Lab 1: Sockets, Mininet, & Performance
CS640 Fall 2022

Released: Tuesday Sep 13 2022
Due: Tuesday Sep 27 2022 11:59PM
Total points: 60

Important notes

- All answers and code that you write for this lab must be your own work. Plagiarism is a serious violation of university statutes and all answers will be thoroughly checked for it.
- Ensure that your answer has proper headings and subheadings wherever necessary.
- Late policy:
    - Up to 30 minutes late – lose 0% points
    - Up to 24 hours late — lose 10% of points
    - Up to 48 hours late — lose 30% of points
    - Up to 72 hours late — lose 60% of points
    - Beyond 72 hours — lose 100% of points

Overview

Iperf is a common tool used to measure network bandwidth. You will write your own version of this tool in Java using sockets. You will then use your tools to measure the performance of virtual networks in Mininet and explain how link characteristics and multiplexing impact performance.

Learning Outcomes

After completing this lab, students should be able to:
- Write applications that use sockets to transmit and receive data across a network
- Describe how latency and throughput can be measured
- Explain how latency and throughput are impacted by link characteristics and multiplexing

===

Part 1: Write Iperfer

For the first part of the lab, you will write your own version of iperf to measure network bandwidth. Your tool, called ***Iperfer***, will send and receive TCP packets between a pair of hosts using sockets. Use java jdk version >= 8.

- Note: A good resource and a starting point to learn about Java socket programs is the Java sockets tutorial.

When operating in client mode, ***Iperfer*** will send TCP packets to a specific host for a specified time window and track how much data was sent during that time frame; it will calculate and display the bandwidth based on how much data was sent in the elapsed time. When operating in server mode, ***Iperfer*** will receive TCP packets and track how much data was received during the lifetime of a connection; it will calculate and display the bandwidth based on how much data was received and how much time elapsed between the received first and last byte of data.

Client Mode

To operate ***Iperfer*** in client mode, it should be invoked as follows:

```
java Iperfer -c -h <server hostname> -p <server port> -t <time>
```

- `-c` indicates this is the iperf client which should generate data
- `server hostname` is the hostname or IP address of the iperf server which will consume data
- `server port` is the port on which the remote host is waiting to consume data; the port should be in the range 1024 ≤ server port ≤ 65535
- time is the duration in seconds for which data should be generated

You can use the presence of the `-c` option to determine if ***Iperfer*** should operate in client mode.

If any arguments are missing or additional arguments are given or arguments are given in wrong order, you should print the following and exit:

- `Error: invalid arguments`

If the `server port` argument is less than 1024 or greater than 65535, you should print the following and exit:

- `Error: port number must be in the range 1024 to 65535`

On the `time` and `hostname` input values, provide input validations that you think are reasonable. As always, state your assumptions through comments in code.

When running as a client, ***Iperfer*** must establish a TCP connection with the server and send data as quickly as possible for `time` seconds. Data should be sent in chunks of **1000** bytes and the data should be a byte-array of all **zeros**. Keep a running total of the number of bytes sent.

After *`time`* seconds have passed, ***Iperfer*** must stop sending data and close the connection. ***Iperfer*** must print a one line summary that includes:

- The total number of bytes sent (in kilobytes)
- The rate at which traffic could be sent (in megabits per second (Mbps))

For example:
```
sent=6543 KB rate=5.234 Mbps
```
You should assume 1 kilobyte (KB) = 1000 bytes (B), 1 kilobit (Kb) = 1000 bits (b) and 1 megabit (Mb) = 1000 Kb. As always, 1 byte (B) = 8 bits (b).

Server Mode

To operate ***Iperfer*** in server mode, it should be invoked as follows:
```
java Iperfer -s -p <listen port>
```
- `-s` indicates this is the iperf server which should consume data
- `listen port` is the port on which the host is waiting to consume data; the port should be in the range 1024 ≤ listen port ≤ 65535

You can use the presence of the `-s` option to determine if ***Iperfer*** should operate in server mode.

If arguments are missing or additional arguments are provided or arguments are given in wrong order, you should print the following and exit:
- `Error: invalid arguments`

If the `listen port` argument is less than 1024 or greater than 65535, you should print the following and exit:
- `Error: port number must be in the range 1024 to 65535`

When running as a server, ***Iperfer*** must listen for TCP connections from a client and receive data as quickly as possible until the client closes the connection. Data should be read in chunks of **1000** bytes. Keep a running total of the number of bytes received.

After the client has closed the connection, ***Iperfer*** must print a one line summary that includes:
- The total number of bytes received (in kilobytes)
- The rate at which traffic could be read (in megabits per second (Mbps))

For example:
- `received=6543 KB rate=4.758 Mbps`

Note: The *Iperfer* server should shut down after it handles one connection from a client.

Testing

We would like you to test the ***Iperfer*** tool on two environments - a wired environment and a wireless environment.

Wired Environment

Run ***Iperfer*** server and client on two separate physical machines that are connected by cable. You can test ***Iperfer*** on any machines you have access to. However, be aware that certain ports may be blocked by firewalls on end hosts or in the network, so you may not be able to test your program on all hosts or in all networks. You should be able to use or ssh into two machines in the CS labs to do ***Iperfer*** without any problems.

Take a screenshot of the measurements (both client and server) when running the tool on the terminal and insert them in a file called `answers.pdf` under the heading **Part 1 - Wired Environment**. Be sure to mark your screenshots with appropriate labels.

Wireless Environment

Now that you have tested ***Iperfer*** in a wired environment, think about what would happen in a wireless environment. Under which environment - wired or wireless, would you expect greater throughput? Note your answer along with the reasoning in `answers.pdf` under the heading **Part 1 - Wireless Environment**.

To measure the throughput in a wireless environment, you would first need to set up a wireless environment and then connect two machines through the wireless router - one that would act as a client and the other that would act as the server, in order to measure the throughput.

There are many ways to set up wireless environments. The most common ways are listed below:

- You could simply connect two machines (laptop) to a wireless router. You can then run the server on one machine and the client on another machine. Beware that routers or firewall settings might cause certain ports to be blocked.
- Connect two machines to UW Net and do the experiment. For example,you can test with one CSL machine and your own laptop under UW Net.
- You could set up a wireless hotspot on most desktops/laptops with some of the latest operating systems.

Take a screenshot of the measurements (both client and server) when running the tool on the wireless environment and insert them into `answers.pdf` under **Part 1 - Iperfer on Wireless Environment**. Be sure to mark your screenshots with appropriate labels. Now compare the throughputs between the wired and wireless environments you had obtained. Did it match the prediction you made at the start of this section? In either case, explain your results. Put your answers into `answers.pdf` under **Part 1 - Iperfer Results**.

Testing on Mininet (Optional step for testing Iperfer)

You can also test your tool using Mininet. Instructions are provided in Appendix A. You should complete Part 2 of this lab before following the instructions in Appendix A.

You should receive the same number of bytes on the server as you sent from the client. However, the timing on the server may not perfectly match the timing on the client. Hence, the bandwidth reported by client and server may be slightly different; in general, they should not differ by more than 2 Mbps. Note, this behavior mirrors the behavior of the actual iperf tool.

==

Part 2: Mininet Tutorial

For the second part of the lab, you will learn how to use Mininet to create virtual networks and run simple experiments. According to the Mininet website, Mininet creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM or native), in seconds, with a single command. We will use Mininet in most labs throughout the semester.

Running Mininet

To run Mininet, you will need to download a VM on your personal machine. Follow the instructions on http://mininet.org/download/ and http://mininet.org/vm-setup-notes/. Download the Ubuntu 20.04.1 VM image that is provided with Mininet release 2.3.0.

Also, you will need to have X11 installed on the machine from which you are connecting. You can install Xming (Windows), XQuartz (Mac OS), or a graphical environment (Linux) to get X11. You should only need X11 to run xterms in Mininet. You might need to use vim/emacs (or others) to edit files.

For Mac Users (instructions above should have everything, so follow those first, but if you are having issues check these steps):

- Follow download instructions for Mininet and VirtualBox
- Create a host only adapter in VirtualBox.
    - Go to File -> Host Network Manager … -> create
    - Keep auto settings and close
    - Go to Settings -> Network -> Adapter 2 -> click Enable Network Adapter
    - Attached to: Host-only Adapter
      Name: vboxnet0 (or whatever you named the host network)
- Log into Mininet (Username: mininet Password: mininet)
    - `ifconfig`
    - `dhclient eth1`
    - `ifconfig`
- SSH from Quartz
    - `ssh -X mininet@[IP eth0]`
- Check that xterm works by running the `xterm` command, then `sudo mn -x`
    - If `xterm` didn't work, X11 is not being passed
    - If `sudo mn -x` doesn work
        - `xauth list $DISPLAY`
        - Copy that line.
        - `sudo -s`
        - `xauth add [LINE_YOU_COPIED]`

        - Try again

For Windows Users (if you have issues with the above instructions):

- Set up and Log in to MiniNet -
    - Download MiniNet and VirtualBox
    - Setup and Login Virtual Box - [Mininet VM Setup Notes - Mininet]
- Before SSH into the VM perform the following:
    - Download PuTTY - [Download Putty (0.76) for Windows, Linux and Mac]
      (PuTTY is recommended for Windows)
    - Read - [SSH into VM]
    - Setup PuTTY - hostname, username, (optional key based login), enable X11 forwarding
    - Finally SSH into VM

- [OPTIONAL]
    - Use key based login for PuTTY - [Use SSH Keys with PuTTY on Windows]
    - Incase Xming does not work-
        - Download VcXsrc - [VcXsrv Windows X Server download]
        - Run VcXrc
        - In PuTTY enable X11 Forwarding option
        - SSH into VM using PuTTY
        - In the PuTTY terminal edit the file bashrc :
            - `nano ~/.bashrc`
        - Add a line in the end :
            - `export XAUTHORITY=~/.Xauthority`
        - In the PuTTY terminal type :
            - `source ~/.bashrc`
        - Read - [Mininet Walkthrough - Mininet]

To transfer files to/from your VM you should use the `scp` (secure copy) command. See the `scp` man page, or find a tutorial online, for instructions on how to use scp. You could also use a revision control system, such as Git or Subversion, and checkout a copy of the repository in your VM.

Mininet Walkthrough

Once you have a Mininet VM, you should complete the following sections of the standard Mininet walkthrough:

- All of Part 1, except the section "Start Wireshark"
- The first four sections of Part 2—"Run a Regression Test", "Changing Topology Size and Type", "Link variations", and "Adjustable Verbosity"
- All of Part 3

At some points, the walkthrough will talk about software-defined networking (SDN) and OpenFlow. We will discuss these during the second half of the semester, so you do not need to understand what they mean right now; you just need to know how to run and interact with Mininet.

To change the colors of xterm windows (black font on white background) to make them more readable, run the following command in your Mininet VM (in a regular shell, not while Mininet is running):

- `echo XTerm*Foreground: black > ~mininet/XTerm`
- `echo XTerm*Background: white >> ~mininet/XTerm`

You do not need to submit anything for this part of the lab.

==

Part 3: Measurements in Mininet

For the last part of the lab you will use the tool you wrote (**Iperfer**) and the standard latency measurement tool `ping` (ping measures Round-trip Time or RTT) to measure the bandwidth and latency in a virtual network in Mininet. You must include the output from some of your experiments and the answers to the questions below in your submission. Your answers to the questions should be put in the file `answers.pdf`.

Read the `ping` man page to learn how to use ping.

You must install Java in your Mininet VM before you can run **Iperfer**. You need to use at least java version 8 (we tested the VM with Java 8 and 11). As an example, you can install Java 8 by running the following commands:

- `sudo apt-get update`
- `sudo apt-get install openjdk-8-jdk`

A python script to run Mininet with the topology described below is located at: https://pages.cs.wisc.edu/~mgliu/CS640/F22/labs/lab1/lab1.tgz. To download the script in your Mininet VM, run:

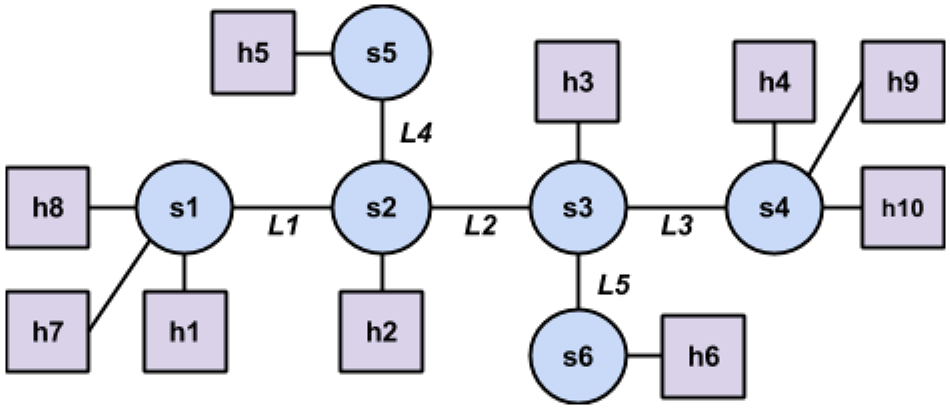
- `wget https://pages.cs.wisc.edu/~mgliu/CS640/F22/labs/lab1/lab1.tgz`
- `tar xzvf lab1.tgz`

Q1: Link Latency and Throughput

To run Mininet with the provided topology, run the Python script `lab1_topo.py` using sudo:

- `sudo python lab1_topo.py`

This will create a network with the following topology:



Hosts (`h1 - h10`) 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 Mininet. The hosts are assigned IP addresses `10.0.0.1` through `10.0.0.10`; the last number in the IP address matches the host number. When running `ping` and ***Iperfer*** in Mininet, you **must use IP addresses**, not hostnames.

First, you should measure the RTT and bandwidth of each of the five individual links between switches (`L1 - L5`). You should run ping with 30 packets and store the output on each link in a file called `latency_L#.txt`, replacing # with the link number from the topology diagram above. You should also run ***Iperfer*** for 30 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.

Q2: Path Latency and Throughput

Now, assume `h1` wants to communicate with `h4`. What is the expected latency and throughput of the path between these two hosts? Put your prediction in your `answers.pdf` file under heading **Part 3 - Q2 Predictions**.

Measure the latency and throughput between h1 and h4 using `ping` and ***Iperfer***. It does not matter which host is the client and which is the server. Use the same parameters as above (30 packets / 30 seconds) and store the output in files called `latency_Q2.txt` and `throughput_Q2.txt`. Put the average RTT and measured throughput in your `answers.pdf` file under heading **Part 3 - Q2 Results** and explain the results. If your prediction was wrong, explain why.

Q3: Effects of Multiplexing

Next, assume multiple hosts connected to s1 want to simultaneously talk to hosts connected to s4. What is the expected latency and throughput when two pairs of hosts are communicating simultaneously? Three pairs? Put your predictions in your `answers.pdf` file under heading **Part 3 - Q3 Predictions**.

Use `ping` and ***Iperfer*** 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`. Use the same parameters as above. You do not need to submit the raw output, but you should put the average RTT and measured throughput for each pair in your `answers.pdf` file under heading **Part 3 - Q3 Results** and explain the results. If your prediction was wrong, explain why.

Repeat for three pairs of hosts communicating simultaneously.

Q4: Effects of Latency

Lastly, assume `h1` wants to communicate with `h4` at the same time `h5` wants to communicate with `h6`. What is the expected latency and throughput for each pair? Put your prediction in your `answers.pdf` file under heading **Part 3 - Q4 Predictions**.

Use `ping` and ***Iperfer*** to conduct measurements, storing the output in files called `latency_h1-h4.txt`, `latency_h5-h6.txt`, `throughput_h1-h4.txt`, and `throughput_h5-h6.txt`. Put the average RTT and measured throughput in your `answers.pdf` file under heading **Part 3 - Q4 Results** and explain the results. If your prediction was wrong, explain why.

==

Submission Instructions

You must submit(one submission for each group):

- The source code for ***Iperfer***—all Java source files for Iperfer should be in a folder called `iperfer`; the folder should include a Makefile that compiles the Java source files.
- Your measurement results and answers to the questions from Part 3—all measurements captured in text files and answers to questions in the file `answers.pdf` should be in a folder called measurement
- A `README` file with the names and CS usernames of all group members, and assumptions made.

You must submit a single tar file containing the above. To create the tar file, run the following command, replacing username1, username2(just username1 if you are doing the lab by yourself) with the CS username of each group member:

- `tar czvf username1_username2.tgz iperfer measurement README`
- `tar czvf username1.tgz iperfer measurement README`

Upload the tar file on the Lab1 tab on course's Canvas page. Please submit only one tar file per group.

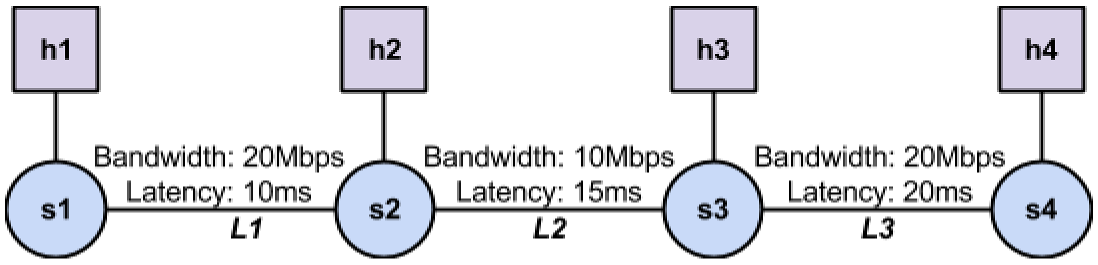
==

Appendix A: Testing Iperfer in Mininet

You must install Java in your Mininet VM before you can run ***Iperfer***. You can install Java by running the following commands:

- `sudo apt-get update`
- `sudo apt-get install openjdk-8-jdk`

You can test ***Iperfer*** in Mininet using the sample topology shown below.



A python script to run Mininet with this topology described below is located at: https://pages.cs.wisc.edu/~mgliu/CS640/F22/labs/lab1/lab1.tgz. To download the scripts in your Mininet VM, run:

- `wget https://pages.cs.wisc.edu/~mgliu/CS640/F22/labs/lab1/lab1.tgz`
- `tar xzvf lab1.tgz`

To run Mininet with the provided topology, run the Python script lab1_test.py using sudo:

- `sudo python lab1_test.py`