**Programming Assignment 2**

**Name: Yuxin Zhang ID#: 50167183**

1. **I have read and understood the course academic integrity policy.**
2. Brief description of the timeout scheme you used in your protocol implementation and why you chose that scheme:

**ABT**: Since no buffer is on the sender side and only one packet is in the channel at any time, my ABT protocol retransmits the latest packet that was sent into the channel and restart timer. Since one-way network delay in the channel is average five time units, the average ARR is two times five time units, which is 10 time units. Considering the other unexpected influence in channel that may cause a slight change of transmission delay, my ABT protocol set the timeout time to 11.5 time units, which is slightly larger than 10, to be more flexible and prevent the unnecessary retransmission.

**GBN**: When timer times out, my GBN protocol retransmits the packets that had sent out but not ACKed yet, in the order from small sequence number to large sequence number. In my protocol, sequence number ranges from 1 to 65535, so all the sequence number of packets are different. The smallest sequence number of packet that was send not ACKed marked by send-base, that only changes when sender successfully gets an acknowledgment from receiver. The largest sequence number marked by sending sequence number that only changes when a packet with certain sequence number was first sent. After sending the last packet, start timer again. Retransmitting the packets in a fixed order makes sure that when there is no corruption and no loss in channel, the sender can receive all the packets. The timeout time I choose is based on the window size.

Time out time = 14.5 + (Window Size/100)

Though the average RTT is 10 time units, there will be more packets in the channel at the same time when GBN works. This cause more delay when transmitting. The larger the window size, the more packets will be in the channel at the same time. So my timer changes with the window size. After many tests, I roughly get this equation between window size and timeout time, that allows enough retransmission to make sure that the sender will immediately retransmit the packets after lost and also prevent too much retransmission from sender when the packets are not lost but has a longer delay.

SR: In this protocol, every single packets has its own timer. When it times out, the sender will send this packets out and restart this timer. The timeout time I choose is changed based on window size.

Timeout Time = 13.5 + (Window Size/200)

The way I choose timeout time is in the same way as GBN protocol. In SR, only when the packet’s time runs out this packet will be retransmitted. There are less packets in the channel than GBN works at the same time, even when the window size is large. So I use a smaller timeout time here.

1. Brief description (including references to the corresponding variables and data structures in your code) of how you implemented multiple software timers in SR using a single hardware timer.

In SR, I use a structure called *sendpktinfo* to store message on the sender side.

*struct sendpktinfo{*

*char data[20];*

*int flag;*

*float timer;}*

data is an array of char in the size of 20 to store message data. Flag is an integer that represents different states of this message. Flag equals 0 means that this message has not been sent out; flag equals 1 means that this message has been sent out and timer is activated; flag equals 2 means that this message has been successfully ACKed and complete its transmission. Timer is a float that remembers the time of this message. Timer only changes when this message’s flag equals 1 and when it is less than or equal to 0, this message will be retransmitted and the timer will be set to timeout time again.

To store the message into the buffer on the sender side, I create an array of structure sendpktinfo in the size of 1100, called *Amsgbuffer*.

*struct sendpktinfo Amsgbuffer[1100];*

For every message delivered from layer 5, I store them into Amsgbuffer and use the index of this array to represent its sequence number.

*int Amsgcount = 1;*

Also, the integer Amsgcount remembers the largest index of array Amsgbuffer[1100] that has data. In another way, Amsgcount-1 shows how many message stores in the buffer, for the Amsgcout starts from 1.

I use integer Amsgsend to remember the largest sequence number that was sent out. And use interger Amsgrecv to remember the largest sequence number that was successfully ACKed.

*int Amsgsend = 1;*

*int Amsgrecv = 1;*

the packets with sequence numbers between Amsgsend and Amsgrecv is all the packet that is transmitting in the channel.

In SR, I set the hardware timer’s timeout time to one time unit, which means *A\_timerinterrupt()* function will be run every one time unit. This function checks all the packets from Amsgsend to Amsgrecv, which are active, to decrement its timer value by one, and then check its timer value. If the timer value less than or equal to 0, retransmit this packet.