Portfolio 1

This portfolio will count towards your final grade and must be solved **individually**. Your submission will be marked with respect to how well you have fulfilled the requirements listed below.

Prerequisites

We have already covered the required concepts in our lectures, labs, and mandatory assignments which you need for this portfolio. Should you be missing lectures and lab sessions, it is your own responsibility to catch up to the competence level that you are supposed to get it from the lectures and lab sessions. Also, try to make good use of the lectures and lab sessions because we will cover some important concepts there.

Overview

iPerf is a tool for measuring network throughput. In this project, you'll design and implement **simpleperf** - your own simplified version of iperf using sockets. The simpleperf tool you write will run on a virtual network managed by mininet inside a virtual machine. You'll use it to measure the performance of a network.

Implement simpleperf

You'll implement a simple network throughput measurement tool - **simpleperf**. Your tool is supposed to send and receive packets between a client and a server using sockets. Your simpleperf tool MUST run in two modes: 1) Server mode, and 2) Client mode

Server mode

When you run in server mode, simpleperf will receive TCP packets and track how much data was received during from the connected clients; it will calculate and display the bandwidth based on how much data was received and how much time elapsed during the connection. A server should read data in chunks of 1000 bytes. For the sake of simplcity, assume 1 KB = 1000 Bytes, and 1 MB = 1000 KB.

To run `simpleperf` in server mode with the default options, it should be invoked as follows:

```
python3 simpleperf -s
```

- `-s` indicates simpleperf is running in a server mode: it should receive data and track the total number of bytes

The server should print:

```
-------------------------------------------
A simpleperf server is listening on port XXXX
-------------------------------------------
```

Table below lists all the available options that you can use to invoke the server:

flag	long flag	input	type	Description
`-s`	`--server`	X	(boolean)	enable the server mode
`-b`	`--bind`	**ip address**	string	allows to select the `ip address` of the server's interface where the client should connect - use a default value if it's not provided. It must be in the dotted decimal notation format, e.g. 10.0.0.2
`-p`	`--port`	**port numr**	integer	allows to use select `port number` on which the server should listen; the port must be an integer and in the range `[1024, 65535]`, default: 8088

flag	long flag	input	type	Description
`-f`	`--format`	MB	string	allows you to choose the format of the summary of results - it should be either in B, KB or MB, default=MB)

Here is an example how one should be able to invoke the server :

```
python3 simpleperf -s -b <ip_address> -p <portnumber> -f MB
```

And the server should print the following and wait for a connection:

```
-------------------------------------------
A simpleperf server is listening on port XXXX
-------------------------------------------
```

When a client connects, a server should print:

```
-------------------------------------------
A simpleperf server is listening on port XXXX
-------------------------------------------

A simpleperf client with <IP address: port> is connected with <server IP:port>
```

At the end of the transfer, `simpleperf` client will send a "BYE" message to the server to indicate that the transfer is complete. The server will then send an acknowledgement ("ACK: BYE") to the client, print the results in the following format, and gracefully close the connection.

```
-------------------------------------------
A simpleperf server is listening on port XXXX
-------------------------------------------

A simpleperf client with <IP address: port> is connected with <server IP:port>


ID              Interval        Received     Rate

IP:port         0.0 - 25.0        X MB       Y Mbps
```

There are four columns here:

1. ID specifies the client_IP:port pair
2. Interval: Total duration in seconds
3. Transfer: **X** stands for the total number of bytes received (in Megabytes, if not specified with `-f`. Note **X** should be an integer.
4. Rate: **Y** stands for the rate at which traffic could be read in megabits per second (Mbps). and **Y** should be a float value with two digits after the decimal point.

```
ID                          Interval          Received       Rate

10.0.0.1:3354               0.0 - 25.0         2544 KB       0.82 Mbps
```

Client mode

When we invoke `simpleperf` in a client mode, it must establish a TCP connection with the simpleperf server and send data in chunks of 1000 bytes (all zeroes or same values) for `t` seconds specified with -t or –time flag. Calculate the total of the number of bytes sent. After the client finishes sending its data, it should send a finish/bye message and wait for an acknowledgement before exiting the program. Simpleperf will calculate and display the bandwidth based on how much data was sent in the elapsed time.

To operate `simpleperf` in client mode, it should be invoked as follows:

```
python3 simpleperf -c -I <server_ip> -p <server_port> -t <time>
```

- `-c` indicates this is the `simpleperf` client mode.
- `-I` specifies the `server_ip` - IP address of the `simpleperf` server will receive data from the simpleperf client
- `-p` specifies the `server_port` in which the server is listening to receive data; the port should be in the range [1024, 65535]
- `time` is the total duration in seconds for which data should be generated and sent to the server.

```
------------------------------------------------------------------------------------------
A simpleperf client connecting to server <IP>, port XXXX
------------------------------------------------------------------------------------------

Client connected with server_IP port XXXX

ID              Interval        Transfer    Bandwidth

IP:port       0.0 - 25.0         X MB       Y Mbps
```

Table below lists all the available options that you can use to invoke the server:

flag	long flag	input	type	Description
`-c`	`--client`	X	(boolean)	enable the client mode
`-I`	`--serverip`	**ip address**	string	selects the `ip address` of the server - use a default value if it's not provided. It must be in the dotted decimal notation format, e.g. 10.0.0.2
`-p`	`--port`	**port num**	integer	allows to select the server's `port number` on ; the port must be an integer and in the range `[1024, 65535]`, default: 8088

flag	long flag	input	type	Description
`-t`	`--time`	**seconds**	integer	the total duration in seconds for which data should be generated, also sent to the server (if it is set with -t flag at the client side) and must be > 0. **NOTE** If you do not use -t flag, your experiment should run for 25 seconds
`-f`	`--format`	MB	string	allows you to choose the format of the summary of results - it should be either in B, KB or MB, default=MB)
`-i`	`--interval`	z	integer	print statistics per z second

flag	long flag	input	type	Description
`-P`	`--parallel`	**no__of__conn**	integer	creates parallel connections to connect to the server and send data - it must be 1 and the max value should be 5 - default:1
`-n`	`--num`	**no__of__bytes**	string	transfer number of bytes specfied by -n flag, it should be either in B, KB or MB

If -c or -s flags are not specfied - you should print the following and exit:

```
Error: you must run either in server or client mode
```

> **NOTE:** When calculating the rate for the overall duration, measure the time elapsed from when the client first starts sending data to when it receives an ackonlowedgement (graceful close of connection) message from the server.

Running client with `-i` flag When simpleperf is invoked in client mode with -i flag, it will then print statistics per t seconds specified after the -i flag. Here is an example:

```
python3 simpleperf -c -I <server_ip> -p <server_port> -t <time>
-i 5

------------------------------------------------------------------------------------------
A simpleperf client connecting to server <IP>, port XXXX
------------------------------------------------------------------------------------------


Client connected with server_IP port XXXX

ID              Interval        Transfer    Bandwidth
```

```
IP:port        0.0 - 5.0        X MB       Y Mbps
IP:port        6.0 - 10.0       X MB       Y Mbps
IP:port        11.0 - 15.0      X MB       Y Mbps
IP:port        16.0 - 20.0      X MB       Y Mbps
IP:port        21.0 - 25.0      X MB       Y Mbps

-------------------------------------------------

IP:port        0.0 - 25.0       X MB       Y Mbps
```

Running client with `-n` or `--num` flag When simpleperf is invoked in client mode with -n flag, it will transfer the amount of bytes specified by -n and display the statistics

Here is an example:

```
python3 simpleperf -c -I <server_ip> -p <server_port> -n 10M
```

The client will establish a TCP connection with the simpleperf server and send 10MB data in chunks of 1000 bytes. Simpleperf will calculate and display the bandwidth.

Running client with `-P` or `--parallel` flag The client will establish parallel TCP connection with the `simpleperf` server and send data in chunks of 1000 bytes for 100 seconds (specified with `-t` flag). At the end of the transfer, `Simpleperf` client will calculate and display the bandwidth.

Here is an example where the client will open two TCP connections in parallel to connect with the server.

```
python3 simpleperf -c -I <server_ip> -p <server_port> -P 2 -t 100
```

```
------------------------------------------------------------------------------------------
A simpleperf client connecting to server <IP>, port XXXX
------------------------------------------------------------------------------------------

Client IP:port connected with server_IP port XXXX
Client IP: port connected with server_IP port XXXX


ID             Interval        Transfer    Bandwidth

IP:port       0.0 - 25.0        X MB       Y Mbps
IP:port       0.0 - 25.0        X MB       Y Mbps
```

> **NOTE:** the arguments must not be positional arguments, i.e., users do not need to remember the position of the arguments.

Performance evaluation

In this part of the assignment, you will use your `simpleperf` tool, and the standard latency measurement tool `ping` (`ping` measures RTT), to measure the bandwidth and latency in a virtual network in Mininet. In addition, you will also use iperf tool in UDP mode to measure the bandwidth.

You must include the output from some of your experiments and discuss the results in your report - **see the project report section below for the guidelines**. Please do **NOT** change the structure (you are welcome to add new sections or subsections in order to enhance the readability).

Follow my tutorial to learn how to use the `ping` tools. You can also look at the man pages - Type: `man ping`

A python script to run Mininet with the topology described below is also available on canvas. Download portfolio_topology.py file.

To run Mininet with the provided topology, run the Python script `portfolio_topology.py` using sudo:

```
sudo python portfolio_topology.py
```

This will create a network with the following topology:

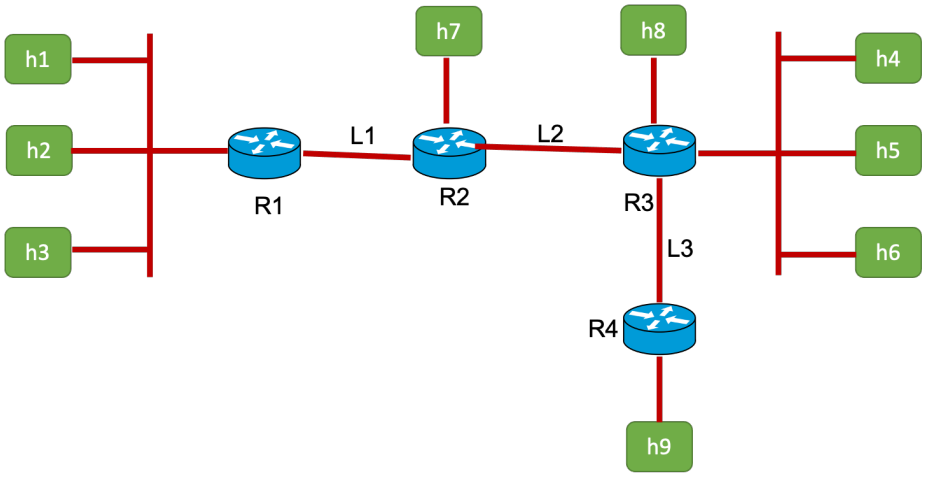

Figure 1: Network Topology for the portfolio task

If you have trouble running the script, a common fix is to first try running `sudo mn -c`, and then try running the script again.

There are 9 hosts (`h1` to `h9`) and 4 routers (`r1` to `r2`). See the portfolio_topology.py file for the ip addresses and how they are connected.

NOTE: When running ping and `simpleperf` in Mininet, you must use IP addresses, not hostnames.

Test case 1: Measuring bandwidth with iperf in UDP mode

Run three separate iperf tests with udp mode with -b `X`M between the follwing client-server pairs:

1. between h1 - h4
2. between h1 - h9
3. between h7 - h9

Store the output of these 3 separate measurements in `throughput_udp_iperf_h1-h4.txt`, `throughput_udp_iperf_h1-h9.txt`, and `throughput_udp_iperf_h7-h9.txt` files.

Question:

1. Which rate (`X`) would you choose to measure the bandwidth here? Explain your answers.
2. If you are asked to use iPerf in UDP mode to measure the bandwidth where you do not know anything about the network topology, what approach would you take to estimate the bandwidth? Do you think that this approach would be practical? Explain your answers.

Test case 2: Link Latency and Throughput

First, you should measure the RTT and bandwidth of each of the three individual links between routers (`L1` - `L3`). You should run ping with 25 packets and store the output of the measurement on each link in a file called `latency_L#.txt`, replacing # with the link number from the topology diagram above. You should run `simpleperf` for 25 seconds and store the output of the measurement on each link in a file called `throughput_L#.txt`, replacing # with the link number from the topology diagram above. Explain your results in the **Results and Discussion** section of your report.

Test case 3: Path Latency and Throughput

Run the following tests and measure path latency and thoughput:

- `h1` wants to communicate with `h4`. What is the expected latency and throughput of the path between the hosts? Measure the latency and throughput between `h1` and `h4` using `ping` and `simpleperf`. `h1` is the client and `h4` is the server. Use the same parameters as above (25 packets

/ 25 seconds) and store the output in files called `latency_h1-h4.txt` and `throughput_h1-h4.txt`. Add your results (only the average RTT and measured throughput) in your report and explain.

- `h7` wants to communicate with `h9`. What is the expected latency and throughput of the path between the hosts? Measure the latency and throughput between `h7` and `h9` using `ping` and `simpleperf`. `h7` is the client and `h9` is the server. Use the same parameters as above (25 packets / 25 seconds) and store the output in files called `latency_h7-h9.txt` and `throughput_h7-h9.txt`. Add your results (only the average RTT and measured throughput) in your report and explain.
- `h1` wants to communicate with `h9`. What is the expected latency and throughput of the path between the hosts? Measure the latency and throughput between `h1` and `h9` using `ping` and `simpleperf`. `h1` is the client and `h9` is the server. Use the same parameters as above (25 packets / 25 seconds) and store the output in files called `latency_h1-h9.txt` and `throughput_h1-h9.txt`. Add your results (only the average RTT and measured throughput) in your report and explain.

Test case 4: Effects of Multiplexing and latency

In this experiment, you will look at the effects of multiplexing. You will run the following experiments.

1. Two pairs of hosts (h1-h4 and h2-h5) will simultaneously communicate using `simpleperf`. Use `ping` and `simpleperf` to measure the latency and throughput when (h1-h4 and h2-h5) communicating simultaneously; store the output in files called `throughput_h1-h4-1.txt`, `throughput_h2-h5-1.txt`, `latency_h1-h4-1.txt`, and `latency_h2-h5-1.txt`.Use the same parameters as above (25 packets / 25 seconds). Mention average RTT and measured throughput for each pair in results section and explain the results.
2. Repeat for three pairs of hosts (h1-h4, h2-h5, and h3-h6) communicating simultaneously and write your results in the discussion section. Store the output in files called `throughput_h1-h4-2.txt`, `throughput_h2-h5-2.txt`, `throughput_h3-h6-2.txt`, `latency_h1-h4-2.txt`, `latency_h2-h5-2.txt`, and `latency_h3-h6-2.txt`. Use the same parameters as above (25 packets / 25 seconds). Mention average RTT and measured throughput for each pair in results section and explain the results.
3. Repeat for h1-h4 and h7-h9 communicating simultaneously; store the output in files called `throughput_h1-h4-3.txt`, `throughput_h7-h9-3.txt`, `latency_h1-h4-3.txt` and `latency_h7-h9-3.txt`. Use the same parameters as above (25 packets / 25 seconds). Mention average RTT and

measured throughput for each pair in results section and explain the results.

4. Repeat for h1-h4 and h8-h9 communicating simultaneously; store the output in files called `throughput_h1-h4-4.txt`, `throughput_h8-h9-4.txt`, `latency_h1-h4-4.txt` and `latency_h8-h9-4.txt`. Use the same parameters as above (25 packets / 25 seconds). Mention average RTT and measured throughput for each pair in results section and explain the results.

Test case 5: Effects of parallel connections

In this experiment, `h1`, `h2` and `h3` connected to `R1` will simultaneously talk to hosts (`h4`, `h5` and `h6`) connected to `R3`. `h1` will open two parallel connections (with -P flag in the client mode) to communicate with `h4`, and `h2` and `h3` will just invoke the normal client (without -P flag). Measure the throughput when three pairs of hosts are communicating simultaneously using `simpleperf` and store your results in the files called `throughput_h1-h4.txt`, `throughput_h2-h5.txt` and `throughput_h3-h6.txt` in the measurement/test-case-6 folder. Use the same parameters as above (total duration: 25 seconds). Explain your results in the discussion section.

> **NOTE** Do not worry too much about starting the clients at the exact same time. As long as the connections overlap significantly, you should achieve the correct results. I suggest you open up terminals for each of the hosts you will use, start the `simpleperf` servers, type in the `simpleperf` client command on each of the client hosts without hitting ENTER, and then quickly hit ENTER on all client hosts so that they start at roughly the same time.

Submission

You must submit **yourname_studentid_portfolio1.zip** through the Inspera exam system. You will get access to inspera two weeks before the deadline.

Your zip file should contain:

1. Source code of simpleperf
   - where the code is well commented. Document all the variables and definitions. For each function in the program, you must document the following:
     - what are the arguments.
     - What the function does.
     - What input and output parameters mean and how they are used.
     - What the function returns.
     - Correctly handling exceptions.

Here is an example:

```
# Description:
# checks if the port and addresses are in the correct format
# Arguments:
# ip: holds the ip address of the server
# port: port number of the server
# Use of other input and output parameters in the function
# checks dotted decimal notation and port ranges
# Returns: .... and why?
#
```

2. A report with your name, ID and title. Your document must be submitted in the pdf format.
   - read the instructions (see below)
3. Your measurement results should be in a folder called `measurements`.
4. A README file: info on how to run simpleperf and tests to generate data

Example final structure of your folder:

```
README
portfolio_topology.py
simpleperf
    simpleperf.py
    utils (if any)
yourname_studentid_portfolio1.pdf
measurements
    test-case-1
        throughput_udp_iperf_h1-h2.txt
        throughput_udp_iperf_h7-h9.txt
    test-case-2
        latency_L1.txt
        latency_L2.txt
        latency_L3.txt
        throughput_L1.txt
        throughput_L2.txt
        throughput_L3.txt
    test-case-3
        latency_h1-h4.txt
        latency_h7-h9.txt
        latency_h1-h9.txt
        throughput_h1-h4.txt
        throughput_h7-h9.txt
        throughput_h1-h9.txt
    test-case-4
        throughput_h1-h4-1.txt
        throughput_h2-h5-1.txt
        throughput_h1-h4-2.txt
```

```
    throughput_h2-h5-2.txt
    throughput_h3-h6-2.txt
    throughput_h1-h4-3.txt
    throughput_h7-h9-3.txt
    throughput_h1-h4-4.txt
    throughput_h8-h9-2.txt
    latency_h1-h4-1.txt
    latency_h2-h5-1.txt
    latency_h1-h4-2.txt
    latency_h2-h5-2.txt
    latency_h3-h6-2.txt
    latency_h1-h4-3.txt
    latency_h7-h9-3.txt
    latency_h1-h4-4.txt
    latency_h8-h9-4.txt
test-case-5
    throughput_h1-h4.txt
    throughput_h2-h5.txt
    throughput_h3-h6.txt
```

NOTE I strongly recommend that you confirm that the archive you uploaded contains what it should by downloading it and unpacking it again after submission. If you deliver the wrong files, there is nothing I can do about it when I check. We also recommend you upload draft versions as you work, to ensure you have a deliverable in the system should you experience last minute technical difficulties etc.

Project report

You also need to submit a project report in the pdf format. **NOTE**: I won't accept word or any other format. The report must have a title page (your name, ID and the title of your work), table of contents, and the following sections:

1. Introduction
2. Implementation of Simpleperf
3. Experimental setup
4. Performance evaluations
    1. Network tools for perfomance evaluation
    2. Perfomance metrics
    3. Test case 1: measuring bandwidth with iperf in UDP mode
        1. Results
        2. Discussion
    4. Test case 2: link latency and throughput
        1. Results
        2. Discussion

    5. Test case 3: path Latency and throughput
        1. Results
        2. Discussion
    6. Test case 4: effects of multiplexing and latency
        1. Results
        2. Discussion
    7. Test case 5: effects of parallel connections
        1. Results
        2. Discussion
5. Conclusions
6. References (if any: mention sources)

The report cannot exceed 20 pages, including the list of references. The page format must be A4 with 2 cm margins, single spacing and Arial, Calibri, Times New Roman or similar 11-point font.

Grading

- Implementation of `simpleperf` (including section 4 (Implementation) in the project report and README file) - `45%`
- Performance evaluation of a virtual network in mininet with `simpleperf` and other tools (including section 5 (Performance evaluations) in the project report) - `45%`
- Project report - `10%`

You will lose points upto **10-20%** if you do not follow the submission guidelines and report format.

Deadline

The deadline for submitting this exam is **Monday April 17 2023 at 12:00** Oslo local time.

This is a HARD deadline, failure to deliver on time will result in fail in this part-exam.

I am not going to handle any enquiries regarding medical extensions and so forth, you must contact the study administration directly.

Follow the submission guidelines. Do not make your respository public, and *do not copy.*