CSE 473 – Introduction to Computer Networks

Studio 2

(*Adapted from Jon Turner's Studios*)

Reminder. The purpose of the studio sessions is to help you get a better understanding of the material, and to help you prepare for the labs. Studios are not graded and there is nothing to turn in, but the more you learn during studio, the better prepared you will be to tackle the labs.

The purpose of this studio is to familiarize you with ONL. Before coming to studio, read the tutorial material available on the ONL web page. This will help you proceed much more quickly when you come to studio.

Before you come to studio, the resources you need for your session will be reserved for you using one of six studio accounts *cse473a*, *cse473b*,..., *cse473f*. Each group will be told which account to use, and you will be provided with the studio account password. These accounts should be used only during studio sessions. You will have your own personal account that you should use for the lab assignments. Your svn repository includes a studio2 directory that contains an ONL configuration file called cse473-studio2.onl, which you should use for this studio.

To get started, the first step is to create an *ssh* connection to ONL that is configured with a tunnel. The web site describes a variety of ways to do this, but for this studio, the simplest approach is to first open a command prompt window and type

ssh -L 7070:onlsrv:7070 cse473X@onl.wustl.edu

where the X should be replaced by *a*, *b*,... *f*, depending on which studio account you have been assigned. When you are prompted for a password, enter the password you have been given. At this point, you should start the RLI (on the Urbauer 214 computers, you will find it in the Start menu, under Programming) and open the provided configuration file (*cse473-studio2.onl*). Then, select *commit* from the File menu. You will be prompted to login using your studio account and password (sorry for the double login, but it is required). If all works correctly, your experimental network will change color and the dashed links will turn solid. If this does not work, first make sure that you have followed all instructions exactly, and then request help if necessary.

At this point, you can also check out a copy of your *bitbucket* repository on *onl*. You'll find that having a copy of your repo on *onl* makes things more convenient. Alternatively, you can open an SFTP graphical client on the Windows machine and use it to transfer files to the *onl* server and access files stored there. This will allow you to use standard Windows applications for editing and/or compiling programs, which you may prefer.

Once you've gotten to this point, you should proceed through the following exercises with your group.

1. Open *ssh* connections to two of the computers in your ONL configuration, specifically the computers labeled *h4x2* and *h7x1* (find these in the RLI window and note that they are on different routers). You can use the SSH client on the Urbauer 214 computers for this purpose. You will be connecting to *onl.wustl.edu* and using your provided studio account. Make two connections to the main onl server, and then in the first window, type

   source /users/onl/.topology
   ssh $h4x2

The first command defines a set of shell variables (including $h4x2) which are used to login to the computers in your *onl* network. The second connects you to the specified computer. In the second window, type

source /users/onl/.topology
ssh $h7x1

Keep these windows open throughout the remainder of the studio. We will refer to these as the *h4x2* window and the *h7x1* window.

2. In the *h4x2* window, type

   ping h7x1

   (note that there is no $ sign in front of *h7x1* in this case). Observe what happens in the monitoring display window. Explain what you see. To terminate the ping, type CTRL-C. Try pinging other computers in your network and observe how the monitoring display window changes.

   Now, in the *h4x2* window, type

   /sbin/ifconfig

   and record the packet counts for interface *data0*. Now run one of the ping commands again and then re-run *ifconfig*. Note how the packet counts change. Make sure that the numbers are consistent with what you observe on the monitoring display.

3. In this part, you will be using *Wireshark* to observe the packets being sent between the computers in your *onl* network. Using *Wireshark* in *onl* requires a little extra effort, since *Wireshark* itself must run on the target computer within *onl*, while the graphical interface needs to appear on your local computer. Start by opening a new command prompt window on your local computer and type

   startxwin

   This causes another window to open. In this new window, type

   ssh –X myLogin@onl.wustl.edu

   This creates an *ssh* connection that forwards "X-windows" commands from *onlusr* back to your local computer. X-windows is a generic windowing system developed at MIT in the 1980s. It is still used for a number of applications, including *Wireshark*. Note, that you do *not* need the tunnel specification in this window (-L 7070:onlsrv:7070), since original *ssh* connection already provides the tunnel. Now, in the new window, type

   source /users/onl/.topology
   ssh –X $h4x2

   This will log you into host *h4x2* and forward X-windows commands from *h4x2* back through *onlusr* to your local computer. Next, type

   sudo wireshark

   After you enter your password, *Wireshark* will start running on *h4x2*, and the *Wireshark* window will open on your local computer. Configure *Wireshark* to capture packets on the *data0* interface and then re-run *remoteScript* in the original terminal window connected to *h4x2*.

   Now, in your original *h4x2* window, run *ping*. Terminate the *ping* command after a few packets. Observe the packets in the *Wireshark* window. Make sure you understand what you are seeing. In particular, notice the delay reported by the ping command. Try to figure out where this delay is coming from.

4. In this part, you will run a provided TCP echo server, similar to the UDP echo server from studio 1. You will find the Java source code in your studio 2 folder. Review the code to make sure you understand what both the client and server do. Transfer the code for both the server and the client to your *onl* studio account. Then, in the *h7x1* window, compile the server by typing

   javac TcpEchoServer.java

   then, run the server in the background by typing

   java TcpEchoServer h7x1 30123 &

   Before starting the client, type the following command in the h7x1 window

   netstat -an | grep 30123

   the *netstat* command (see http://en.wikipedia.org/wiki/Netstat for information on *netstat* and its syntax) produces a list of active sockets and the *grep* command filters out all lines except those containing the string 30123. After typing the command, you should get a line describing a TCP socket in the listening state.

   Now, run the client in the *h4x2* window, using the command

   java TcpEchoClient h7x1 30123

   and run the *netstat* command in the *h7x1* window again. Observe what has changed, and make sure you understand why. Type a few lines into the client and verify that the echoed responses are as expected, then enter a blank line to terminate the program. Re-run the *netstat* command at the end of this sequence.

5. In this part, you will be using the *iperf* utility to send traffic through your *onl* network (see https://iperf.fr for a short description of *iperf* and its syntax). To simplify your use of *iperf*, we have provided a couple shell scripts. Before running the scripts logout of the *h4x2* window, by typing CTRL-D. This will bring you back to the main *onl* server (*onlusr*) and from here you can run the scripts you will be using.

   The scripts can again be found in your studio2 folder (they are called *udpSenders*, *udpRcvrs*, *tcpSenders*, *tcpRcvrs* and *mix*) and also need to be transferred to your *onl* studio account. Review the two *udp* scripts and make sure you understand what they are doing. Ask about anything you're not sure of. Run *udpRcvrs* and then run *udpSenders*. Observe the traffic in the monitoring windows. How much traffic is being sent by each sender? How much is being received by each receiver? Explain any discrepancies you observe. Terminate *udpSenders* by typing CTRL-C.

   Now, run *tcpRcvrs*, followed by *tcpSenders*. Compare the results in this case to the results in the previous case. Explain why they are different.

   Now, run the *mix* script. What do you observe? Explain why these results are different from the previous two.