CSE 589

ASSIGNMENT 2

Reliable Transport Protocols

WASIF ALEEM, 50161250

I have read and understood the course academic integrity policy.

Introduction

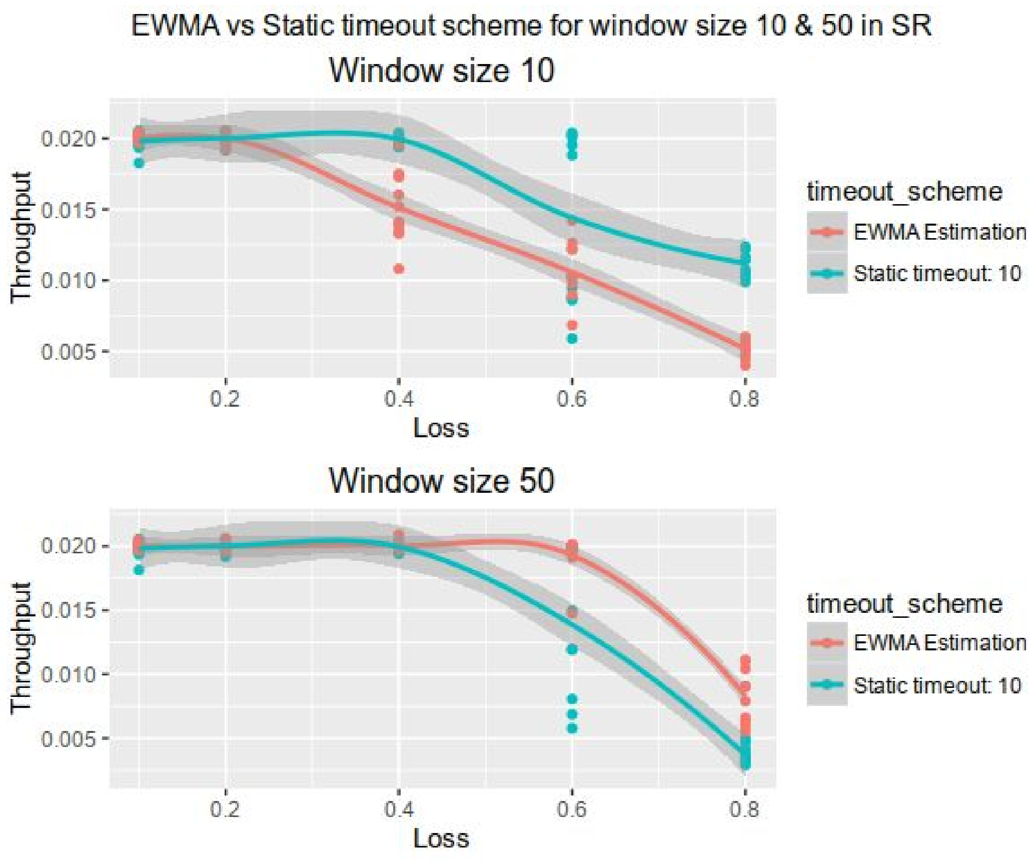
For this assignment we implement three reliable data transport protocols: Alternating-Bit (ABT), Go-Back-N (GBN), and Selective-Repeat (SR). Following sections briefly document my implementation and comparison/analysis of these protocol including the use of timers & timeout scheme.

Timeout Scheme

Choosing a good timeout scheme is crucial, if the timeout is too small, it leads to many retransmissions and too large value are not receptive to loss. The timeout should be at least be greater than RTT.

For each of the three protocols we use the Adaptive EWMA TCP timeout scheme(section 3.5.3 in textbook), which in helps us adapt to loss and retransmissions.

As we can observe from the following graph for Selective Repeat which compares the static timeout scheme with TCP like EWMA scheme, we can observe that for window size 10, static timeout has higher throughput, but as window size increases to 50, EWMA has higher throughput. This is due to increase in utilization when window increases and EWMA can adapt to the instantaneous number of packets in the system(affecting timeouts), but the static timeout remains the same and leads to more retransmissions as loss increases.

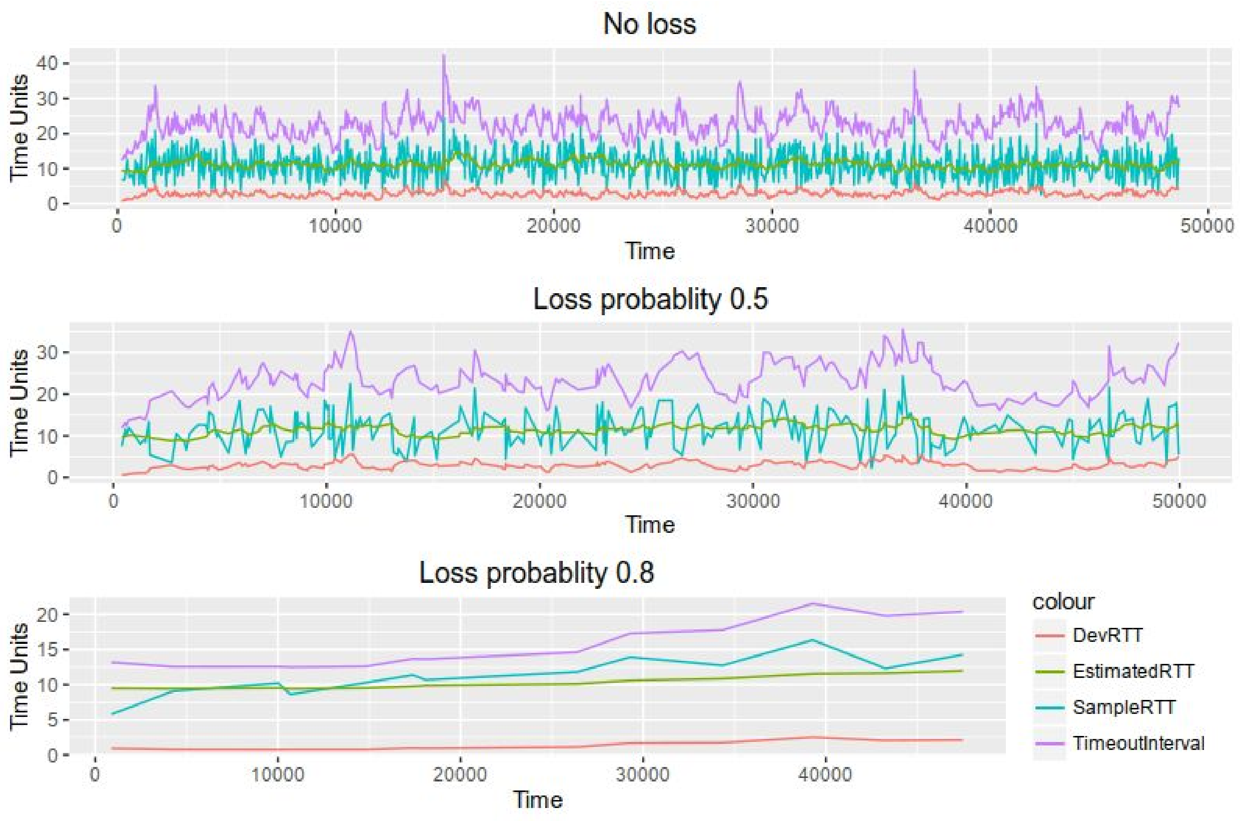


Since the PA2 description mentions that packet sent into the network takes an average of 5 time units to arrive at the other side, we initialize SampleRTT, EstimatedRTT to 10 time units and DevRTT to 0. We use the standard alpha(0.125) and beta(0.25) values.

We record sent time for each packet and compute SampleRTT, EstimatedRTT and DevRTT on successful ACK(carefully excluding retransmissions by flagging them).

On timeout, we double the value of TimeoutInterval only for this retransmission to avoid premature timeout and it also provides a very limited form of congestion control.

Following graph details these measurements in case of protocol Selective Repeat for loss probabilities 0.0, 0.5 and 0.8. As the loss probability increases we get few measurements because most of the packets are not acknowledged. In case of loss probability 0.8 we can clearly see TimeoutInterval increasing, reflecting recent values of SampleRTT.



Multiple Software timers

In Selective Repeat, we maintain timers for each unacknowledged packet. For this purpose we use the single timer routines provided and implement multiple timers by maintaining a priority-queue of timers(`std::priority_queue<timer> timers`), each such timer has an associated sequence number and the time at which it should expire, this gives us a nice property that the first timer in the queue is the first to expire.

On `A_init` we start the single timer with a clock tick of 0.1. Whenever an interrupt fires we check the consecutive top elements of priority-queue and pop the software timer off queue if it expired(`timer.time <= get_sim_time()`) and retransmit(if not cancelled), until we encounter an non-expired timer. This leads to complexity of O(k) where k is the number of timers expired within a clock tick.

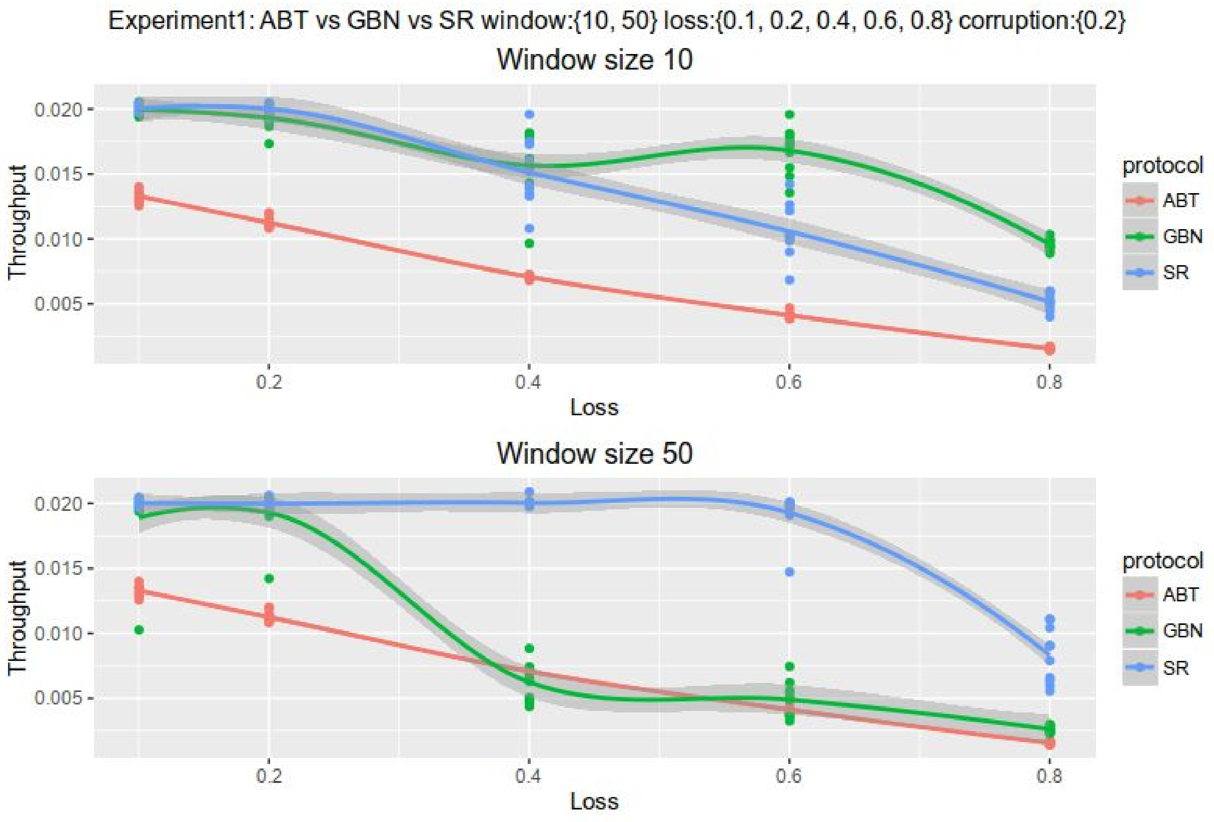
Similarly inserting a timer into priority queue is at most log(n).

We also need a way to cancel a timer. Since STL priority-queue interface does not explicitly provide a delete method, we maintain a set(`std::set<int> cancelled_timers`) of sequence numbers which have requested cancellation of the timer. This allows to trivially check for cancellation(`if (!cancelled_timers.count(timer.seq))`) while processing the queue on hardware interrupt. This leads to cancellation complexity of long(n).

Experiment1

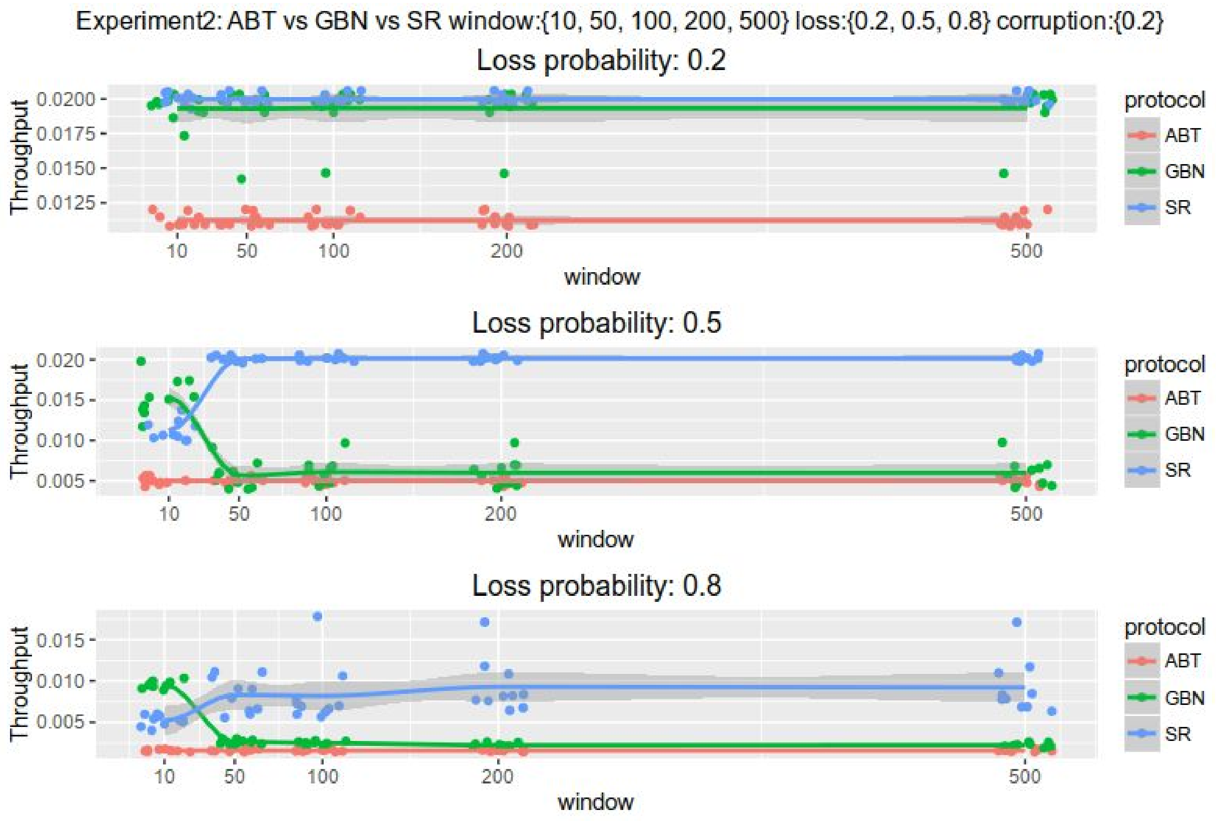
In this experiment we compare the protocols with two window sizes 10 & 50 and increasing loss probabilities. As we increase the loss probabilities, throughput will decline, similarly if the window size increases, throughput should increase because there will be more # of packets in transit(increasing utilization).

As we can observe from the following graph our expectations are indeed met when loss & window size increases. In case of window size 10 throughput of GBN is better than SR, this can be explained due to the fact that on timer interrupt GBN performs retransmission of it's window, which is small enough in this case, but as we see from the graph with window size of 50, GBN throughput decreases with high loss , this is due to the overhead of spurious retransmissions, as expected SR performs considerably better because it's receiver buffers & acknowledges individual packet's reducing retransmissions.



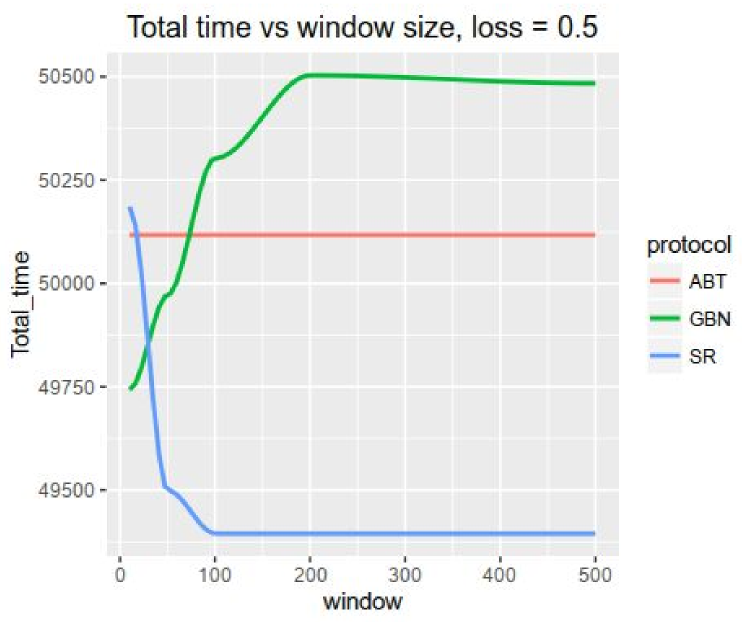
Experiment2

In this experiment we compare the protocols with window sizes 10, 50, 100, 200 & 500 and with loss probabilities 0.2, 0.5 & 0.8. We expect that high window sizes along with high loss probabilities will bring down the throughput of GBN considerable when compared to SR, which should perform better under all three loss probabilities compared with GBN, because it maintains separate logical timers which helps it in detecting loss eagerly.

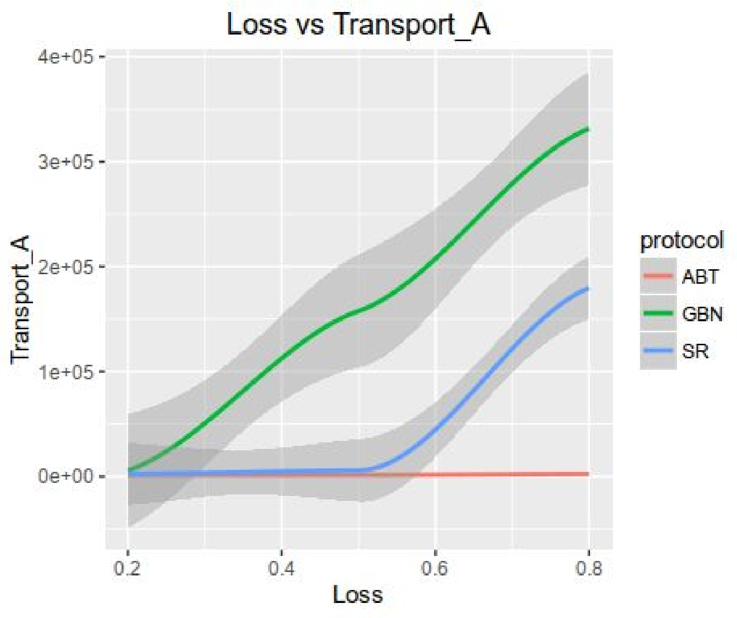


From the graph, ABT has constant throughput with respect to window size and decreases as loss increases, in the case of window size 10 GBN performs well, but as window size increases & high loss we notice a substantial decline in GBN throughput(as previously attributed to retransmissions). We can observe that SR performs better overall, this again is due to it's ability to buffer packets as large as its window size, eager loss detection which retransmits only the necessary packets even in high loss conditions.

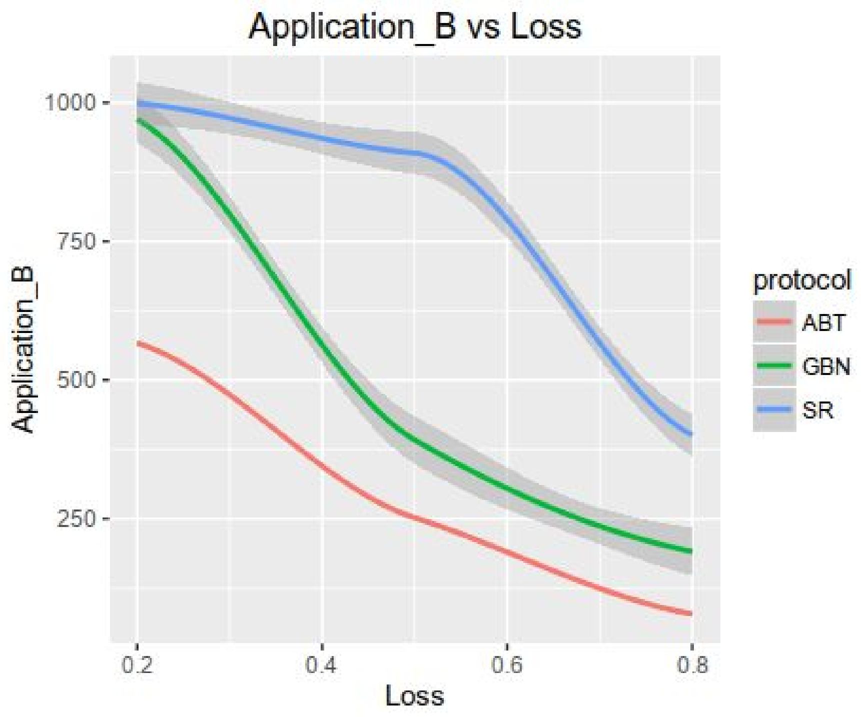
Following graph compares Total time taken and window size at a particular loss of 0.5, we can clearly see that total time taken by GBN is considerably large, whereas SR has the lowest total time as expected.

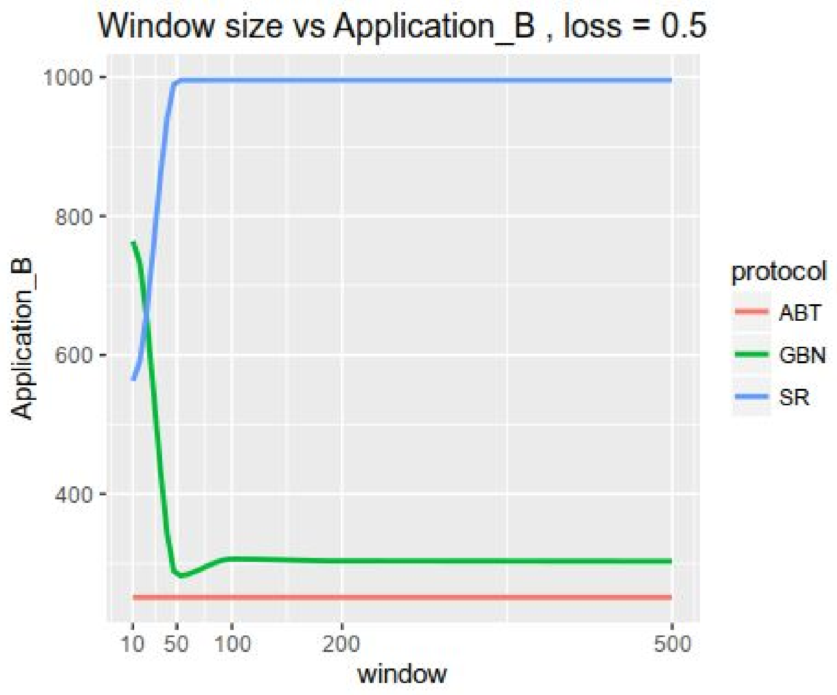


Following graph compares the Transport_A as loss increases, as expected, in GBN # of messages transmitted increases with loss, which affects its throughput as seen in previous graphs.



Following two graphs compare Application_B with Loss and window size. As expected the performance of SR in delivering messages at B is high as loss increases.


Application_B vs Loss
Application_B
Loss
protocol
ABT
GBN
SR


Window size vs Application_B , loss = 0.5
Application_B
window
protocol
ABT
GBN
SR