###### *CSE 473 – Introduction to Computer Networks*

Studio 4

##### (*Adapted from Jon Turner’s Studios*)

In this studio, you will be building a program that monitors a running application and prints out a short report when the application terminates. The Monitor is implemented as a Java module with its own thread and it operates between two other modules, an *application module* (SrcSnk) that generates packet payloads and receives payloads sent by a remote peer, and a *substrate module* (Substrate) that handles socket IO in conjunction with *sender* (Sender) and *receiver* (Receiver) modules.

In your repository, you will find an ONL configuration file and several Java files, SrcSnk.java, Monitor.java, Substrate.java, TestMonitor.java and Packet.java. We won’t be using *onl* for the first part of the studio. In the first part of the studio, you should ideally be working with a partner next to you.

1. Review the source code with your partner. Make sure you understand what all the parts are doing and how they fit together. Make a diagram that shows a client and server instance of the program and shows how they communicate. Your diagram should include blocks for the SrcSnk, Monitor, Substrate, SenderandReceiver modules.

**Note**: The *studio4Notes\_2018.docx* document includes such a diagram and is provided to make sure that you don’t get stuck with an incorrect understanding of the system’s structure. However, it is in your best interest **NOT** to look at it until you have completed your own analysis of the code.

1. Next, fill in the missing parts of the Monitor. Your finished program should keep track of the number of packets and chars sent and received. It should include a copy of the number of packets and chars sent so far in each packet sent. At the end, it should print a short report summarizing the number sent, the number received, and the number that were lost in transit.
2. Test your program running on a single computer. Both of you should login to the computer where you’re going to do your testing. One should start the server, and the other start the client. Try different values for the command line arguments and observe the results. Fix any problems you discover in your program. While doing this testing, choose a server port number that is not being used. You can use netstat to check if a given port number is in use. After you are satisfied that things are working correctly, run the provided script0 as well (note that script0 expects the server port number as a command line argument).
3. In this part you will be testing your program in ONL. Each partner should create a directory to work in. Name these directories studio4a and studio4b. Take turns with your partner to carry out this testing. You should test both versions of the Monitor that you and your partner wrote. Run the provided script1 to verify that things are working correctly in this environment (script1 is similar to script0). You will need to edit script1 to use the proper directory.
4. Once both versions of the program are working correctly, you should test them together. That is, you should use your program as the server, and the program produced by your partner as the client, and vice-versa. You will have to modify script1 to allow you to carry out this test. Correct any problems you run into. Make sure that the pair of programs works the same way no matter which of the two is the server and which is the client.
5. Start an iperf udp server on h6x1 using the provided udpRcvrscript. Then run script2 (after editing the directory names) with an argument of 15 (script2 runs TestMonitor and iperf simultaneously and this argument is the iperf client’s sending rate). Notice what happens on the monitoring displays. What percentage of the TestMonitor traffic sent gets lost when running script2? What percentage of the iperftraffic is lost? Are they the same or different? Any idea why? Adjust the iperf sending rate until there is no queueing on the inter-router link. What is the largest iperfsending rate that you can sustain without any loss? If you increase the iperf sending rate to 30, what percentage of the TestMonitortraffic is lost? What percentage of the iperf traffic is lost?