CSE 473S – Introduction to Computer Networks
Fall 2009

Assignment 2

Introduction

This lab assignment explores the behavior of the XSTCP protocol under different conditions using the ONL NPR testbed. XSTCP is an eXtremely Simple Transmission Control Protocol that is described below.

If you require clarification, send email to kenw@arl.wustl.edu AND cse473s@gmail.com clearly stating your question. There will be an FAQ maintained at:

http://www.arl.wustl.edu/~kenw/courses/cse473s/current/FAQ.html.

The XSTCP Protocol

The XSTCP protocol is an extremely simple protocol that transmits fixed-length packets. The *xsnd* code sends packets using UDP where the front of the UDP packet contains an XSTCP header containing a packet sequence number starting at 0 and a timestamp. The *xrcv* code uses an *in-order accept* policy; That is, it has a *nxtsnum* variable that is initialized to 0. If the first packet has a sequence number of 0, it sends an ACK packet with a sequence number of *nxtsnum* and increments *nxtsnum*. Thus, *nxtsnum* indicates the sequence number it expects next. If it receives an unexpected sequence number, it drops the packet silently and goes back to waiting for the next expected packet sequence number.

The sender uses a *tripling slow-start* algorithm as long as it gets back ACK packets from the receiver. That is, it sends out packet 0 and waits for an ACK. Every time it gets an ACK, it sends out the next three packets. After the first packet drop, the sender will timeout waiting for an ACK, and it will begin its *tripling slow-start* algorithm again by sending only one packet (the smallest numbered packet that had not been ACKed), waiting for an ACK, and then sending out the next three packets, and so forth, thus starting another send-timeout cycle.

Both *xsnd* and *xrcv* have optional flags that are documented in the README file. For example, both have a *-v* flag (verbose) that causes them to output additional information about packet sending and receiving.

Here are some other details that are peculiar to the implementation that you should know about.

- The default timeout used in *xsnd* is 400 msec which can be changed using the "*-t T*" command-line option where T is the number of milliseconds.
- The *xsnd* program sends fake packets that have the correct headers but packet bodies that contain zero-bytes.
- The *xsnd* program sends out three *ping* packets to ensure ARP tables are initialized before beginning packet transmission. And it sends three FIN packets when it is done. These FIN packets consist of negative sequence numbers (actually the negative of the number of packets sent).

Overview

The assignment has three parts:

A) Expected behavior of a Single XSTCP Flow

B) Actual behavior of a Single XSTCP Flow

C) Actual behavior of Multiple XSTCP Flows

Parts A, B and C lead you through the conduct of experiments. However, we ask that you explain why the results make sense. Following these three parts is a section that describes what and when to submit.

In the description of the exercises to follow, questions for which you **MUST** include in your report are preceded by the **Question** label.

Reading

From Lab 1, you should already understand the basic material in the *NPR Tutorial* section labeled **The Remote Laboratory Interface**. This lab uses additional features that are listed below. Pay particular attention to the ones marked with ***** since they discuss features that you will need to use in this lab. The others discuss features that are used by the configuration file *lab2-f09.exp* and are a prerequisite for understanding the marked pages.

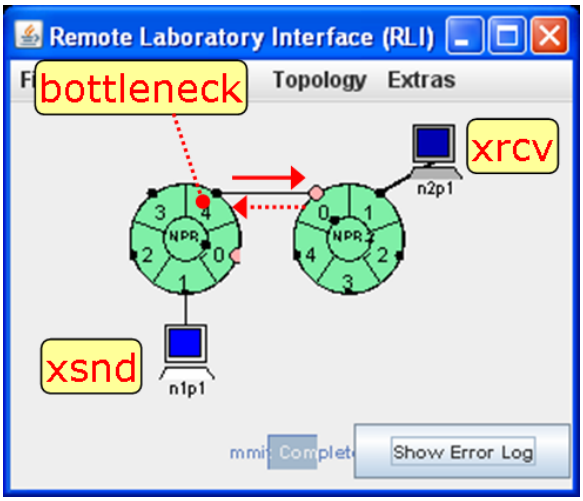
- **Filters, Queues and Bandwidth** ⇒
  - Basic Packet Filtering
  - Setting Port Rate
  - Generating Traffic With Iperf
  - ***** Mapping Flows to Queues
  - Monitoring Queue Length and Packet Drops
  - ***** Using Iperf With TCP
- **Router Plugins** ⇒
  - Plugin Basics
  - ***** Installing a Plugin
  - ***** Sending a Message to a Plugin
- **Examples** ⇒
  - Filters, Queues and Bandwidth
  - TCP With Delay

    The important part of this page is Example B which illustrates how to use the *delay* plugin.

In addition, you should be able to write a simple shell script so you can modify the scripts provided i.e., run-lab2-sndrs and run-lab2-rcvrs.

Part A: Expected Behavior of a Single XSTCP Flow

This part of the lab assignment explores the behavior of a single XSTCP flow. In Parts A and B, you should use the configuration file *lab2-f09.exp* described below. It will create the dumbbell configuration shown below with a 12 Mbps bottleneck at output port 1.4 in which there will be one host (n1p1) acting as an XSTCP sender and one host (n2p1) acting as an XSTCP receiver. The traffic from n1p1 will be directed to queue 64 at output port 1.4. Queue 64 will have a threshhold of 1,200,000 bytes (i.e., it can hold 1,200,000 bytes). Packets from n1p1 will go through a 50 msec delay plugin at NPR1 before being queued in queue 64 at port 1.4. ACK traffic from n2p1 will go through a 50 msec delay plugin at NPR 2 before going to queue 64 at port 2.0. This means that there are two filters. One is at input port 1.1 that directs packets to the NPR 1 delay plugin and then on to output port 1.4. The other is at input port 2.1 that directs packets to the NPR 2 delay plugin and then on to output port 2.0.



- Get the configuration file *lab2-f09.exp*.

  The file can be found at the following URL:

  http://www.cse.wustl.edu/~jain/cse473-09/ftp/lab2-f09.exp.

- Copy the XSTCP sender and receiver code and the README file to your ONL directory.
  - The files are at ~kenw/src/xstcp/{xsnd,xrcv,README}.
  - The command-line flags are described in the source code and summarized in the README file.
- Get a one-hour reservation for one NPR pair and six (6) hosts.

  If you only plan to do this part of the assignment, you only need two (2) hosts. All other parts will require six (6) hosts.

- Commit the configuration file.

  You should see three (3) charts: bandwidth, queue length, and drops (packet).

- Verify that traffic can flow between n1p1 and n2p1 by using *ping*.

  A.1: Report the path capacity in both forward and reverse directions. Set the bottleneck link capacity to 12 Mbps for forward direction (from n1p1 to n2p1) Report what RTT do you observe? Submit screen shots of the output of the ping command both sender and receiver as well as the bandwidth chart during the transmission. Note that all bandwidth chart in lab2 assignment must be rescaled to Mbps (Y-scale).

- Verify that you can successfully send 16 1000-byte XSTCP packets by running the sender on n1p1 and the receiver on n2p1 in verbose (-v) mode.

  A.2: Submit the output from the sender and receiver (sending 16 1000-byte XSTCP packets in verbose mode).
  Submit screen shots of bandwidth, queue length, and packet drop charts during the transmission.

  > There are several ways that you can do this: 1) You can redirect stdout and stderr to files and submit the files; or 2) You can make a snapshot of the sender and receiver windows.

- Theoretically determine when the first packet drop will occur if you attempt to send 8,000 1000-byte XSTCP packets from n1p1 to n2p1.

  A.3: Submit a derivation for the sequence number of the first packet drop (consider bottleneck link, queue size, RTT, duration of packet transfer, etc.). Assume that sequence numbers begin with 0 and are incremented for each new packet.

Part B: Actual Behavior of a Single XSTCP Flow

- Experimentally determine when the first packet drop occurs if you attempt to send 8,000 1000-byte XSTCP packets from n1p1 to n2p1, and compare measurements with theory (based on your derivation from Question A4).
  - Here are the receiver and sender call-lines:

    ```
    $n2p1> xrcv
    $n1p1> xsnd -r n2p1 -n 2000
    ```

  - Note that the above does not use the -v flag to avoid excessive stdout overhead.
  - **Note:** *xsnd* sends three (3) *ping* packets to the recevier host to make sure that ARP entries are installed along the packet path. Then, it goes into its packet transfer mode. In addition, in case you can't ssh to *nXnX* , rerun the configuration or you can ssh to *onlXXX* instead (Check the mapping by right click at the host)

B.1:

a) What is the sequence number of the first packet drop reported by *xrcv*?

b) Discuss how much the measurement matches or doesn't match your derivation, and explain why you think there are differences.

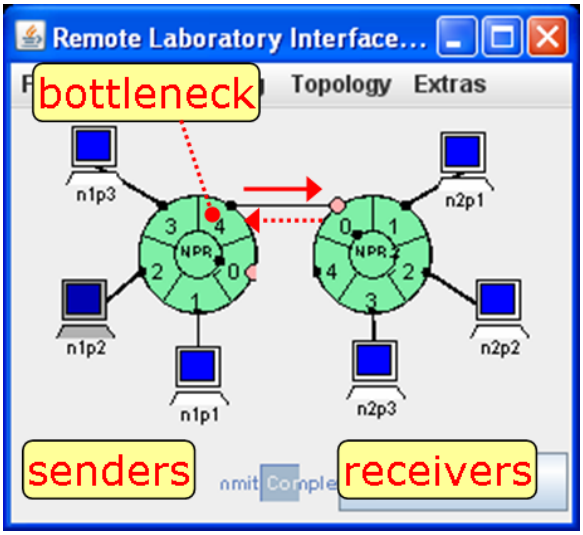
c) Submit screenshots of the bandwidth chart and the queue length chart.

d) Explain in what respects the charts match/ do NOT match your expectations based on theory.

Part C: Actual Behavior of Multiple XSTCP Flows

This part of the lab assignment explores the behavior of multiple (3), *concurrent* XSTCP flows where host n1pY sends packets to n2pY where Y is 1, 2 or 3. There are no changes to the bottleneck port. You will need to add four hosts, four links and four (4) filters. The filters should direct the packets to a *delay* plugin. Now, there will be three (3) XSTCP flows contending for output port 1.4 and packets from these flows will contend for transmission from queue 64 at output port 1.4.

Each of the senders will start at the same time and each will send 8,000 1,000-byte packets.



- Copy the two shell scripts *run-lab2-rcvrs* and *run-lab2-sndrs* into your bin directory (or where ever you put your executables for this lab).

  The two scripts are on the onlusr host in the directory ~kenw/bin/. You should look at the two scripts to make sure you understand what they are doing.

- Starting with the *lab2-f09.exp* configuration file, add four more hosts and links so that there are hosts nXpY where X is 1 or 2 and Y is 1, 2 or 3 (i.e., there will be hosts attached to ports 1.1, 1.2, 1.3, 2.1, 2.2 and 2.3).

- Add filters at input ports 1.2 and 1.3 so that any incoming packets will go to queue 64 at output port 1.4.

  The filters at input ports 1.2 and 1.3 should be identical to the one at input port 1.1, and the filters at input ports 2.2 and 2.3 should be identical to the one at input port 2.1.
  (Hint: **Port Y** ⇒ **Configuration** ⇒ **Filter Table**)

- Add filters at input ports 2.2 and 2.3 so that any incoming packets will go to the *delay plugin* on NPR 2.

- Verify that traffic can flow between the two new pairs of hosts using *ping*.

- Add four plots to the *bandwidth* chart that show the following four bandwidths: 1) into port 1.2; 2) into port 1.3; 3) out of port 2.2; and 4) out of port 2.3.

- Determine how long it takes to send 8,000 1000-byte XSTCP packets from each of the three senders.

  – You should be able to start all of the receivers by running the *run-lab2-rcvrs* script from the onlusr host. But if you are troubleshooting something, you may need to SSH to a particular host and enter some of the commands in the script. The receivers should automatically reinitialize themselves once they have started and do not need to be restarted unless something terrible happens.

– You need to start the receivers FIRST. Then, you should be able to start all of the senders by running the *run-lab2-sndrs* script from the onlusr host. If you want to repeat an experiment, all you have to do is run the *run-lab2-sndrs* script again.

C.1:

a) Submit screenshots of the output (n1p1 to n2p1, n1p2 to n2p2, and n1p3 to n2p3) from the ping command before and after adding filters to delay the packet. Show all three transmission paths in both forward (packet) and reverse (ack) directions

a) Submit screenshots of the bandwidth chart, packet drop, and the queue length chart.

b) Explain in what respects the charts match your expectations. Include a discussion of: how the charts are consistent with each other; and

c) Explain in what respects the charts do NOT match your expectations based on theory.

What to Submit

Prepare a report covering tasks: A, B, and C and also answer the questions in each section. A hard copy of the report must be submitted to the instructor on the following Monday before the class starts.