Analysis_Assignment2
I have read and understood the course academic integrity policy.

Name: Srinivas Ravi
UB Number: 50244669

Brief description of the timeout scheme you used in your protocol implementation and why you chose that scheme.

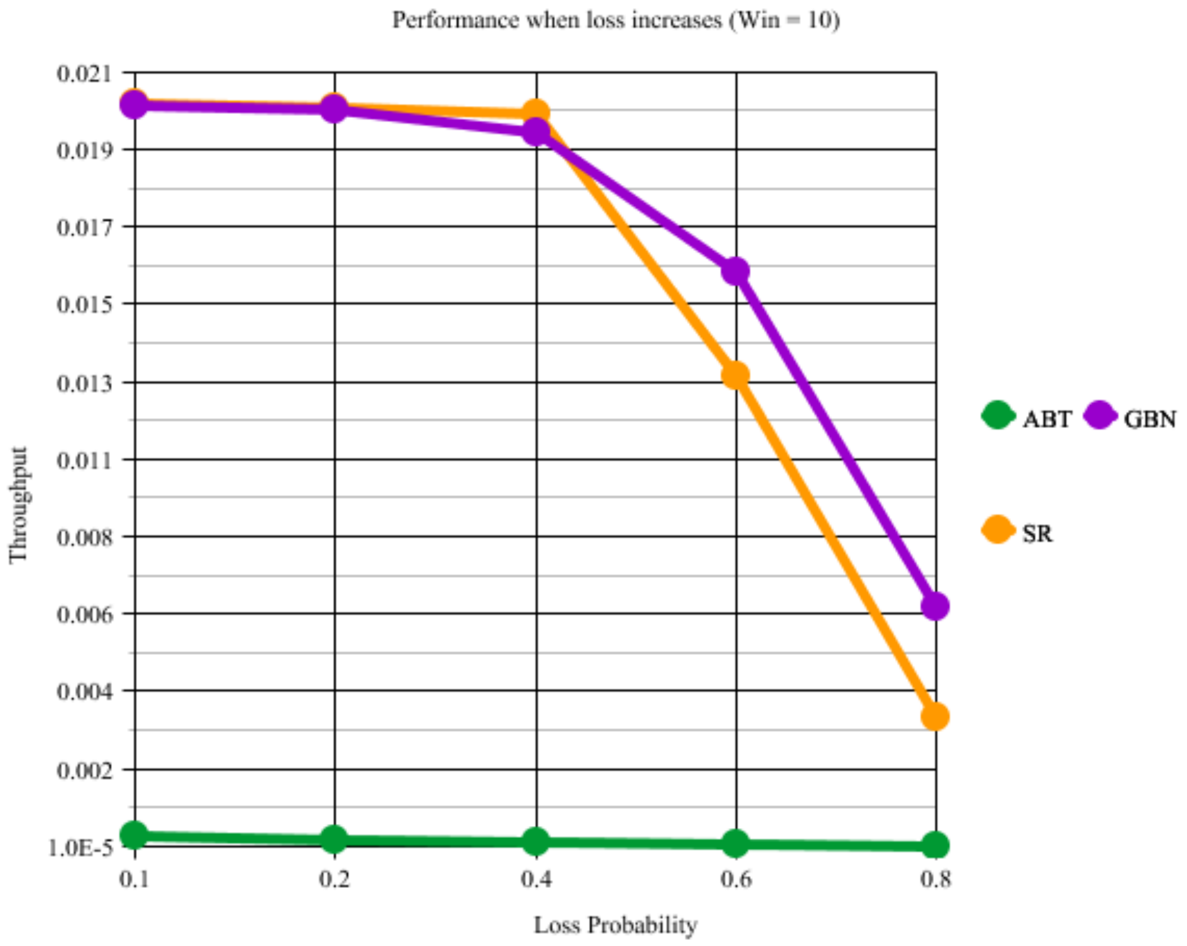
- I used static timeout of 20-30 time units. 30 units for ABT and GBN and 20 units for SR.
- A value lower than 10 units would fail to send messages even with zero corruption and loss. That's because the time from one end to another end is around 5 units i.e RTT is around 10 units. Thus values above 10 are preferable. Also, as the corruption and losses increase, even slightly higher value than 10 would be ineffective. That is why I avoided going below 20.
- A very high value of timeout >40 would make the communication slower. Though such a value would avoid congestion, longer timeouts means more time before retransmission and if those packets too get corrupted and lost, even more delay in transmission.
- Hence, I chose the optimal range of 20-30 time units.

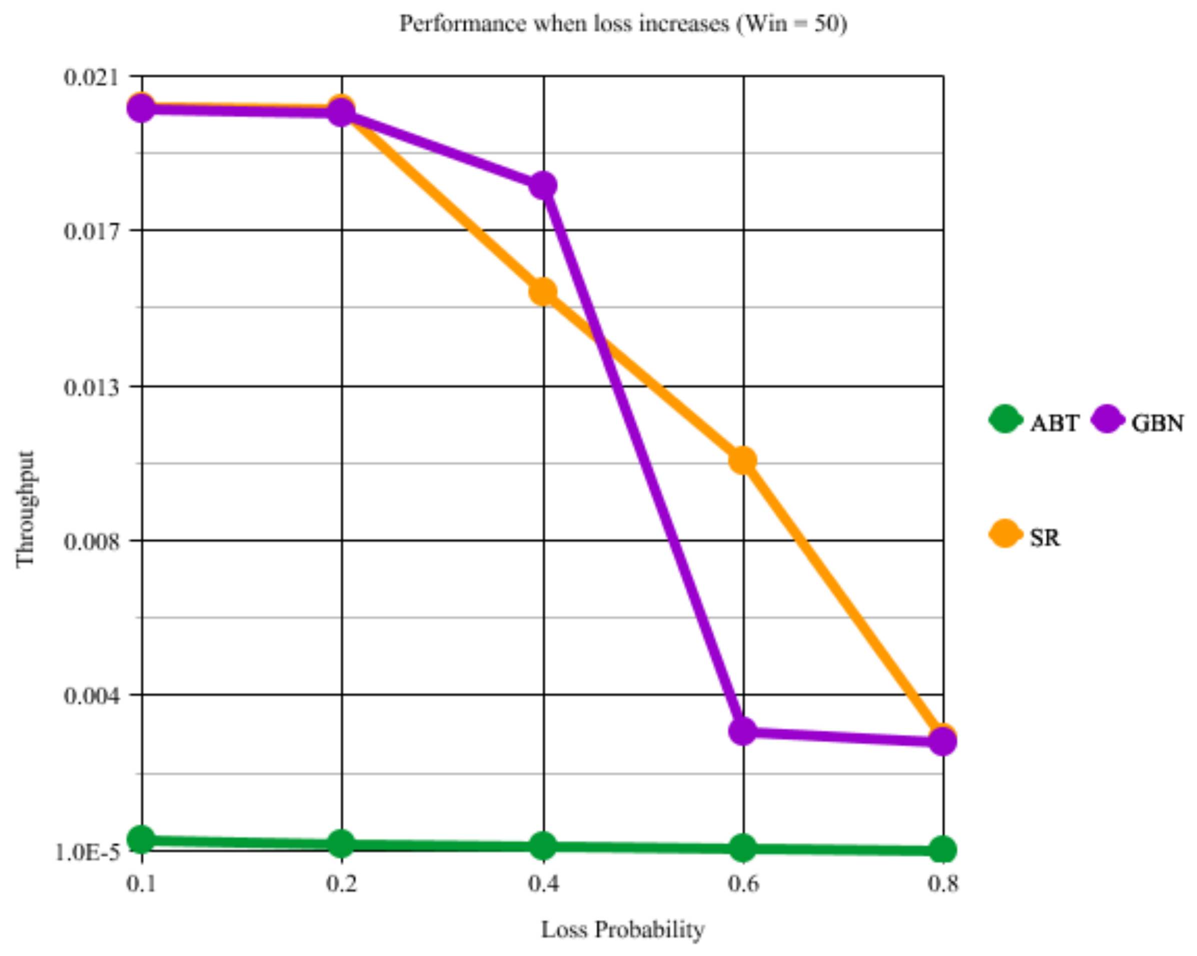
Brief description (including references to the corresponding variables and data structures in your code) of how you implemented multiple software timers in SR using a single hardware timer.

- I implemented multiple software timers in SR by storing the time each packet is sent to a network in a linked list. The highlight is the time stored is relative to the previous packet sent to the network. For the first packet the relative time is irrelevant. Because when the first message is sent, the single hardware timer is called with the timeout value(20 units).
- When second packet arrives, it's timeout is calculated with respect to how much more time after the end of current hardware timeout will it's own timeout needs to happen. So a packet arriving 5 units after the first packet, would have timeout stored for itself as 5 time units. This way all the packets being sent to the network store their expected timeout relative to the previously sent packet. If all packets are acked and a message arrives from application, it starts it's own hardware timer with timeout value (20 units).
- The nodes of this linked list having the information of the packets will be created and deleted by the hardware timer interrupt. It's also created each time new message is sent into the network. So when the hardware interrupt happens, it picks up the first node of the linked list, checks if the packet in that node has been acked yet. If yes, then it simply deletes that node. If not, it retransmits the packet at that node and creates a new node at the end of linked list with the updated packet details – relative timeout with respect to the last node's packet, current time (for the next packet to calculate it's relative time), acknowledgement flag = 0 (0=not acked) and the packet itself. The frontmost node is then deleted. It then retrieves the relative timeout information of the new frontmost node and set's the value present in that timeout. At that timeout this cycle is repeated. If only 1 node is found during timer interrupt and it's acked then delete the node and do nothing. Else retransmit that packet again with the default timeout value.
- On arrival of ack, the node in the linked list containing the packet is marked as acked. So that the timerinterrupt knows to delete it once it encounters it.

Experiment 1:

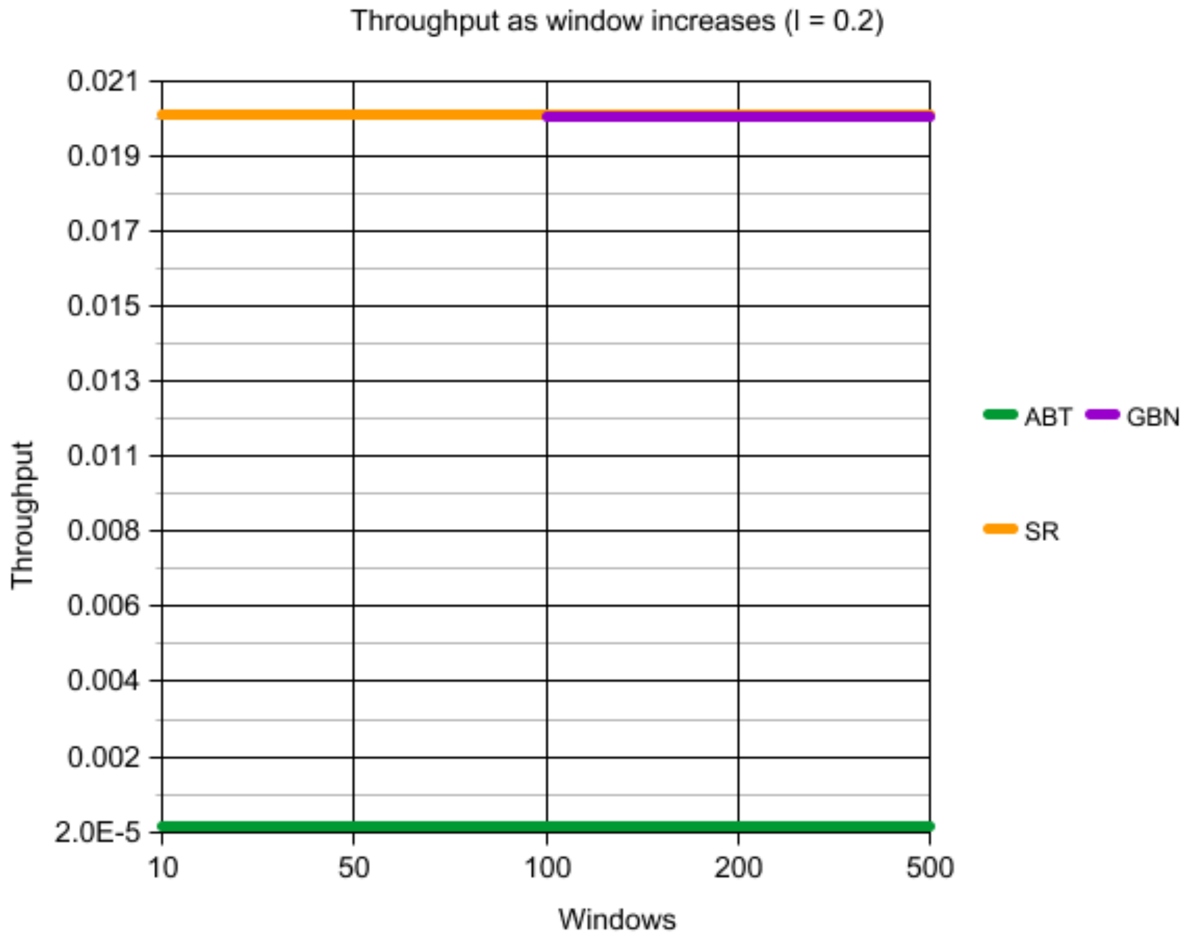
With loss probabilities: {0.1, 0.2, 0.4, 0.6, 0.8}, compare the 3 protocols' throughputs at the application layer of receiver B. Use 2 window sizes: {10, 50} for the Go-Back-N version and the Selective-Repeat Version.

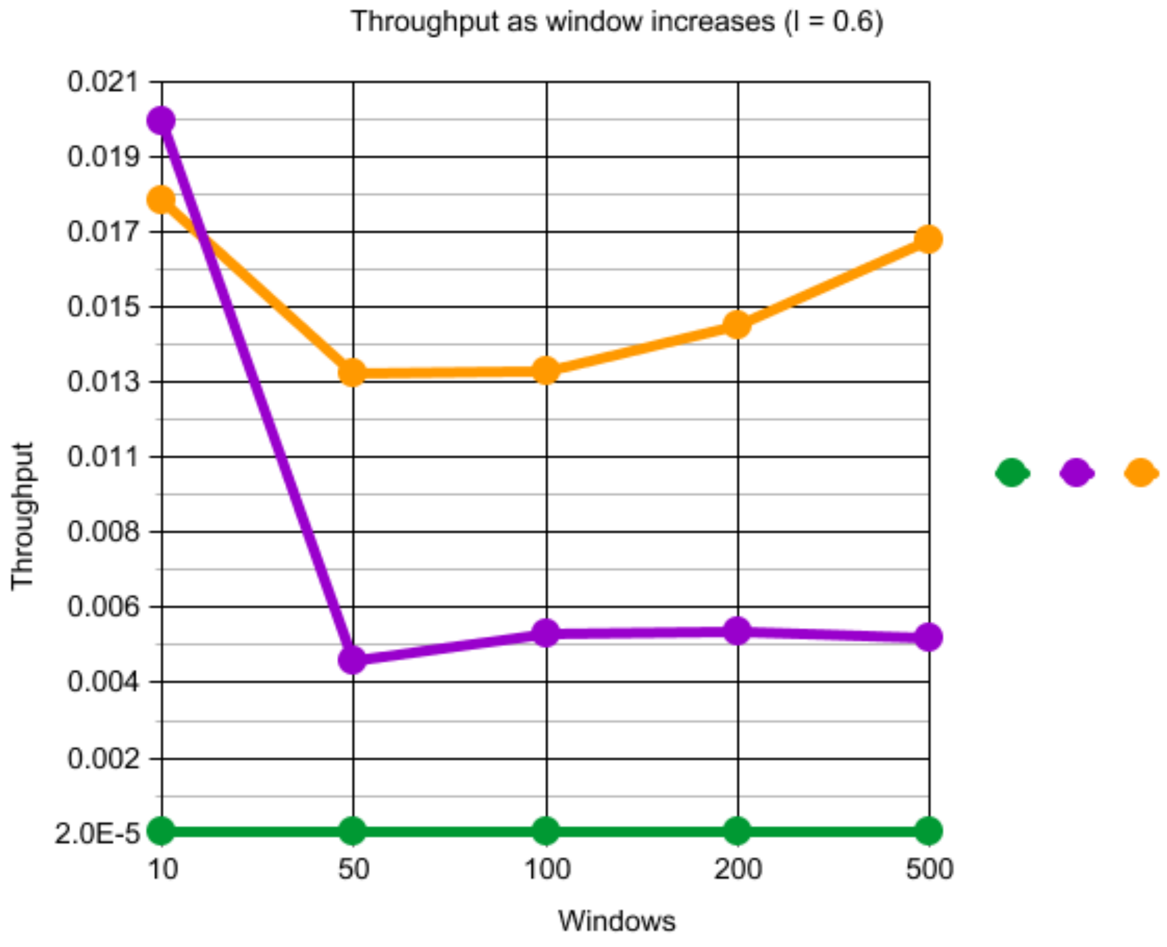


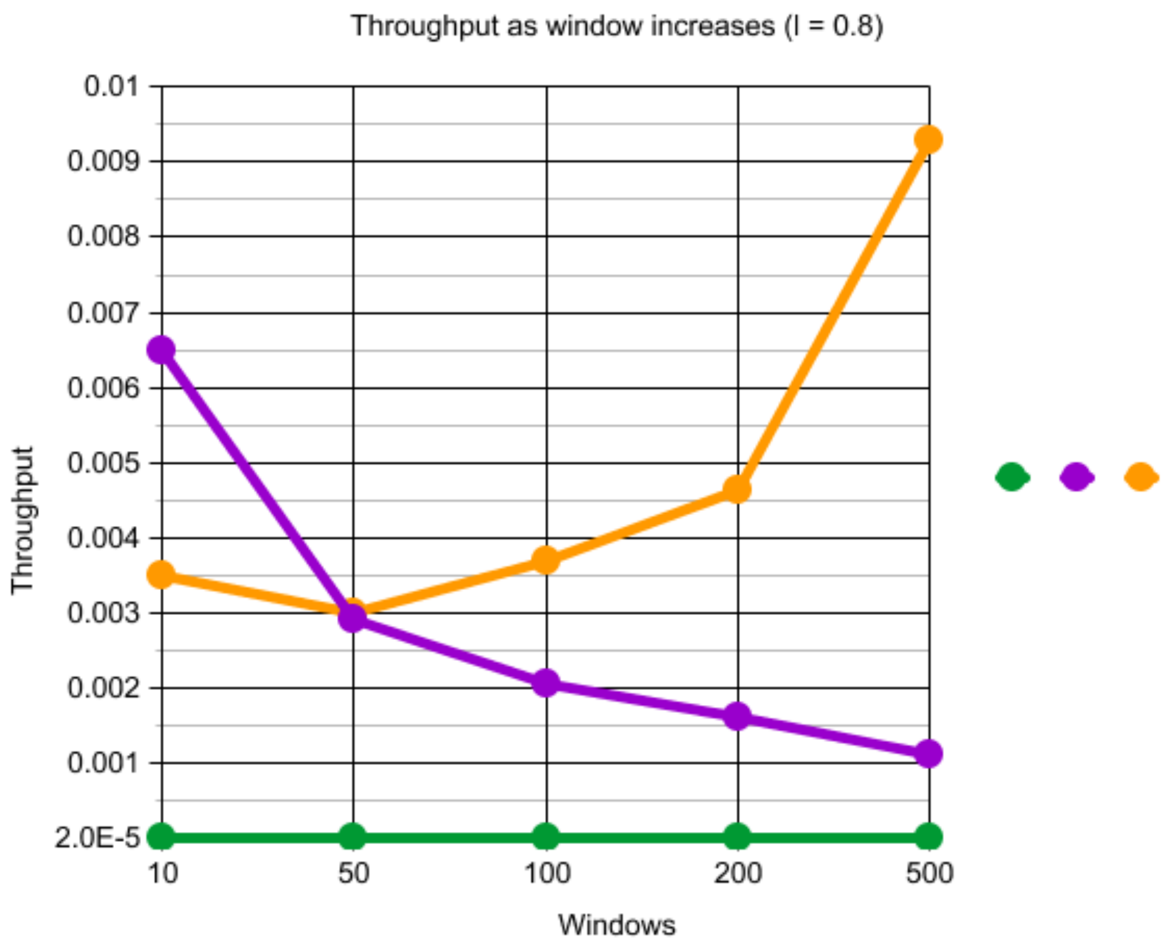


As loss probability increases, number of packets being dropped increases and that decreases the throughput. This fact holds true for all the three protocols. This expected property of decreasing throughput with increasing loss probability can be seen from the two graphs for window sizes 10 and 50 (for GBN and SR) in our experiment.

Experiment 2:
With window sizes: {10, 50, 100, 200, 500} for GBN and SR, compare the 3 protocols' throughputs at the application layer of receiver B. Use 3 loss probabilities: {0.2, 0.5, 0.8} for all 3 protocols.




Throughput as window increases (l = 0.6)
0.021
0.019
0.017
0.015
0.013
0.011
0.008
0.006
0.004
0.002
2.0E-5
Throughput
10
50
100
200
500
Windows



SR displays better throughput as compared to GBN as the repetitive cumulative retransmissions in GBN increase the congestion and makes communication slower. This is particulary expected as window size increases.