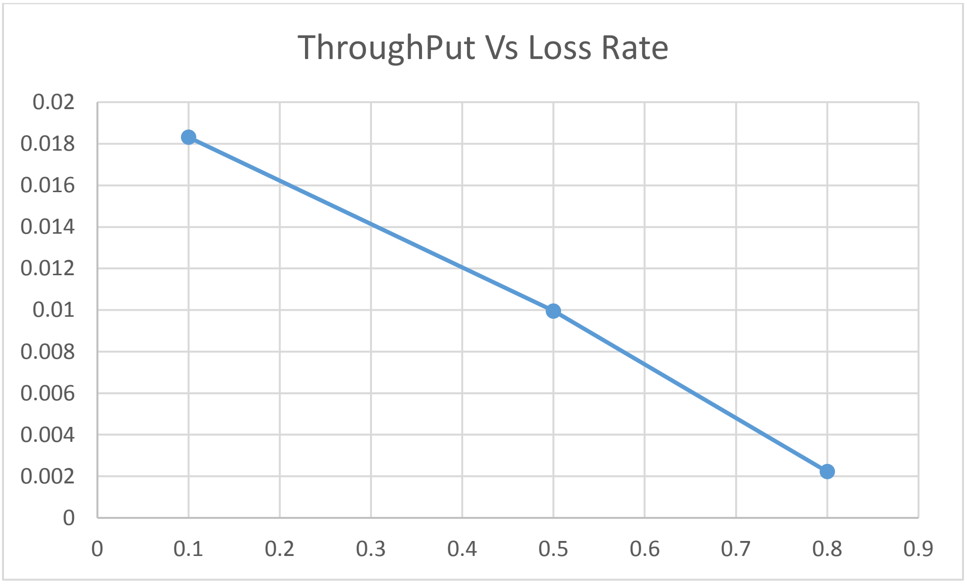
Experiment 1 Observations:

Abt: Running Conditions:

Varied loss probabilities (0.1,0.2,0.4,0,6,0.8) over 0.2 corruption probabilities sent 1000 messages form sender side.

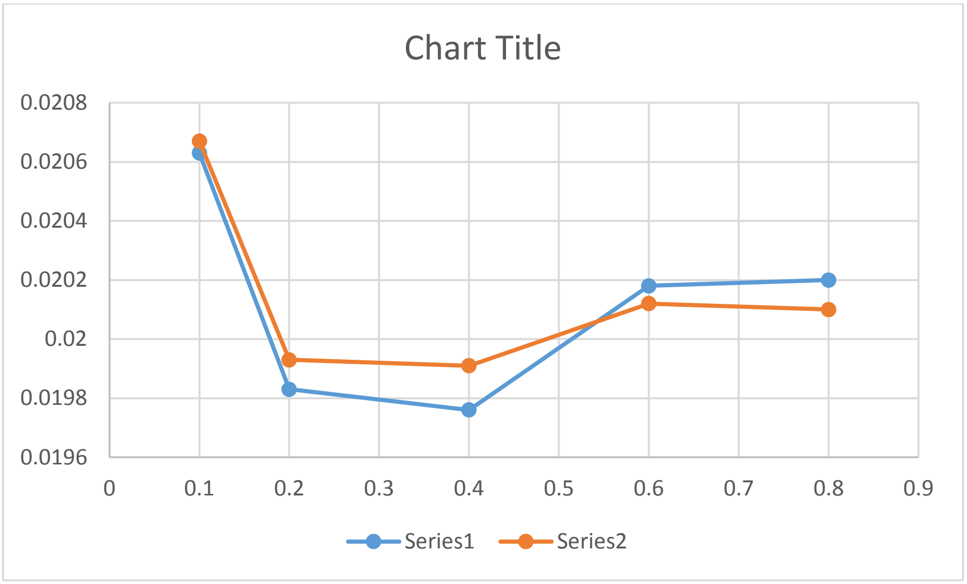
Loss Probability	Throughput	Messages Delivered to Side B	Timeout
0.1	0.01832	900	15
0.2	0.01698	855	15
0.4	0.1214	612	15
0.6	0.00727	359	15
0.8	0.00223	113	15



This graph compares the throughput vs the loss rate, due to the fact that ABT does not buffer any packets on either side and drops new packets until acknowledgments for past packets are received, it can’t cope with higher loss rates.

GBN Observations:

Loss Probability	Timeouts	Throughput		Timeouts	Throughput
	Window Size 10			Window Size 50	
0.1	25	0.02063		25	0.02067
0.2	25	0.01983		25	0.01993
0.4	25	0.01976		25	0.01991
0.6	20	0.02018		20	0.02012
0.8	15	0.02020		16	0.02010

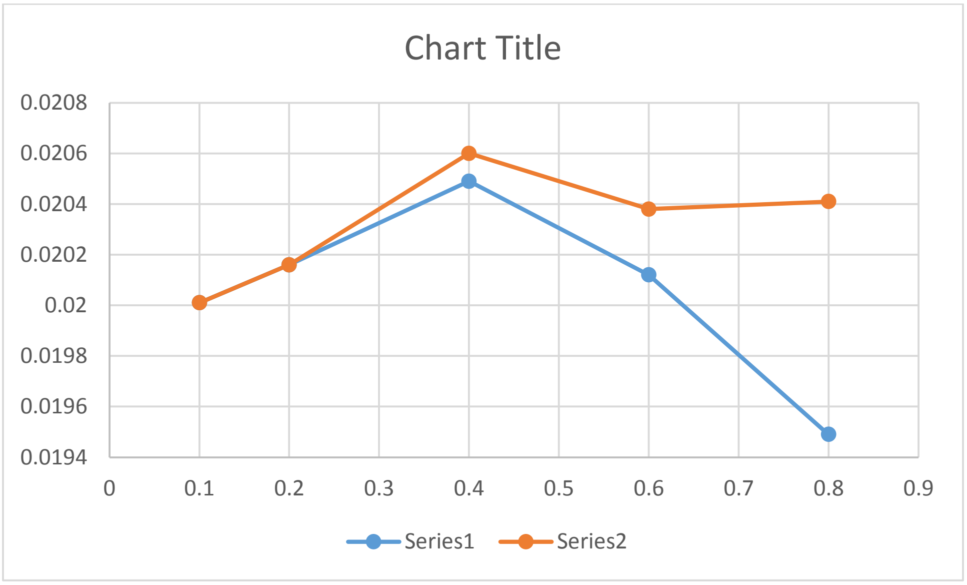


In Go Back-N, the timer is set for only the last packet sent, If initially while sending, the buffer is not full then every-time new packet is sent to side B, timer is set (in my implementation). Due to this behavior, GBN sender is sluggish in reacting to losses, therefore the timeout values are fairly low, especially for the higher loss rates. In case of both windows 10, 50 the timer values are fairly same and my sender is able to deliver almost all the packets 999 and in one or two cases 998 packets are sent over.

Intuitively, as the loss rate goes up the throughput should drop and that behavior is maintained in my observations as well. Interestingly the throughput does go up for loss rates 0.6 and 0.8 for window size 10, this is due to the lower timeouts that I have set for these cases, almost assuming the packets are lost.

SR Observations:

Loss Probability	Timeouts	Throughput		Timeouts	Throughput
	Window Size 10			Window Size 50	
0.1	33	0.02001		33	0.02001
0.2	33	0.02016		33	0.02016
0.4	33	0.02049		33	0.0206
0.6	27	0.02012		27	0.02038
0.8	27	0.01949		27	0.02041



In case of selective repeat my sender, is able to deliver almost all messages (999, in some cases 996-997) as in case of GBN. The throughput varies a lot in SR's case that is due to the timeout values suiting some loss rates more than the other. Comparing two windows, it is clear the throughput is more in case of window 50, In fact the gap widens as we go to higher loss rates. Higher throughput must be happening due to higher window size, In case of window size 50, suppose a message at sequence no. 10 is not delivered then messages 40-50 can still be delivered, which gives message at 10 more chances to get delivered. IN case of window size 10 if a message at 10 is stuck and not delivered then only messages up to 20 might be delivered. Thus in case of window size 10 a message which is not delivered soon enough can hold up the whole system, in case of window size 50 there is less chance of that.

The fact that receiver must acknowledge every packet separately in case of selective repeat can be a bottleneck in performance, not only the sender has to deliver them again, the acknowledgements must make their way back to sender. This is more problematic in case of higher loss rates. I have tweaked the receiver a little to achieve better performance. Whenever the receiver receives a duplicate packet of a lower sequence number than it's window start (meaning that acknowledgement was lost), it will send out all the acknowledgements between that sequence number and it's window start. E.g. receiver window starts at 10 and it receives packet 7 again then my receiver will send all acknowledgements from 7 to 10. Though it is overhead and can slow down the network, in practice it has worked really well. This tweak, in only implemented for higher loss rates (>0.5).

To give an estimate of how much the above sending acknowledgements on a roll tweak works..

For windows size 10, loss rate 0.8 only 537 packets made it to receiver and throughput was a dismal 0.0157 without this tweak.

537 of packets received at the Application layer of Receiver B

Total time: 50786.160156 time units

Throughput = 0.01057 packets/time units

With this tweak, window size 10 loss rate 0.8 almost **double** throughput is achieved 0.01949 and 996 packets make way to receiver.

996 of packets received at the Application layer of Receiver B

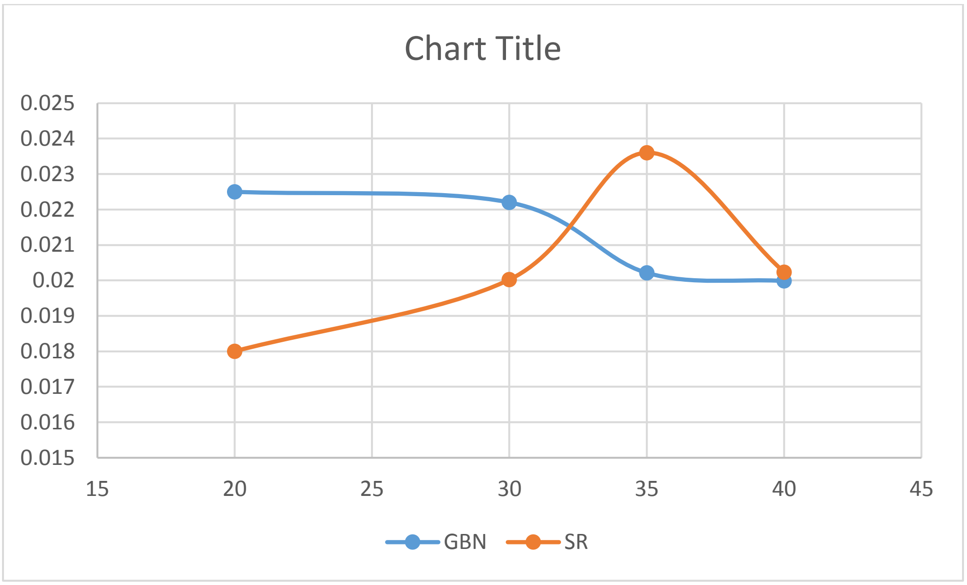
Total time: 51092.777344 time units

Throughput = 0.01949 packets/time units

This tweak, works equally badly for lower loss rates as those acknowledgments flood the network (as they make their way on the other side on more occasions), and thus is not used for loss rates less than 0.5

Summarizing Experiment 1:

The sliding window protocols definitely deliver the goods compared to Abt, they do deliver fast faster and can cope with harsh networks. Now, comparing the two sliding window protocols:



This graph compares two protocols for loss rate 0.5 and for varying time out values compares their throughput. It clearly proves that, every protocols has their own comfort zone w.r.t. timeout and outside that particular timeout range it’s performance will be poorer.

Lower timeouts suite GBN as it has single timer for the last packet, and is sluggish in responding to packet losses. It could be argued that because GBN sends all the packets belonging to window higher timeout may be a good idea as these packets may flood the network, but it doesn't seem to affect much in practice may be due to higher loss rates.

With selective repeat, It has separate timer for each every packet, therefore higher (compared to GBN) timeout makes sense. In fact, lower timeouts result in very poor performance, only 890 packets getting delivered out of 1000. Lower timeouts also mean false alarms, which trigger unnecessary retransmissions flooding and slowing down network even further, and cycle goes on causing more false alarms. Thus higher timeout's work for selective repeat.

Experiment 2:

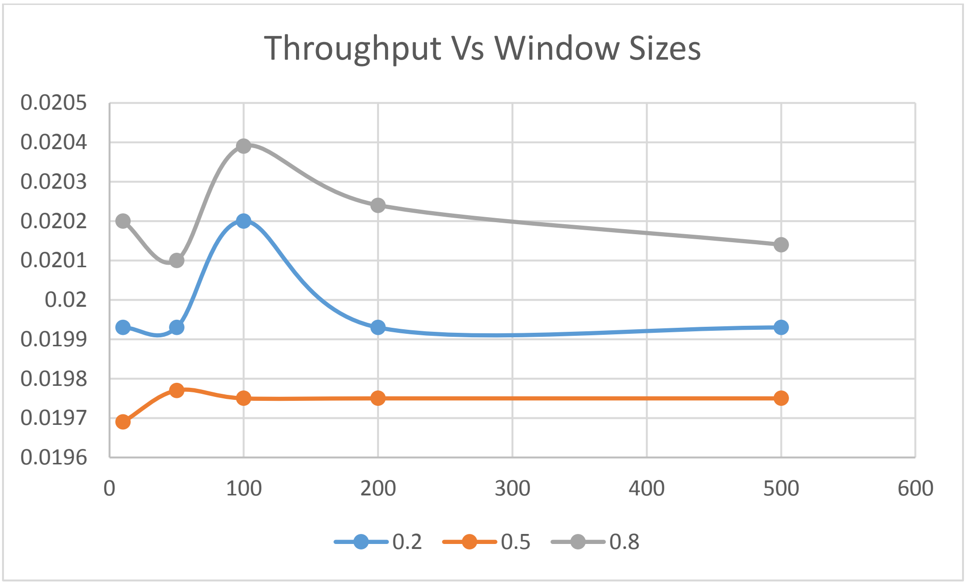
Abt:

Loss Probabilities	Throughput	Number of packets delivered
0.1	0.01832	900
0.5	0.00996	506
0.8	0.00223	113

Clearly, with higher loss rates ABT drops incoming packets form layer 5, which results in poor throughput.

Go Back-N:

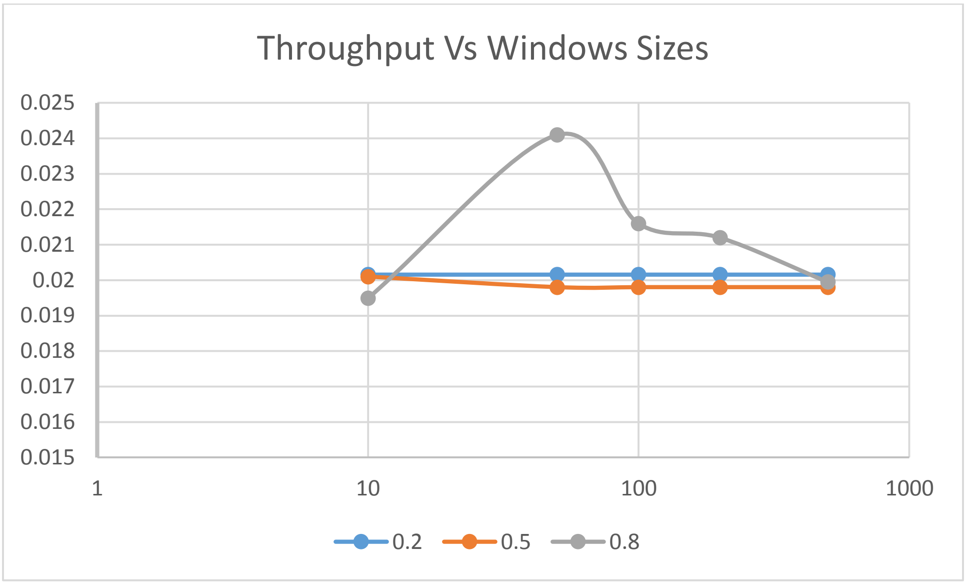
Loss	Throughput for various windows sizes				
Probabilities	10	50	100	200	500
0.2	0.01993	0.01993	0.0202	0.01993	0.01993
0.5	0.01969	0.01977	0.01975	0.01975	0.01975
0.8	0.0202	0.0201	0.02039	0.02024	0.02014



The Go back-N performs best when window size is 100, in case of sizes 10 and 50 performance is poorer because windows sizes are too small, so if a packet in the window is not delivered soon enough then it may hold the progress of the window. For larger too many messages must be flooding the network, thus reducing its performance.

Selective Repeat:

Loss	Throughput for various windows sizes				
Probabilities	10	50	100	200	500
0.2	0.02016	0.02016	0.02016	0.02016	0.02016
0.5	0.0201	0.0198	0.0198	0.0198	0.02011
0.8	0.01949	0.0241	0.02016	0.02012	0.01995



This graph shows the varying throughput vs. window sizes, for selective repeat. Its clear form graph that for window size 50 the protocol performs relatively better than for other window sizes. For window size 10 the problem is that it is too small to feel performance of SR. With sizes 100, 200, 500 the number of messages are too large for the network, which result in network congestion and thus false alarms w.r.t timeout.