIRECTORY IN THE VM AND TO SAVE YOUR WORK IN YOUR HOME DIRECTORY BEFORE QUITING.**

**Several students have reported that the mn tool does not have access to their mounted home directory folder. Note that, since mn is run with sudo, the sshfs command that you use to mount your home directory should also be run with sudo. To avoid typing sudo with each command, we recommend that you run sudo –s immediately after logging in to the VM. You will then remain in superuser mode for the rest of the session and will no longer need to type sudo before mn or other commands.**

## Overview

In this lab, you will learn how to build custom topologies using Mininet Python API where certain parameters like bandwidth, delay, loss and queue size can be set individually for different links in the topology. You will also learn how to undertake performance testing of these custom topologies using *ping* and *iperf* tools. It is highly suggested to review the Week 1 Lecture notes and Section 1.4 of the textbook before starting this lab.

## Introduction to Iperf

The *iperf* tool is a commonly used network-testing tool for measuring the bandwidth and quality of a network link. The tool can create TCP and UDP data streams and measure the throughput of a network that is carrying these streams. The iperf tool implements both client and server functionality, and can measure the throughput between the two end hosts, either uni-directionally or bi-directionally.

The following figure illustrates an example where Iperf is installed on a Linux and Microsoft Windows machine[1]. The Linux machine is used as a client and Windows machine as a server.

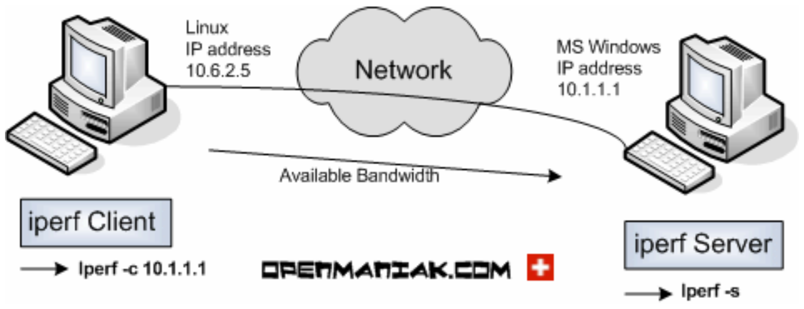

Figure 1 A sample client/server in Iperf.

For more information about the iperf tool and bandwidth monitoring in general, refer to the following:

- The article titled "Testing Network Performance with Iperf" which is available on the lab resources page.
- The link: http://openmaniak.com/iperf.php.
- An overview of bandwidth monitoring tools for Ubuntu users in http://www.ubuntugeek.com/bandwidth-monitoring-tools-for-ubuntu-users.html

[1] The source of the sample topology is in http://openmaniak.com/iperf.php.

## Exercise 1: Investigate Performance of a Linear Topology

In this exercise you will work with a linear topology, as shown in Figure 2. A linear topology consists of an equal number ($n$ in Figure 2) of hosts and switches such that each host is connected to one switch and the switches are connected as a linear topology. Figure 3 shows (incomplete) sample code for implementing the linear topology using the high-level Python API provided by Mininet

**Question 1:** Complete the source code shown in Figure 3. Code can be downloaded from the main lab exercise page. In particular, you will need to complete function *init* in the code. Include your source code as lineartoplogy_experiment.py with your submission. Remember to save it in your (mounted) CSE home directory.

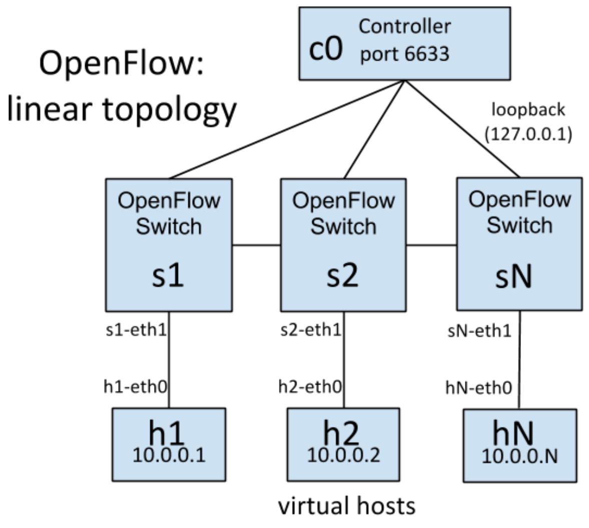

Figure 2 Hosts and switches connected in a linear topology.

```python
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import CPULimitedHost
from mininet.link import TCLink
from mininet.util import irange,dumpNodeConnections
from mininet.log import setLogLevel

class LinearTopo(Topo):
   "Linear topology of n switches, with one host per switch."

   def __init__(self, n=2, **opts):
       """Init.
           n: number of switches (and hosts)
           hconf: host configuration options
           lconf: link configuration options"""
       super(LinearTopo, self).__init__(**opts)
       self.n = n

       """ Complete the rest of this method. """

def simpleTest():
   "Create and test a simple network"
   topo = LinearTopo(n=4)
   net = Mininet(topo)
   net.start()
   print "Dumping host connections"
   dumpNodeConnections(net.hosts)
   print "Testing network connectivity"
   net.pingAll()
   net.stop()

if __name__ == '__main__':
   setLogLevel('info')
   simpleTest()
```

Figure 3 Template code for an experiment on a linear topology.

The *SimpleTest* method in the code above initiates a ping test (pingAll()) amongst all hosts, i.e. each host pings every other host in the network. Once you have completed the code in the above file, run it by typing the following (assuming that the current working directory contains the .py file):

```
mininet> sudo python lineartopology_experiment.py
```

What is the output of the *simpleTest* experiment? Report it in your lab report.

Figure 4 shows a simple experiment method called *perfTest*. The experiment executes the iperf tool between two hosts h1 and h4. Details about this function can be found here. Add this experiment to the linear topology code and execute it as above.

**Question 1**: Report and explain the results of the above experiment.

```python
def perfTest():
    "Create network and run simple performance test"
    topo = LinearTopo(n=4)
    net = Mininet(topo=topo, host=CPULimitedHost, link=TCLink)
    net.start()
    print "Dumping host connections"
    dumpNodeConnections(net.hosts)
    print "Testing network connectivity"
    net.pingAll()
    print "Testing bandwidth between h1 and h4"
    h1, h4 = net.get('h1', 'h4')
    net.iperf((h1, h4))
    net.stop()
```

Figure 4 Simple method for performance experiment.

**Remember to clean up (sudo mn –c) and also to save your work in your CSE home directory before moving forward.**

We will now work with the linear topology and use the Mininet CLI to run iperf (a different approach compared to writing experiment functions). Download the lineartopology.py file from the lab exercise page. Copy and paste the exact same lines of code that you wrote in the lineartoplogy_experiment.py (for creating the topology) in to this file where indicated. Now run the topology and enter the Mininet CLI by typing the following:

```
mininet> sudo mn --custom lineartopology.py --topo lineartopo
```

Now open separate terminals for hosts h1 and h4. This will allow you to execute iperf commands on those two hosts. Type the following:

```
mininet> xterm h1 h4
```

To run iperf we need to execute a server on one host (say h4) and the client on the other (say h1, you can reverse this if you wish). First obtain the IP address of h4 which will execute the server. (How? - see Lab3a). Then run the iperf server by typing the following in the xterm window for h4:

```
root@mininet-vm> iperf -s
```

Next execute the client by typing the following in the xterm window for h1:

```
root@mininet-vm> iperf -c IP_ADDRESS_OF_IPERF_SERVER
```

Substitute the IP address of h4 in the above command. You can find out various settings and options for iperf by typing *iperf –h.*

**Question 2:** For the above experiment, report the performance results.

**Question 3:** By default only the bandwidth from the client to the server is measured. Find out how to run a bidirectional test between the client and server in a sequential manner, i.e. first test the performance of the forward path (client -> server) and then test the performance of the reverse path (server -> client). Report the performance results.

**Question 4**: Next find out how to conduct the bidirectional test simultaneously (i.e. forward and reverse paths at the same time). Report the performance results.

**Question 5**: By default all iperf experiments are conducted over TCP. Find out how to conduct a performance test using UDP. Report the performance results.

**Remember to clean up (sudo mn –c) and also to save your work in your CSE home directory before moving forward.**

## Exercise 2: Learning to use network configuration

In addition to basic behavioural networking, Mininet provides performance limiting and isolation features, through the `CPULimitedHost` and `TCLink` classes. There are multiple ways that these classes may be used, but one simple way is to specify them as the default host and link classes/constructors to `Mininet()`, and then to specify the appropriate parameters in the topology.

There are two important performance settings supported by the Mininet classes. The first one is for configuring the CPU resources in a host machine. To this end, method `self.addHost(name, cpu=f)` is used for defining a host. The second argument in this method allows you to specify a fraction of overall system CPU resources which will be allocated to the virtual host.

The second method is `self.addLink( node1, node2, opts)`, where opts refers to link options. For example, the statement `self.addLink( node1, node2, bw=10, delay='5ms', max_queue_size=1000, loss=1, use_htb=True)` adds a bidirectional link with bandwidth of 10 Mb/s, (propagation) delay of 5ms, loss rate of 1% (i.e. 1 packet out of 100 packets is lost) and a maximum queue size of 1000 using the Hierarchical Token Bucket rate limiter and netem delay/loss emulator.

The parameter bw is expressed as a number in Mb/s; delay is expressed as a string with units in place (e.g. '5ms', '100us', '1s'); loss is expressed as a percentage (between 0 and 100); and maximum_queue_size is expressed in packets.

You may find it useful to create a Python dictionary to make it easy to pass the same parameters into multiple method calls. There are two ways to specify the dictionary as indicated below:

```
linkopts1 = dict(bw=10, delay='5ms', loss=1, max_queue_size=1000, use_htb=True)
linkopts2 = {'bw':10, 'delay':'5ms', 'loss':1, 'max_queue_size':1000, 'use_htb':True}

self.addLink(node1, node2, **linkopts1)
self.addLink(node2, node3, **linkopts2)
```

Figure 5 Configuring link performance parameters.

Repeat the previous experiments (Questions 3 - 6) on the linear topology (from Exercise 1) using the link performance settings listed in Figure 5. You should use the provided settings (linkopts1 or linkopts2) for all inter-switch links in the linear topology (i.e. for the links between the switches, s1-s2, s2-s3 and s3-s4).

**Question 7**: Download custom_lineartopology_experiment.py script from the lab exercise page. You will need to copy and paste the same code that you wrote in Exercise 1 into this file where indicated. You will also need to add the above dictionary (at the top before the class decleration in your code) and use it as an argument when creating the links between the switches (shown in the code in Figure 5). Once you have updated the code, execute the following which will setup the topology and run the mininet CLI:

```
mininet> sudo python custom_lineartopology_experiment.py
```

Now run the experiments described in Questions 3 -6 (from Exercise 1) for this custom topology. Compare the results for TCP and UDP throughput with those for the original linear topology (in Exercise 1).

**Remember to clean up (sudo mn –c) and also to save your work in your CSE home directory before moving forward.**

## Exercise 3: Performance Evaluation for a Larger Topology

In this exercise you will use the *ping* and *iperf* tools to measure the latency and throughput for a more complex topology. You must include the output from some of your experiments and the answer the questions in your lab report submission.

A python script to run Mininet with the topology shown in Figure 6 is located on the main lab exercise page named lab3b_exercise3_topo.py.

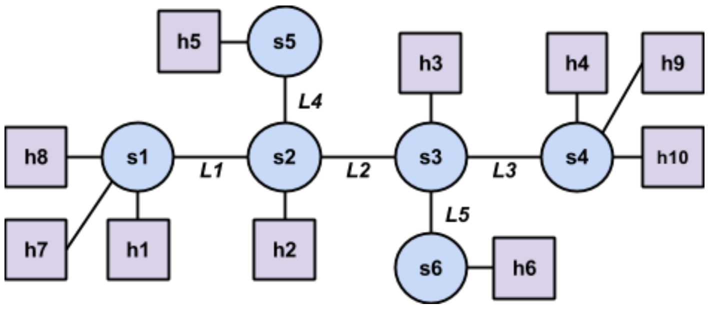

Figure 6 Network topology for Exercise 3.

As can be observed from Figure 6, hosts (h1 - h6) are represented by squares and switches (s1 - s6) are represented by circles; the names in the diagram match the names of hosts and switches in provided scripts. In each of the following questions you need to run a simple experiment for measuring various performance metrics for the above network. You can run the provided topology by executing the following command:

```
mininet> sudo python lab3b_exercise3_topo.py
```

This will start the Mininet CLI. You can run the experiments using the CLI.

## Link Latency and Throughput

**Question 8:** You should measure the RTT and TCP throughput for each of the five individual links between the switches (i.e. L1 - L5). For the former use ping and for the latter use iperf. For each link run *ping* with 20 packets ( use the '-c 20' option) and store the output of the measurement on each link in a file called *E3Q8_RTT_L#.txt*, replacing # with the link number from the topology diagram above (e.g. E3Q8_RTT_L1 for L1). To measure the performance of link L1 you will need to run the ping test between a host connected to s1 and another host connected to s2. For example, you could run the ping test between hosts h1 and h2 for L1. Similarly for L3 you can run the ping test between h3 and h10.

Next you should measure the TCP throughput by running iperf for 20 seconds (use the '-t 20' option, the default time is 10 seconds). Recall from Exercise 1, that you will need to run the iperf server on one host and the iperf client on the other host. You should store the output of the measurement on each link in a file called *E3Q8_throughput_L#.txt*, replacing # with the link number from the topology diagram above. Include all text files in your submission.

Are the RTT and throughput measurements consistent with the network topology (check the link properties in the topology).

## Path Latency and Throughput

In this part, we assume h1 wants to communicate with h4. We evaluate the latency and throughput of the path between these two hosts.

**Question 9:** Measure the RTT and TCP throughput between h1 and h4 using the ping and iperf tools. Use the same parameters as Quesiton 8 (20 packets for ping / 20 seconds for iperf) and store the output in files called *E3Q9_RTT.txt* and *E3Q9_throughput.txt*. Note the average RTT and average throughput in your submission and explain the results. Include all text files in your submission.

## Effects of Multiplexing

Now we assume that multiple hosts connected to s1 want to simultaneously communicate with hosts connected to s4. We evaluate the expected latency and throughput when two pairs of hosts are communicating simultaneously.

**Question 10:** Use the ping and iperf tools to measure the latency and throughput when there are two pairs of hosts communicating simultaneously; it does not matter which pairs of hosts are communicating as long as one is connected to s1 and one is connected to s4 (e.g., you could choose h1-h4 and h7-h9 as the two pairs). Use the same parameters as Question 8. You do not need to submit the raw output, but you should note the average RTT and average TCP throughput for each pair in your lab report and explain the results. Make sure that you conduct the measurements simultaneously. When measuring the TCP throughput make sure that you conduct the measurement for both pairs in the same direction (i.e. from s1->s4 or s4->s1). In other words, you should run the iperf server on the hosts connected to same switch (e.g. h4 and h9, assuming the pairs used are h1-h4 and h7-h9).

Repeat this experiment for three pairs of hosts (i.e. 3 hosts connected to s1 communicate with 3 hosts connected to s4) communicating simultaneously and report the average RTT and TCP throughput. Use same parameters as Question 8. Explain and compare the results with the case of two pairs of hosts.

## Effects of Latency

Now we assume h1 wants to communicate with h4 at the same time when h5 wants to communicate with h6.

**Question 11**: Use the ping and iperf tools to conduct measurements, and store the raw output in files called *E3Q11_RTT_h1_h4.txt*, *E3Q11_RTT_h5_h6.txt*, *E3Q11_throughput_h1_h4.txt*, and *E3Q11_throughput_h5_h6.txt*. Use same parameters as in Question 8. Note the average RTT and TCP throughput in your lab report and explain the results. As in Question 10, when using iperf make sure that you are conducting the measurements in the same direction (i.e. h1->h4 and h5->h6 or both in the reverse direction). Include all text files in your submission.

**Remember to clean up (sudo mn –c) and also to save your work in your CSE home directory before moving forward.**

## Exercise 4: Measurement in Mininet

In this exercise we repeat the experiments described in Exercise 3 on the network topology shown in Figure 7. As you can see, the network consists of three links L1 (s1-s2), L2 (s2-s3) and L3 (s3-s4). The bandwidth and delay for these links are shown in Table 1.

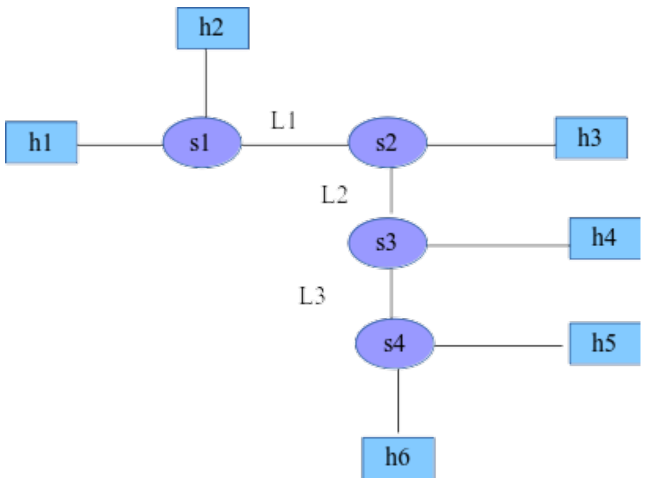

Figure 7 Network topology for Exercise 4.

Table 1 The bandwidth and delay for links in topology shown in Figure 7.

| Link | Bandwidth | Delay |
|---|---|---|
| L1 | 20 | 10 |
| L2 | 10 | 15 |
| L3 | 20 | 20 |

**Question 12:** Write a Python script for generating the network topology shown in Figure 7 and performance parameters reported in Table 1. Store and submit your script in a file named *exercise4_topo.py*.

**Question 13:** Repeat the experiment described in Question 8 for all links (L1, L2, L3) in the network topology shown in Figure 7. Use the same parameters as in Question 8. Store the results of this experiment in *E4Q13_RTT_L#.txt* and *E4Q13_throughput_L#.txt*. Explain the results. Submit the text files with your submission.

**Question 14:** Measure the RTT and TCP throughput between h1 and h5 using the ping and iperf tools.. Use the same parameters as Question 8 and store the output in files called *E4Q14_RTT.txt* and *E4Q14_throughput.txt*. Report average RTT and throughput in your submission and explain the results. Include all text files in your submission.

**Question 15:** Measure the RTT and TCP throughput when there are two pairs of hosts communicating simultaneously (i.e. two hosts connected to s1 communicate with two hosts connected to s4); it does not matter which pairs of hosts are communicating as long as one is connected to s1 and one is connected to s4. Use the same parameters as Question 8. Make sure that you conduct the measurements simultaneously. When measuring the TCP throughput make sure that you conduct the measurement for both pairs in the same direction (i.e. from s1->s4 or s4->s1). In other words, you should run the iperf server on the hosts connected to same switch (e.g. h5 and h6, assuming the pairs used are h1-h5 and h2-h6). You do not need to submit the raw output, but you should note the average RTT and throughput for each pair in your report and explain the results.

**Question 16:** Now we assume h1 wants to communicate with h6 at the same time when h3 wants to communicate with h4. Measure the RTT and TCP througput with the same parameters as in Question 8 and store the output in files called *E3Q16_RTT_h1_h6.txt*, *E3Q16_RTT_h3_h4.txt*, *E3Q16_throughput_h1_h6.txt*, and *E3Q16_throughput_h3_h4.txt*. Note the average RTT and throughput in your submission and explain the results. As in Question 15, when using iperf make sure that you are conducting the measurements in the same direction (i.e. h1->h6 and h3->h4 or both in the reverse direction). Include all text files in your submission.

## Useful Resources

- Introduction to Mininet: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet
- Mininet Walkthrough: http://mininet.org/walkthrough/
- Mininet Python API Reference Manual: http://mininet.org/api/annotated.html