Name: Khushboo Tekchandani
Person Number: 50134861
PA2

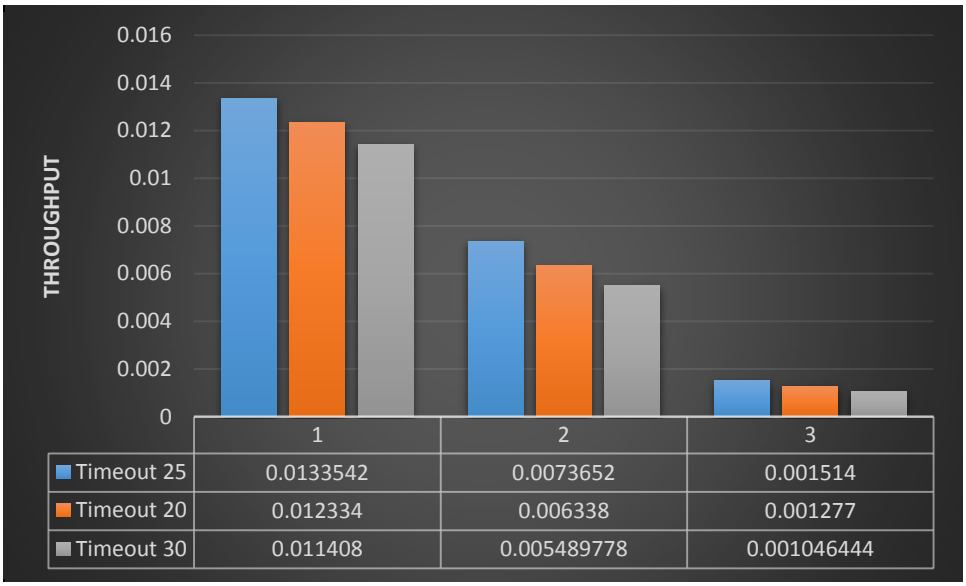
I have read and understood the course academic integrity policy located under this link:
http://www.cse.buffalo.edu/faculty/dimitrio/courses/cse4589_f14/index.html#integrity

4.0

ii) Timeout Scheme

To fix the timeout for my implementation, I compared the results of experiments with different time out values. Time out value of 25 gave the optimum results. Hence, I used the timeout value of 25 units for ABT.

Based on this, I fixed on time out of GBN and SR as 30 units. The time out value for GBN and SR should be a little higher than that for ABT so as to avoid excessive retransmissions.



	1	2	3
Timeout 25	0.0133542	0.0073652	0.001514
Timeout 20	0.012334	0.006338	0.001277
Timeout 30	0.011408	0.005489778	0.001046444

*Loss probabilities: 1 -> 0.2, 2 -> 0.5, 3 -> 0.8

iii) SR multiple timers using one hardware timer

The selective repeat protocol requires the user to set separate timers for different packets. The hardware limits the ability of the user to set explicit timers using one hardware timer. So, to solve this issue, one needs to manipulate the single timer.

To do this, I have fixed a timeout (**TIMEOUT**) value to begin with. Upon sending a packet from the sender, I keep track of time at which the packet was sent. I am doing this using a **struct pkt_ACK**.

```
struct pkt_ACK{
    int seqnum;
    int flag_ACK_recv;
    float time_sent;
}
```

This structure stores the sequence number of the packet along with value of the time at which the packet was sent and whether ACK was received for this packet. The time can be noted using the global variable **'time'.**

We know that the timeout event for any packet will be generated at a time time_sent + TIMEOUT. Using this knowledge, when a timeout event occurs or when an ACK is received for a packet, I set the timeout for the next packet. And start a timer for the newly calculated timeout.

TIMEOUT_new = time_sent + TIMEOUT - ack_time

TIMEOUT_new = time_sent + TIMEOUT - interrupt_time

4.1

Validations:

ABT:

khushboo@ubuntu:~/Desktop/MNC/PA2/PA2_new$ ./a.out -s 1234

----- Stop and Wait Network Simulator Version 1.1 --------

Enter the number of messages to simulate: 10

Enter packet loss probability [enter 0.0 for no loss]:0.2

Enter packet corruption probability [0.0 for no corruption]:0.2

Enter average time between messages from sender's layer5 [> 0.0]:1000

Enter TRACE:1

Protocol: ABT

[PA2]10 packets sent from the Application Layer of Sender A[/PA2]

[PA2]17 packets sent from the Transport Layer of Sender A[/PA2]

[PA2]14 packets received at the Transport layer of Receiver B[/PA2]

[PA2]9 packets received at the Application layer of Receiver B[/PA2]

[PA2]Total time: 11587.205078 time units[/PA2]

[PA2]Throughput: 0.000777 packets/time units[/PA2]

GBN:

khushboo@ubuntu:~/Desktop/MNC/PA2/PA2_new$ ./a.out -s 1234 -w 10

----- Stop and Wait Network Simulator Version 1.1 --------

Enter the number of messages to simulate: 20

Enter packet loss probability [enter 0.0 for no loss]:0.2

Enter packet corruption probability [0.0 for no corruption]:0.2

Enter average time between messages from sender's layer5 [> 0.0]:50

Enter TRACE:1

Protocol: GBN

[PA2]20 packets sent from the Application Layer of Sender A[/PA2]

[PA2]37 packets sent from the Transport Layer of Sender A[/PA2]

[PA2]30 packets received at the Transport layer of Receiver B[/PA2]

[PA2]17 packets received at the Application layer of Receiver B[/PA2]

[PA2]Total time: 1157.228394 time units[/PA2]

[PA2]Throughput: 0.014690 packets/time units[/PA2]

SR:

khushboo@ubuntu:~/Desktop/MNC/PA2$ ./a.out -s 1234 -w 10

----- Stop and Wait Network Simulator Version 1.1 --------

Enter the number of messages to simulate: 20

Enter packet loss probability [enter 0.0 for no loss]:0.2

Enter packet corruption probability [0.0 for no corruption]:0.2

Enter average time between messages from sender's layer5 [> 0.0]:50

Enter TRACE:1

Protocol: SR

[PA2]20 packets sent from the Application Layer of Sender A[/PA2]

[PA2]41 packets sent from the Transport Layer of Sender A[/PA2]

[PA2]30 packets received at the Transport layer of Receiver B[/PA2]

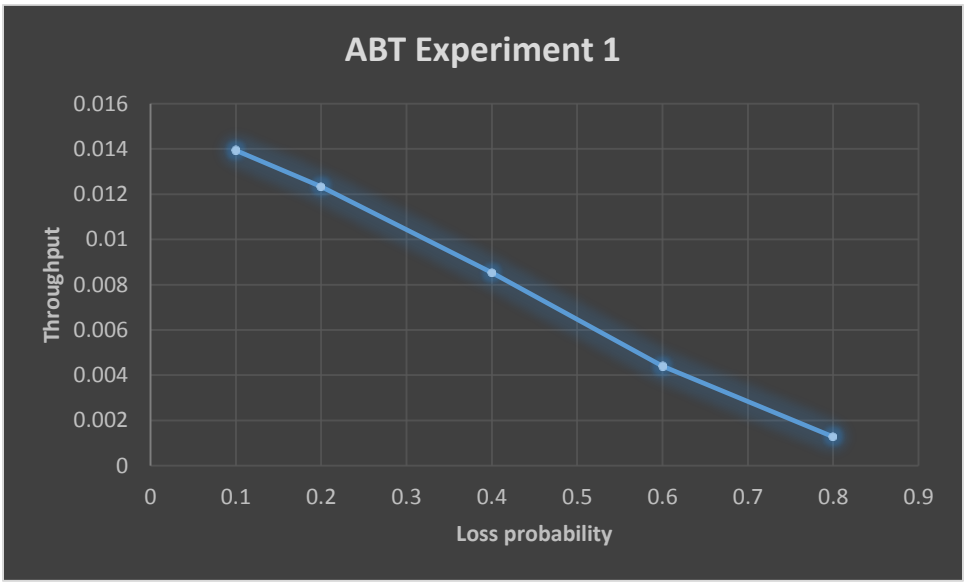
[PA2]19 packets received at the Application layer of Receiver B[/PA2]

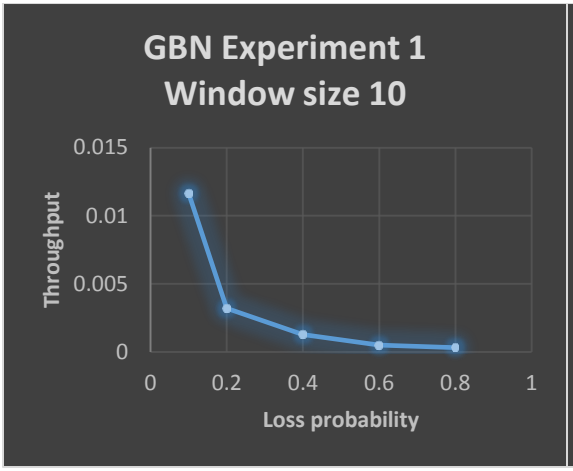
[PA2]Total time: 1100.528320 time units[/PA2]

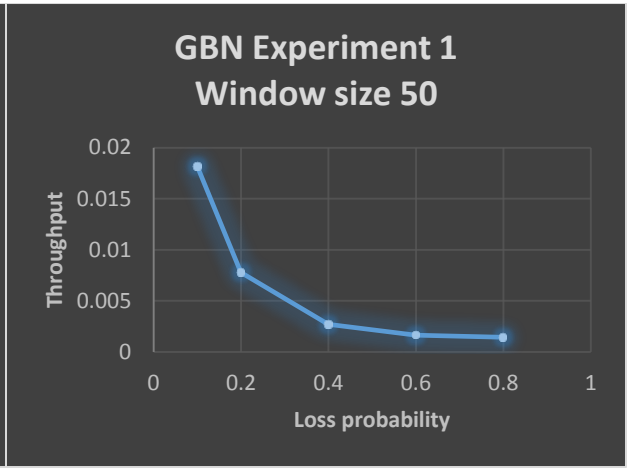
[PA2]Throughput: 0.017264 packets/time units[/PA2]

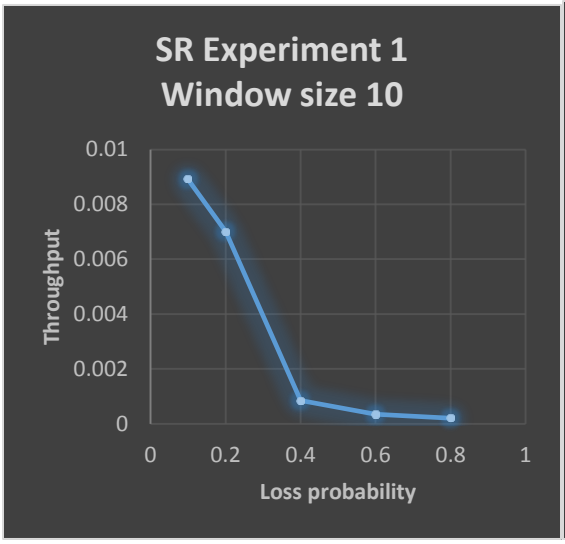
4.2

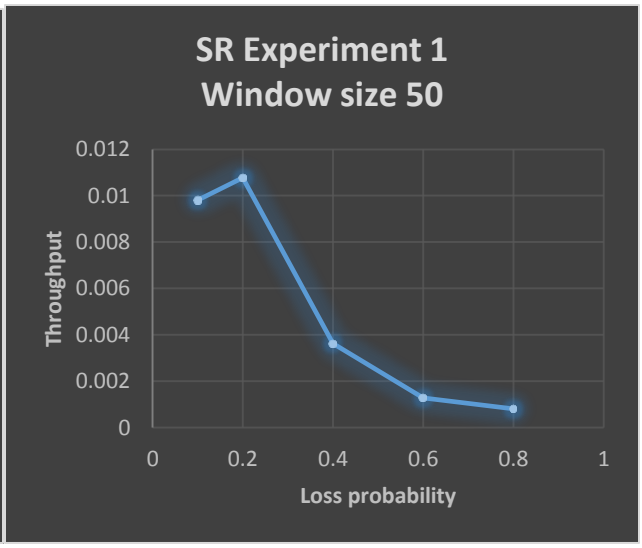
Results of Experiment 1:



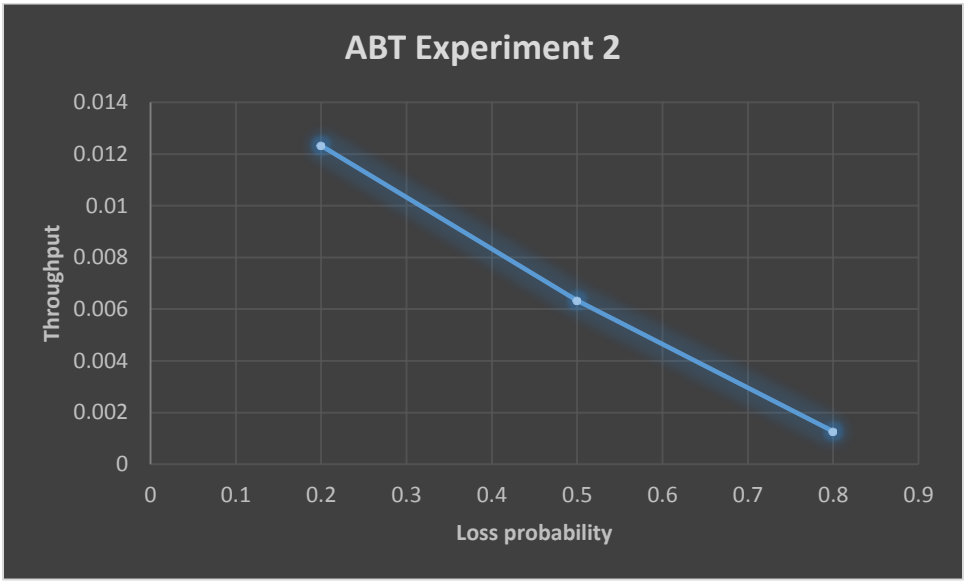


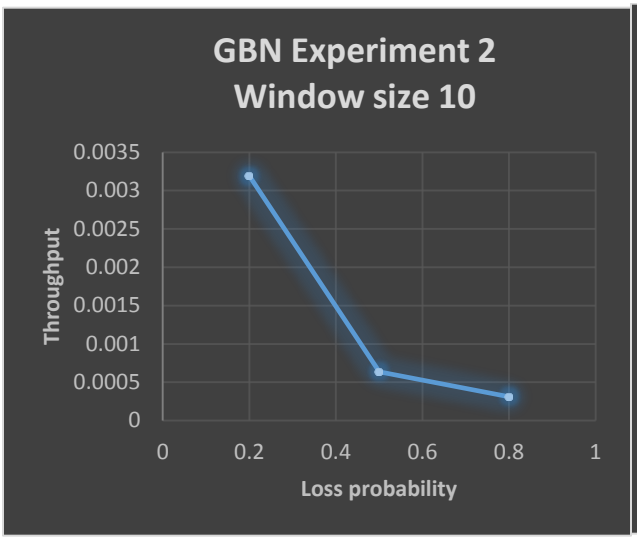


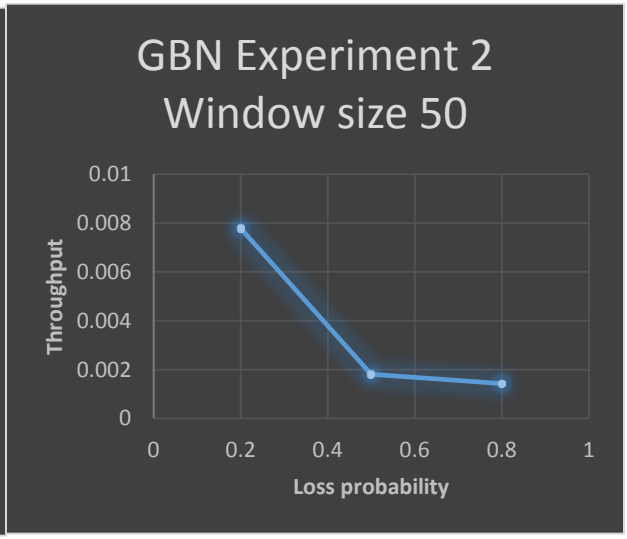

SR Experiment 1
Window size 10
Throughput
0.01
0.008
0.006
0.004
0.002
0
0
0.2
0.4
0.6
0.8
1
Loss probability

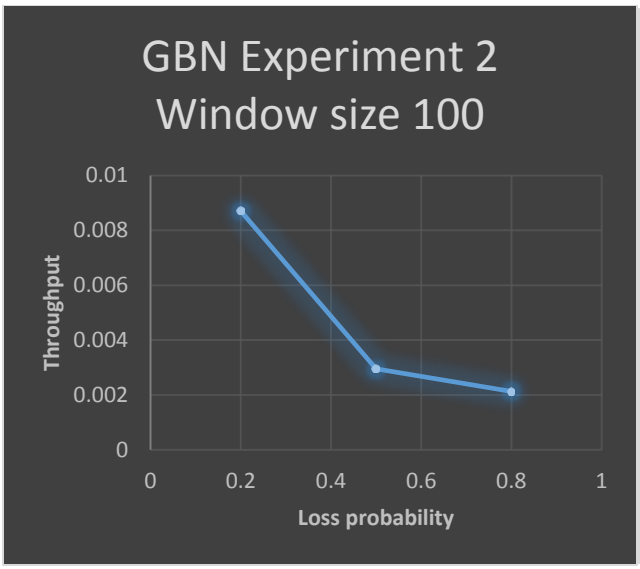

SR Experiment 1
Window size 50
Throughput
0.012
0.01
0.008
0.006
0.004
0.002
0
0
0.2
0.4
0.6
0.8
1
Loss probability

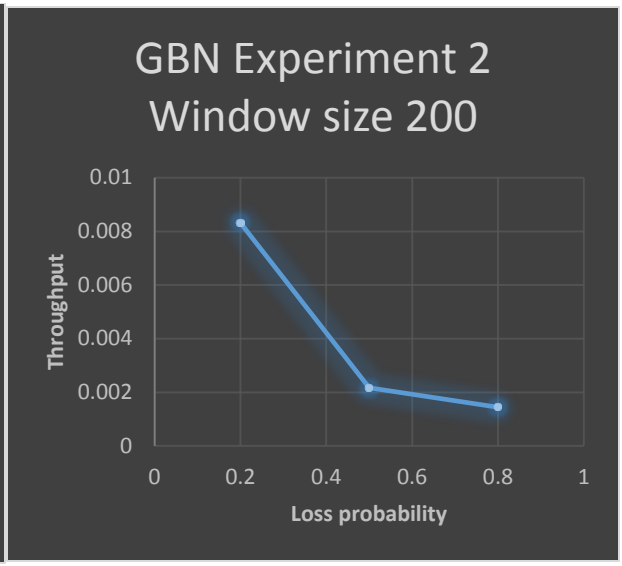
Results of Experiment 2:

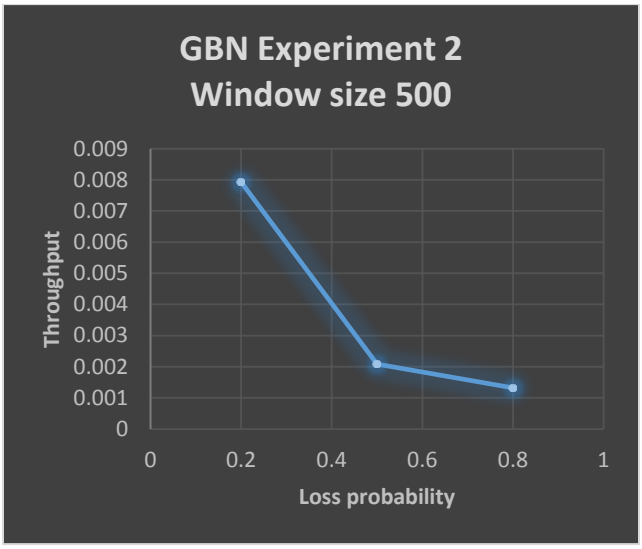


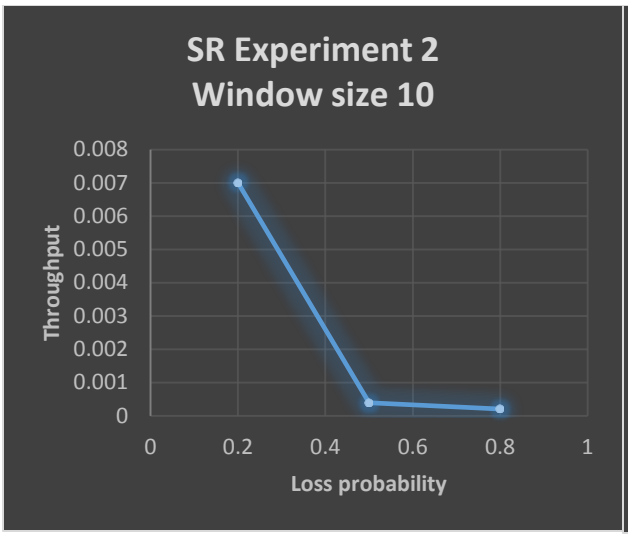


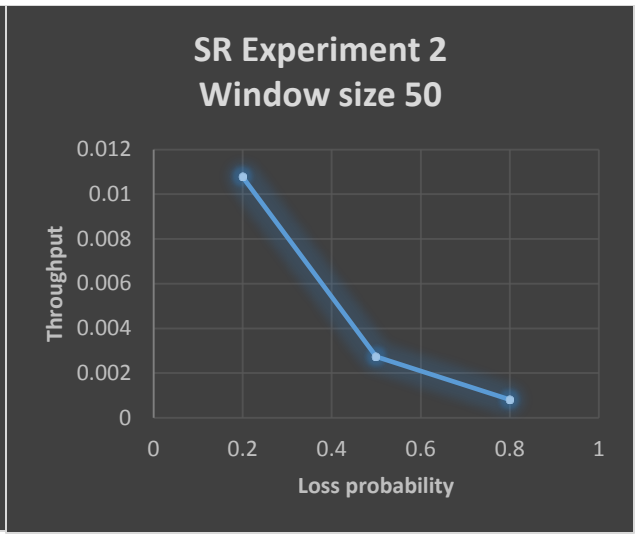


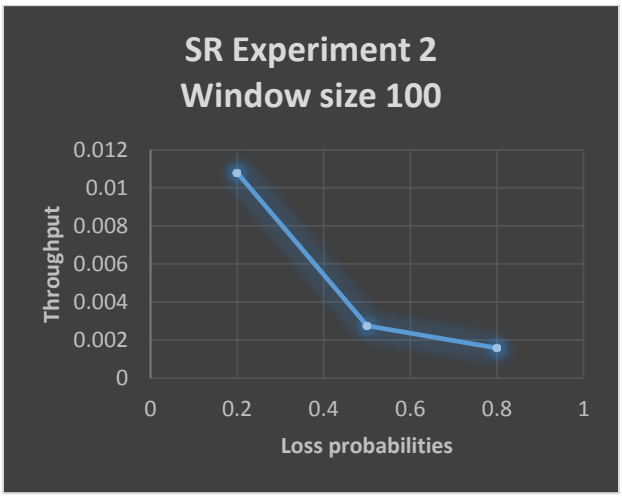

GBN Experiment 2
Window size 100
Throughput
0.01
0.008
0.006
0.004
0.002
0
0
0.2
0.4
0.6
0.8
1
Loss probability

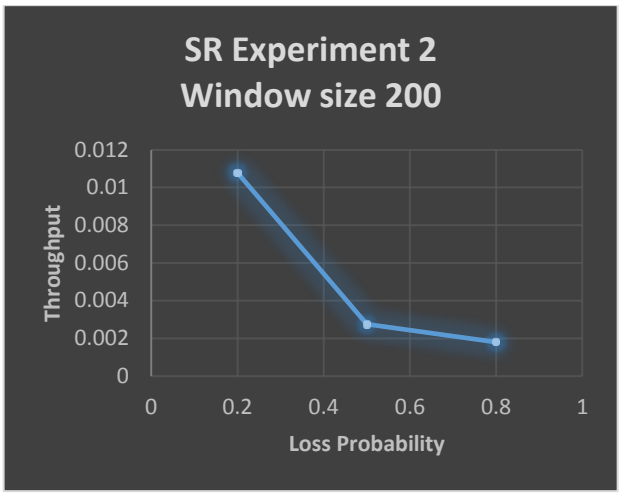

GBN Experiment 2
Window size 200
Throughput
0.01
0.008
0.006
0.004
0.002
0
0
0.2
0.4
0.6
0.8
1
Loss probability


GBN Experiment 2
Window size 500
Throughput
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0
0.2
0.4
0.6
0.8
1
Loss probability









Comment on Window Size:

GBN:

The performance of the protocol is the best when the window size is kept to 100. For the give number of packets and the set of Window size values, 100 is the optimum. This is because, the sender is allowed to send a large number of packets at a time (when loss probabilities are low) and the retransmissions lesser than those in higher window sizes (when the loss probabilities are high).
Thus, 100 is a balanced window size in the two cases.

SR:

The performance of SR increases with the increase in window size. This is because it allows the sender to send higher number of packets and only the lost/corrupted packets are retransmitted.

Comparison between the protocols:

From all the experiments, we can notice that the performance of ABT < GBN < SR. The practical results are in tandem with theory.

The performance of SR is better than GBN as it retransmits only selected packets which are dropped/corrupted or the ACK for which have not been received during the expected time interval as opposed to GBN which retransmits an entire window of packets.

And, the performance of GBN is better than that of ABT as it allows the sender to send a whole window of packets unlike ABT which can send only one packet at a time.

Comparison with theory:

The behavior of the implementation of protocols is quite close to theory. However, the absolute throughout values that I observed from the various experiments are a little lower than expected theoretical values.