COM SCI 118: COMPUTER NETWORK FUNDAMENTALS

*Project 2: Simple Window-based Reliable Data Transfer*

Yash Choudhary, 704630134

Sparsh Arora, 804653078

**Introduction**

In this project our purpose was to extend our understanding of sockets and TCP/ reliable data transfer to implement a TCP over a UDP socket using the Selective Repeat (SR) protocol.

**High Level server design description**

**Packet Class:**

We came up with a packet class with appropriate header variables and a data section. These are used by both sides of the socket to send, receive and buffer packets. The packet has the following structure in pseudo code:

***Packet:***

Header: (10 bytes)

Sequence number (2 bytes)

Acknowledgement number (2 bytes)

SYN flag (2 bytes)

ACK flag (2 bytes)

FIN flag (2 bytes)

Data Section: (1014 bytes)

**Client Side:**

The client initiates connection with the server by sending an initial SYN packet that also includes the filename in the data section of the packet. Once the handshake is complete, the client side waits for the sender to start sending packets of data from the file. The client then uses SR (through a 5- circular array) to buffer out of order packets and then writes them back together, sending ACKs appropriately. The circular array is used to store packets by placing them in an index given by index = packetNum % 5 and then writing to file whenever the head pointer pointed to the expected packet. In this case, client writes output to file till head pointer encounters an empty space in the array (NULL).

**Server Side:**

The server gets the SYN from the client and then confirms the handshake. If the file exists in the server, it starts to divide it up, create packets by attaching headers through helper functions in the packet class and sending them through the socket to the client. The Server also ensures Reliable data transfer by using local timers to keep track of the RTO of each packet and retransmits. The server also terminates sending packets by sending a FIN packet when all ACKs have been received. Like the client, the server also keeps track of a window by tracking the first and last byte number and then incrementing them once an ACK in the window or a group of ACKs from the window are received.

**2. Difficulties faced**

One of the major difficulties we faced was in coming up with standards of the packet. We had to decide our header lengths carefully so that we had enough space to include relevant information. We also had to make sure that the header wasn’t too big, so we could send more data with each packet.

Another difficulty we faced was with making sure that the window size movement considered the maximum sequence number and we if the packet was within the new window accurately.

**3. Compiling and Testing**

To compile, simply use the command:

**tar -xzvf 804653078.tar.gz**

Then, to compile the code use,

**make**

And then to run the code:

**./server <port\_num>**

**./client <host> <porn\_num> <filename>**

**4. Sample Outputs**

Given below are some outputs for sample case of a transmission of a file (~40,000 B).

Client Side

Server Side

 