CS378 Lab 5: Bottleneck Bandwidth Estimation using UDP Sockets

Note: This lab can be done individually or in a group of 2. You will be required to specify in a Google form before an early deadline if you plan to do this lab in a group or individually. You are expected to conform to what you say in the Google Form. Read the problem statement before making your decision.

Overall Goal of Lab: Finding the bottleneck bandwidth on an end-to-end path

What is Bottleneck Bandwidth?

Suppose a network path between source S and destination D in the Internet has "n" intermediate routers: R1, R2, …, Rn (see Figure below). Let us assume that each of the (n+1) links on the end-to-end path has link speed C_i bits/sec, where link i=1 corresponds to the link between S and R1, i=2 corresponds to the link between R1 and R2 and so on, and i=(n+1) corresponds to the link between Rn and D. A packet sent from S to D crosses links 1, 2,...,n+1. It enters router Rj on link j and exits Rj on link (j+1). Router Rj (for every j) has an input queue into which it receives packets. Only when a complete packet (that is all bits including the last bit in the packet) is received in the input queue is it forwarded to an output queue corresponding to link (j+1). Both queues are FIFO queues. The output queue is emptied with constant bit-rate C_{j+1}.

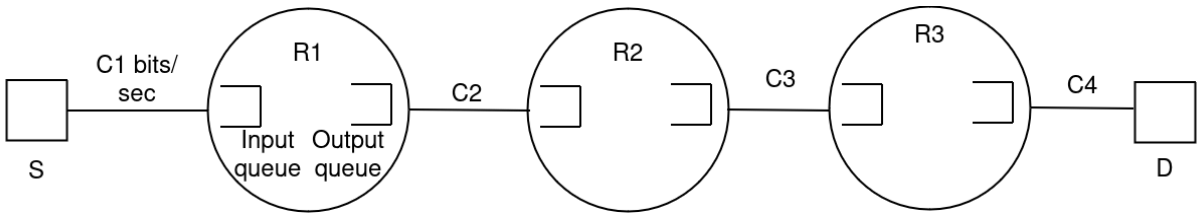

Figure: Network path between Source S and Destination D. In this example there are 4 links on the path. A packet from S to D passes through input and output FIFO queues at every router.

The bottleneck link speed "C" is defined as the smallest link speed among all links of a path.

Why is knowledge of bottleneck link speed important?

The maximum throughput of any application sending data from S to D is upper bounded by "C". The higher the value of C, the better.

Part 1: (Theory) Suppose S sends out two packets of size P bits each, one after the other, that is the first bit of the second packet is transmitted onto link 1 immediately after the last bit of the first packet. We are ignoring lower layer headers here, and approximating them to zero. **Assume that there are no other packets in any of the queues on the end-to-end path.**

Suppose t1 is the time of arrival of the last bit of the first packet at D, and t2 is the time of arrival of the last bit of the second packet at D.

In your report: Prove that P/(t2-t1) equals bottleneck link speed C.

(*Hint:* you can try to prove the result by induction: that is show that given the result is true for a sub-path consisting of the first *k* links, it is also true for a sub-path consisting of the first *(k+1)* links. You will have to replace C by the bottleneck link speed for the various sub-paths in this proof. In addition, you must prove why the statement is true for k=1.)

Part 2: (Implementation)

It seems, in theory, that we can now estimate the bottleneck link speed simply by sending two packets out, one after the other, and measuring their spacing at the destination. You must build a tool which accomplishes this using Datagram Sockets.

What are Datagram Sockets? If an application needs to use the UDP transport protocol, then it does so using a “Datagram Socket”. Think of a socket as a portal or interface between the application and transport layers. An application can create a Datagram Socket to which it binds a “port” number. The port number along with the IP address identifies that socket. The application can read data from the socket and also write data to it.

You can learn all about network sockets from Beej’s guide to network programming. This guide describes “Stream Sockets” which use TCP as the transport layer in addition to Datagam Sockets. We won’t need Stream Sockets in this lab though.

Your Network Tool: Your network measurement tool should have two programs sender.c and receiver.c. The former (sender.c) should take as command line input

- the packet size in bits “P”
- the destination IP address
- the spacing in time (in milli seconds) between successive packet-pairs (for example, if this is 1000, then one pair of back-to-back packets is sent out and after a wait of 1000 ms another one is sent out)
- the total number of packet-pairs to send out

Each packet should contain a “packet number” field which is 1 for the first packet and is incremented for each subsequent packet. You can send any additional information in the packets, as you think appropriate.

The latter (receiver.c) is run at the receiver. It should take as command-line input:

- Name of output text file

Receiver.c measures the spacing between every packet-pair (with at least micro-second accuracy). It prints one value of P/(t2-t1) to the output text file for every successfully received pair of packets. Note that if one packet in a pair is dropped (due to any reason), then no value should be printed out corresponding to that pair. **Each value of P/(t2-t1) is called an estimate of “C”.**

Note: You are allowed to use ChatGPT or any other generative AI tool to help create your own measurement tool. In your programs, give credit to the tool(s) you used to create it.

In your report: Explain which lines in your code **(a) Create Datagram Sockets, (b) Send/write data to the socket, c) read data from the socket, (d) measure the time of arrival of packets**. You can cut and paste each relevant line from both programs into your report. **Also describe each function used in these particular lines of code and describe what they take as arguments**. In addition, **(e) explain how you know that your sender.c program sends two packets out one after the other,** with no gap between them.

Part 3: Experimentation

Part-1 neglected that there can be other packets in the network besides the packets generated by the probing tool. In practice, these other packets may end up before, or in between, the probe packet pairs in various queues along the path, and hence the final spacing at the receiver may not equal P/C. In addition, delays can be introduced due to MAC layer protocols, etc. Hence the values printed at receiver.c to the text file may not be constant, and may not equal C.

Experiment-1 Run both sender.c and receiver.c on the same machine. Restrict the loopback interface on the machine to have maximum throughput of 10Mbps using the **“sudo tc”** command (should work on Linux and Mac; for Windows you will have to find an equivalent). You may encounter a “burst size”, which you can set to a little bigger than the packet size you are using (e.g. if your packet size is 8kbits, then set burst to 9 kbits). You can take GPT’s help here too to find the right “tc” command. Send at least 100 pairs of probe packets for this experiment.

In your report: Plot a histogram of estimates of “C”. The suggested bin size for the histogram is 1 Mbps, but you can use some other bin size as well. Describe what you observe. Can someone looking at the histogram easily see the bottleneck link speed as equalling 10 Mbps?

Experiment-2 Run sender.c and receiver.c on different machines. Have no restrictions on throughput for any interface on either machine (i.e. remove any rate limit you had put on the

default interface at the sender in any earlier experiment). Place the sender and receiver at different locations on IITB campus. Use “traceroute” to ensure that there are at least 4 hops between sender and receiver (at least 5 lines should be printed by traceroute).

In your report: Give the traceroute output for each of the 3 paths. For each path, plot a histogram of estimates of “C”. Suggested bin size for the histogram is 10 Mbps, but you can use some other bin size as well. Describe what you observe. Can someone looking at the histogram easily see the bottleneck link speed? Comment on what you think is the bottleneck link speed, based on the observations.

Note: Run experiment-2 for at least 3 different paths on campus. You can do this by shifting the location of the receiver and/or the sender.

What to submit on Moodle: Your report should be a pdf file named <rollno>-lab5-report.pdf (or <rollno1>_<rollno2>-lab5-report.pdf if you are doing it in a group). Submit a file <rollno>-lab5.tgz (or <rollno1>_<rollno2>-lab5.tgz if doing in a group) which includes the report, sender.c, receiver.c, and a README file explaining how to compile and run the code).