CS 640 Introduction to Computer Networks **Spring'24**

Assignment 1

Due at noon on: Feb 13

Overview

`iperf` is a common tool used to measure network bandwidth. You will write your own version of this tool in Java using sockets (and we call it, well, "Iperfer"). You will then use your tools to measure the performance of virtual networks in Mininet and explain how link characteristics and multiplexing impact performance.

Environment Setup

Option 1: Use a CSL machine for Part 1, and ssh into mininet VM for Part 2.
To run Mininet, you will use the mininet VMs which have already been assigned to you by the course staff. You have one VM per group.
Please connect yourself to the WiscVPN or an instructional lab computer before connecting to your VM. You and your partner should have already received a password from us, which will be your credential when SSH into the VM. This password is shared in your group only, and works only on the VM you are assigned. When you connect via SSH, you may include the `-X` option to tunnel X11 (i.e., graphics) over SSH, assuming you have a X11 server running locally. For example if you are assigned VM `mininet-85`:

```
ssh -X mininet@mininet-85.cs.wisc.edu
```

You will need to have X11 installed on the machine from which you are connecting. On a personal machine, you can install Xming on Windows, XQuartz on Mac OS, or any graphical environment / desktop environment on Linux to get X11. You should only need X11 to run xterms in Mininet. For editing files now is the opportune time to learn a text editor that doesn't require a GUI! Vim and emacs are the most popular ones.
To transfer files to/from your VM, you may use `scp` or `rsync`. See man pages for `scp` and `rsync` or find a tutorial online for detailed instructions. You could also use a version control system, such as Git or Subversion, and checkout a copy of the repository in your VM. If you opt to use a version control system, please make sure you don't inadvertently allow others (other than your group members) to access your work.

Option 2: Download the VM image, and import it into VirtualBox installed on your own PC.
If you have issue accessing your VM or the VPN isn't working out for you, you may also download a pre-configured system image from here. (Warning: 2GB download, 10GB decompressed). The image can be launched locally in VirtualBox - a virtualization software available on Windows, Linux and Mac OS w/intel chip that you may install on your personal computer. For Mac OS with apple chip, please use the image from here and launch it with UTM. The image will have the same setup as the VM with the only addition of a graphical environment for your convenience. The instructions for assignments also work exactly the same on your local VirtualBox as the CS VM assigned to you.
FYI: The machine is configured to auto-login and you will not be prompted for password from sudo. The password for the `mininet` user is `user`, and `root` for the root user. Feel free to change them if you wish.

Please make sure your deliverables compile and run on either one of these two options. This is the environment where we grade your assignments.

Learning Outcomes

After completing this programming assignment, students should be able to:

1. Write applications that use sockets to transmit and receive data across a network
2. Describe how latency and throughput can be measured
3. Explain how latency and throughput are impacted by link characteristics and multiplexing

1 Implement your Iperfer

For the first part of the assignment you will write your own version of iperf in Java 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. Although this guide was written for Java 8, the socket API has not changed and is still applicable to latest version.

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 $1024 \leq$ server port ≤ 65535.
- Time is the duration in seconds for which data should be generated.
- You may assume that parameters are always passed in this order.

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 client must stop sending data and close the connection. Before your program terminates, it must print a one line summary including:

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

```
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 $1024 \leq$ listen port ≤ 65535.

You can use the presence of the `-s` option to determine Iperfer should operate in server 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 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 server must also print a one line summary that includes the following and then terminate:

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

```
received=6543 KB rate=4.758 Mbps
```

Testing Your Implementation

You can test Iperfer on any computer you have access to. However, be aware the 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 two machines in the CS labs without any problems. You can also test your tool using Mininet. Instructions are provided in Appendix A. You should complete Part 2 of this assignment 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.

2 Mininet Tutorial

For the second part of the assignment, 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 programming assignments throughout the semester.

Once you have a Mininet VM, you should complete the following sections of the standard 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 assignment.

3 Measurements in Mininet

For the last part 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.

Java 11 has been preinstalled on your VMs, you may compile your implementation of Iperfer on your VM. A python script to run Mininet with the topology described below is located at: assign1.tgz.

Extract the scripts using the following command:

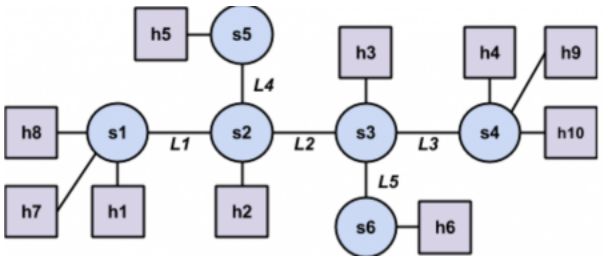
```
tar -xzvf assign1.tgz
```

Q1: Link Latency and Throughput

To run Mininet with the provided topology, run the Python script `assign1_topo.py` as root:

```
sudo python assign1_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? What about 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 and record your answers in `answers.txt` as well.

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 - all Java source files for Iperfer should be in a folder called `iperfer`; the folder should include a Makefile that compiles the Java source
- Your measurement results and answers to the questions from Part 3 - all results and `answers.txt` should be in a folder called `measurement`
- A README file with the names and CS usernames of both group members

You must submit a single `tgz` file containing the above. Note that `tar.gz` is equivalent to `tgz` internally, but please use the `tgz` extension. To create the tar file, run the following command, replacing `cslogin1` and `cslogin2` with the CS username of each group member:

```
tar -czvf cslogin1_cslogin2.tgz iperfer measurement README
```

Upload the tar file to the Assignment 1 dropbox on canvas. Please submit only one tar file per group.

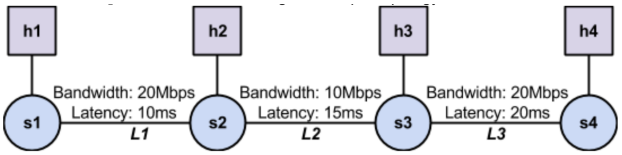
Late Submission Policy

Late submission policy for the assignments (any reason, e.g., sickness, not working system, etc.):

- Upto 24 hours late — You'll lose 30% of points
- Upto 48 hours late — You'll lose 60% of points
- Beyond 48 hours — You'll lose 100% of points

Appendix A: Testing Iperfer in Mininet

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



A python scripts to run Mininet with this topology described below is included in the `assign1.tgz` provided to you.

To run Mininet with the provided topology, run the Python script `assign1_test.py` using sudo:

```
sudo python assign1_test.py
```