#### **DO NOT READ UNTIL AFTER YOU HAVE GIVEN AN EARNEST ATTEMPT AT COMPLETING THE STUDIO**

*CSE 473 – Introduction to Computer Networks*

Studio 4 Notes

There are five buffers in Studio 4 that we need to worry about:

* Monitor’s fromSrc buffer,
* Sender’s sendq,
* the socket buffer,
* Receiver’s rcvq,
* Monitor’s toSnk.

To understand why and how those buffers can affect the behavior of a given experiment, it is important to step through the logic of each component on which the studio relies. After starting the server but before starting the client, the server’s SrcSnk program immediately starts sending messages to the fromSrc buffer, which will then deposit packets into sendq. The code checks that there is room in the buffers so that they won’t overflow, but those packets will be held there until the client starts (and in the process provides the server with the IP address of where to send the packets). Both buffers can store 1000 packets, so up to a total of 2000 packets can accumulate across both buffers and be ready to be sent the instant the client is started. Once the client starts, the server’s receiver program retrieves the IP address and port from the first packet received from the client and passes it to the sender program that immediately starts transmitting to the client all the packets it has stored in its buffers. The socket buffer has a finite amount of storage, which depending on its size and the number of packets stored in the sender’s buffers could result in many packets being lost (remember, this is a UDP and not a TCP socket, and so there is no guarantee and if the socket buffer is full, new packets are simply dropped). The default UDP configuration on ONL machines uses a socket buffer size of 212,992 bytes and allocates at least 256 bytes of memory for each packet, *i.e.,* a packet smaller than 256 bytes still takes 256 bytes. Assuming 256 bytes packets, this translates into a total of 21,992/256 = 832 packets, which is smaller than our maximum burst size of 2,000 packets. So if there is enough delay between the server and client start times so that more than 832 packets have been stored across the two buffers at the sender, we would be losing all those “excess” packets in the initial sender burst. In order to avoid this, we need to increase the UDP socket buffer size. In the current configuration, it should have been set to 3,407,872 bytes, which should eliminate all possibilities of initial losses (you can check that this is indeed the case by typing cat /proc/sys/net/core/rmem\_default at a command line prompt).

The TestMonitor is used to handle all the arguments for running the programs and to start up the SourceSink, Monitor, and Substrate. Each one of these programs runs on its own thread, so the TestMonitor is used to tie them all together. The Substrate is used only to declare the Sender and the Receiver, which deal with socket IO. As with the TestMonitor, the Substrate just keeps the Sender and Receiver tied together.

*Program Diagram*: The diagram below shows the flow of data from SrcSnk in one host to SrcSnk in another host. Arrows indicate the direction of data transfer. Approximate equivalence to layers in the standard TCP/IP stack is shown on the left.

**Test Monitor**

Application Layer

Transport Layer

**SrcSnk**

toSnk

fromSrc

**Monitor**

**Substrate**

Link Layer

**Receiver**

**Sender**

rcvq

sendq

sendq

rcvq

**Sender**

**Receiver**

Link Layer

**Substrate**

Transport Layer

toSnk

fromSrc

**Monitor**

**Test Monitor**

**SrcSnk**

Application Layer