Programming Assignment 1: Sockets, Mininet, & Performance

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 programming assignment, students should be able to:

- Use Mininet to do network experiments
- Describe how latency and throughput can be measured
- Explain how latency and throughput are impacted by link characteristics and multiplexing

Clarifications

- The `Iperfer` server should shut down after it handles one connection from a client.

Step 1: Write Iperfer

For the first step of the assignment 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.

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 received the 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 `Iperfer` should operate in client mode.
If any arguments are missing or additional arguments are provided, you should print the following and exit:

```
Error: missing or additional 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
```

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 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) and 1 megabyte (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 `Iperfer` should operate in client mode.
If arguments are missing or additional arguments are provided, you should print the following and exit:

```
Error: missing or additional 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
```

The `Iperfer` server should shut down after it handles one connection from a client.

Testing

You can test `Iperfer` on any machines you have access to.
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.

Step 2: Mininet Tutorial

Prepare the Mininet environment

For this step, 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 the following programming assignments.

To prepare the Mininet environment, you have two options.

(1) Install Virtual Box and run the Ubuntu OS image prepared by the course. This Ubuntu OS image has already installed the Mininet. After starting the Ubuntu on Virtual Box, you are ready to use Mininet.

This option is recommended!!

Download the Ubuntu+Mininet image from URL (TO BE ANOUNCED BY TA IN THE WECHAT GROUP)

The download password is *123456*.

After you install VirtualBox and download the Ubuntu image. Import the Ubuntu image in VirtualBox, then you can start the Unbuntu there. Note the root passcode is "*stu*".

After the Ubuntu is running, open a terminal and then you can start using Mininet.

(2) If you prefer to use Mininet of your existing linux, you can do the following.

```
cd ~
```

```
sudo apt-get install mininet
```

This option is NOT recommended!!

Mininet Walkthrough (Note: do not download the image provided by the Mininet website)

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 later, so you do not need to understand what they mean right now; you just need to know how to run and interact with Mininet. You do not need to submit anything for this step of the assignment.

Step 3: Measurements in Mininet

For the last step of the assignment you will use the tool you wrote (`Iperfer`) and the standard latency measurement tool `ping` (ping measures 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 a file called `answers.txt`.

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 can install Java by running the following commands:

```
  sudo apt-get update
  sudo apt-get install openjdk-7-jdk
```

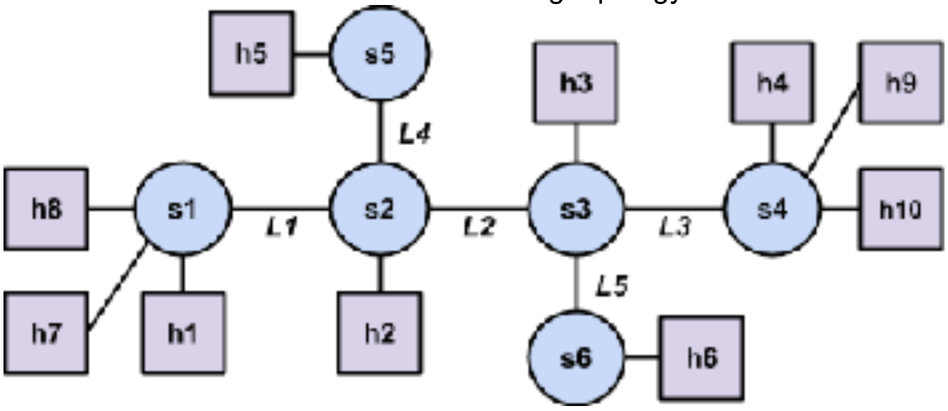
A python script to run Mininet with the topology described below is called `progAssign1_topo.py`

Q1: Link Latency and Throughput

To run Mininet with the provided topology, run the Python script `progAssign1_topo.py` using sudo:

```
   sudo python progAssign1_topo.py
```

This will create a network with the following topology:



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 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 20 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 `Iperfer` for 20 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 the hosts? Put your prediction in your `answers.txt` file.

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 (20 packets / 20 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.txt` file 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.txt` file.
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.txt` file 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.txt` file.
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.txt` file and explain the results. If your prediction was wrong, explain why.

Submission Instructions

You must submit:

- The source code for `Iperfer.` A readme file explaining how to run your code.
- Your measurement results and answers to the questions from Step 3—all results and a text file called `answers.txt` should be in a folder called `measurement`

You must submit a single tar or zip file named `assign1_username` containing everything.

Grading

1) Code quality and readability. 20%

2) Measurement 80%

Acknowledgements

We acknowledge CS 640: Computer Networks with University of Wisconsin-Madison.