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.

3. 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.

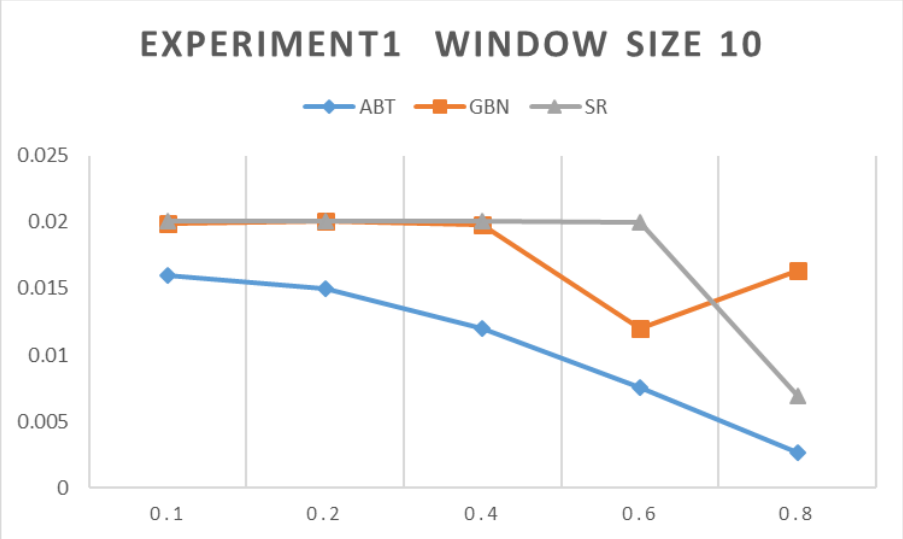
int Amsgsend = 1;
int Amsgrecv = 1;

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. If the flag is either 0 or 2, ignore it. If the flag is 1, which means this packet is active, then decrement its timer value by one, and then check its timer value. If the timer value is less than or equal to 0, retransmit this packet.

Performance Comparison

Experiment 1

Window size 10

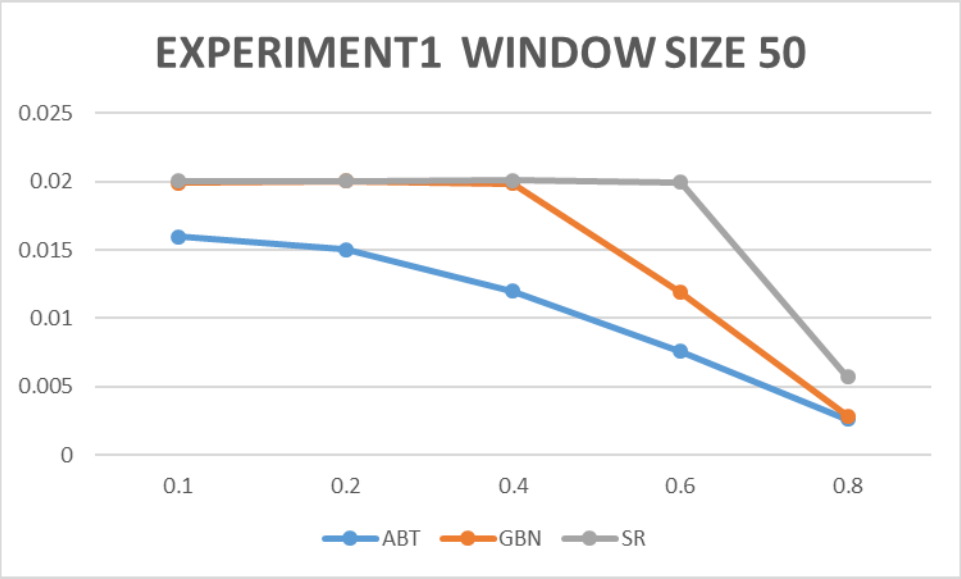


The throughput of ABT is lowest among the three, and decreases as the loss rate goes up, the same as I expected. This because ABT only transmits one packet at a time and has no buffer to store message. It will waste a lot of time on waiting the ACK. When the loss rate rises up, it retransmits current packet to complete transmission. However, since the loss rate is higher and higher, when simulator stops, the successful received packets are less and less.

The throughput of GBN keeps high when loss rate is below 0.4. In this case, GBN successfully transmits almost all packets to receiver. However, when the loss rate is greater than 0.4, the throughput goes down. My expectation is the through put will goes down when loss rise up as it is in ABT. But when loss rate reaches 0.8, the throughput rise up and is higher than the throughput when loss rate is 0.6 but still lower than the highest value. This may because that when loss rate is 0.6, the receiver side touches more packets than the case that loss rate is 0.8. It will keep busy in checking the checksum and check the sequence number to decide whether or not to deliver it to layer 5. This result in when the loss rate is 0.6, the receiver side wastes a lot of time to do the check and when simulator stops, it handles less packets than the loss rate is 0.8.

The throughput of SR keeps high before loss rate reaches 0.8, that successfully delivers almost all messages to receiver. However, when loss rate reaches 0.8, the throughput dramatically drops to 0.006917, which is lower than GBN but still higher than ABT. This may because that SR assigns to each packet an individual timer, which cannot make sure enough retransmission times when loss is high. What's more, the window only moves when it gets the expected sequence number. When this expected sequence number is lost, it will wait until another time out occur to get a new one. But in GBN, when times out, it will retransmit all packets that already in the channel, which makes sure enough retransmission times. These reasons result in that the throughput of AR at loss rate is 0.8 is lower than the throughput of GBN. Another point worth to mention is that when the loss rate is 0.6, the throughput of SR is higher than GBN. This is because that in SR, the receiver side has buffer to store ACKed Packet. Once This packet is successfully arrived to receiver is side, it will not be retransmitted again. But in GBN, the receiver side only accepts the packets that arrived in order. Otherwise, this packet need to be retransmitted again. This result in that in this case, SR performs better than GBN.

Window size 50



When the window size rise to 50, the throughput of ABT still is the lowest one.

The throughput of GBN remains the same before loss rate reaches 0.6 as window size is 10. But when loss rate reaches 0.6, the throughput start to drop. When loss rate reaches 0.8, the throughput drops again, which is different from the one of window size 10. This may because that in this case, the time wastes more on retransmission rather than checking. Here the window size is 50, 5 times of 10. In GBN, one retransmission will handle 50 packet in most of the time, which costs a lot of time. The data shows below also explains this.

Window Size	Loss	Transport Packets	Loss	Transport Packets
10	0.6	21253	0.8	43488
50	0.6	135154	0.8	202015

Figure 1The Numbers of Transport Packets of GBN

The number of packets that sender side sends when window size is 50 is much higher than the number when the window size is 10. This results in that when simulator stops, the sender side wastes too much time in retransmission to transmit more packets.

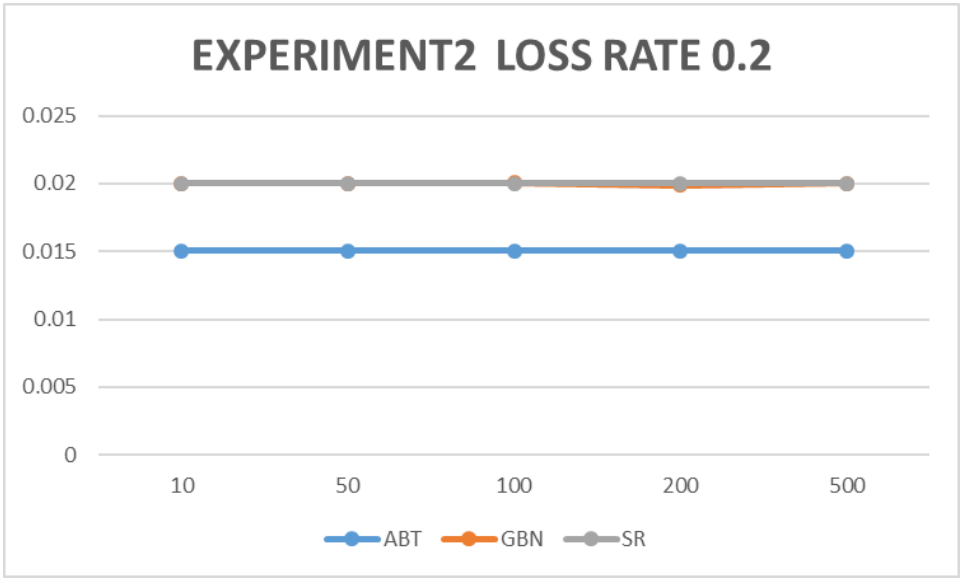
As the window size is 10, SR keeps high throughput when the loss rate is less than 0.8. But when loss rate reaches 0.8, the throughput drops dramatically. However, this time the throughput is a little bit less than the window size is 10. This because that now it has more timers and more retransmissions to handle, which costs more time. The number of packets that sender sends out also explains this point.

Window Size	Loss	Transport Packets
10	0.8	13320
50	0.8	103354

Figure 2 The Number of Transport Packets of SR

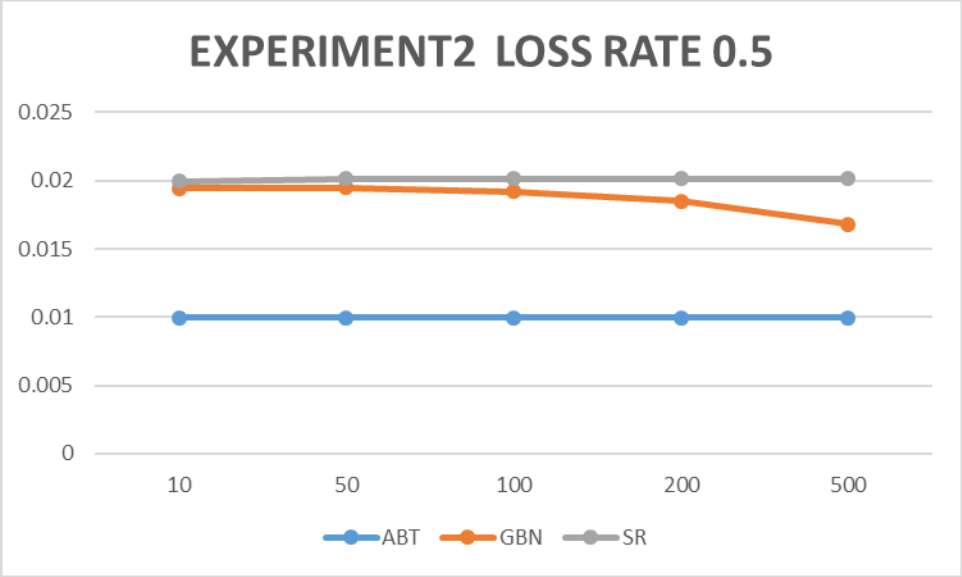
Experiment 2

Loss Probabilities 0.2



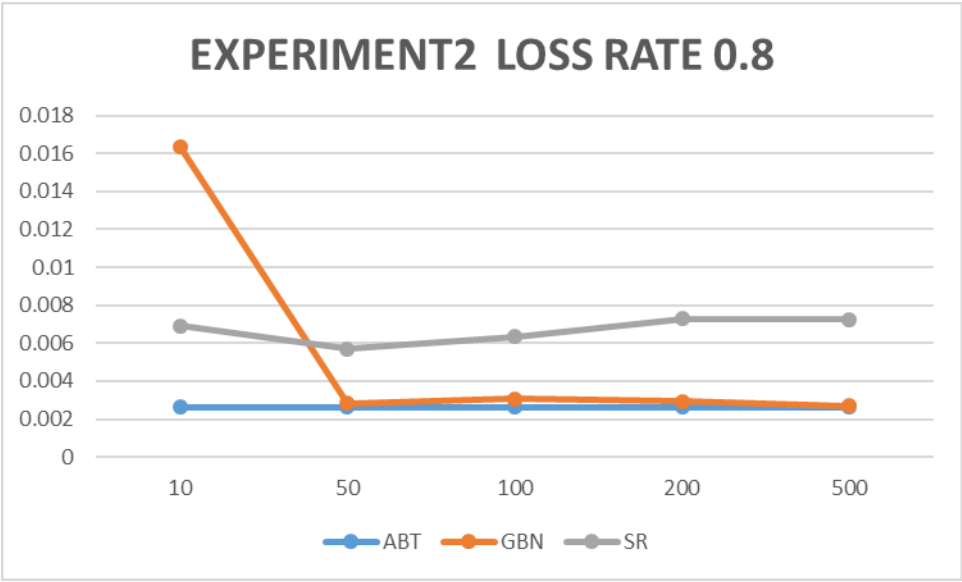
Because ABT doesn't have a window size, so its throughput in this graph is stays in this same value. Both GBN's and SR's throughputs keeps high along the window size changes from 10 to 500, and are both higher than ABT. For they both successfully delivered nearly 999 packages out of 1000. This shows that, under relatively stable channel, GBN and SR performs better than ABT. And, window size doesn't affect the throughput in this case.

Loss Probabilities 0.5



When loss rate rise to 0.5, some changes happen. The SR and ABT remains the same as it is under the loss rate is 0.5. But the throughput of GBN starts to drop as the window size becomes larger and larger, but still higher than ABT. This may because that when window size becomes larger, once times out, the sender will retransmit all the packets in the window. The larger the window, the more packets will be retransmitted. The retransmission costs a lot of time when GBN works. So when simulator stops, in GBN, the sender costs a lot of time in retransmission to successfully deliver enough packets.

Loss Probabilities 0.8



When loss rate reaches 0.8, things are quite different. The throughput of ABT is still the lowest. However, the throughput of SR and GBN also dropped.

As I expected that GBN's throughput dropped dramatically when window size dropped. It is near to ABT's throughput when the window size is larger than 50. This is because that high loss will cause the timer of GBN times out many times so that it will retransmit all the packets in the sender's window. Retransmission costs a lot of time that reduce the throughput. Additionally, the receiver only accepts the packets in a given order. In other words, in this case, large window size causes a lot of retransmissions that cannot be sent in a given time. So when simulator stops, the sender wastes much time in retransmission and receiver fails to get enough packets that in order. For example, when window size is 500, if the second packet was lost, only one packet was successfully accepted by receiver. Even if the rest 498 packets was not lost and corrupted, the receiver will ignore it. Sender need to retransmit all 499 packets. This mechanism wastes time when loss and window size is both high.

Loss Rate	Window Size	Number of Transport Packets
0.8	10	43488
0.8	50	202015
0.8	100	370867
0.8	200	600671
0.8	500	1137533

Figure 3 Transmit packets number of GBN

In my expectation, the throughput of SR will rise slightly when the window size become larger. But when window size is 10, the throughput is higher than the one when window size is 50 and 100. When I check the retransmission packets number on the sender side, I found these data:

loss rate	window size	number of transport packets
0.8	10	13320
0.8	50	103354
0.8	100	196405
0.8	200	383890
0.8	500	717848

Figure 4 Transmit packets number of SR

When window size is 10, the packets that was send out is 13320. But when the window size is 50, the number of packets that was send out is almost 8 times of the one when the window size is 10. The other data shows the same pattern. In my understanding, when window size is 10, the restriction of throughput is loss, which means that here the low throughput is mainly caused by that the packets that were sent out by sender can't reaches the receiver side. Once it reaches the sender side it doesn't need to retransmit again after it get ACK from sender side. Because the sender and receiver both has buffer to mark every individual packets if they are successfully delivered. Once the receiver gets all the packets in order, it will deliver it to application layer

and move the window. So as the sender side. But when window size is higher than 50, the main restriction is possibility that successful delivers the key packet. Only when the receiver gets the packets that is in order, the receiver can move the window to next sequence number. If the key packets keeps loss, the receiver will never move window to next one. This performs the same as it is in the GBN, but since SR has buffer on the receiver side, it has higher throughput than GBN.

Summary

ABT performs worst when the channel is not stable. One needs more time to finish the transmission before the next message come.

GBN performs better than ABT. But when loss is high and window size is large, its throughput is almost the same as ABT. And when the channel is not stable, the performance will drop dramatically when window size become larger.

SR performs best among the three protocol. Its performance rise as the window size become larger. When channel is not stable, it can still perform better than GBN. But it requires a lot buffer size to store the message information and timer, which costs a lot memory space.