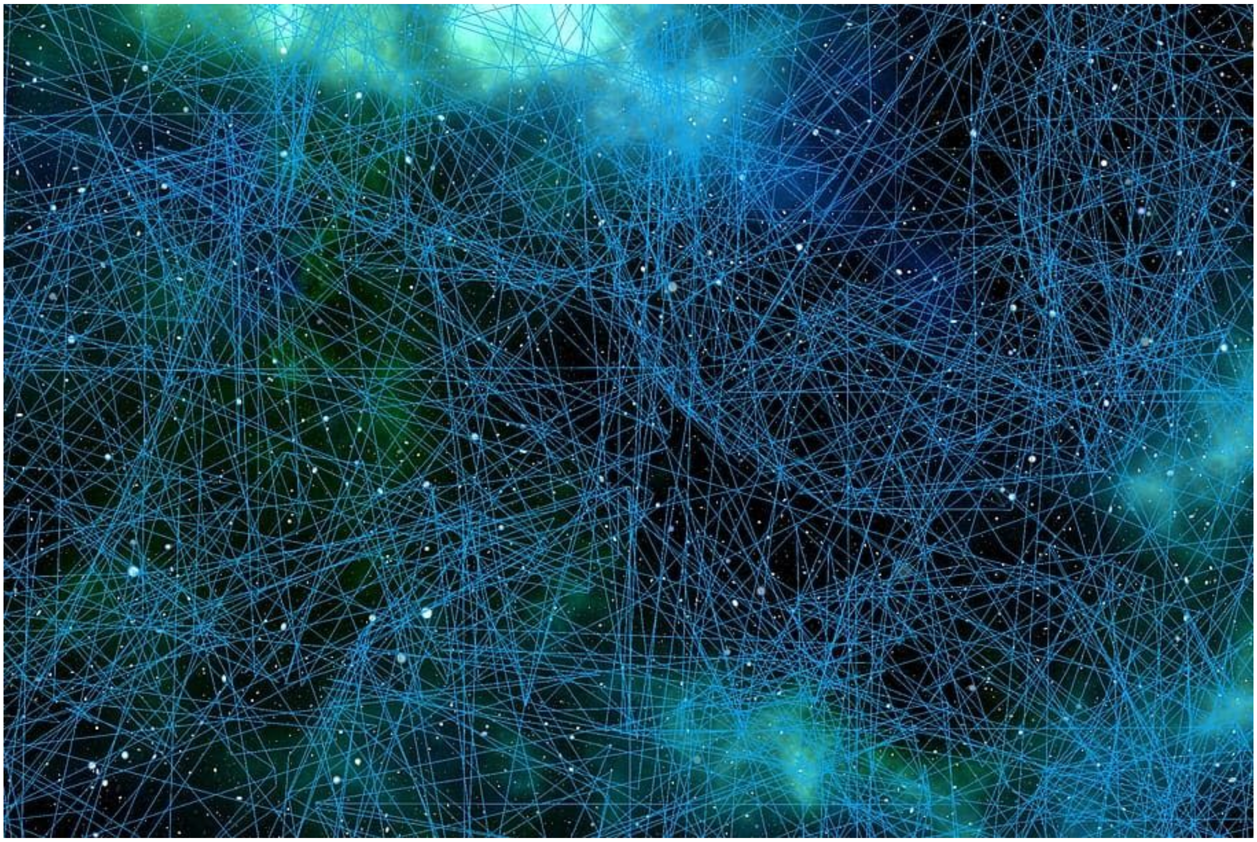
# ***Analysis_Assignment2***



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

Author - Gursimran Singh

UBITName - gursimr2

Person Number - 50320441

# ***Contents***

## Brief description of the timeout scheme.

As mentioned in the Project description, the average time taken by one packet to arrive to the other side is 5 units time , given that there are no other messages in the medium. So, the average RTT for a packet will be 10 units time. Given that, I am also taking a cushion time of additional 5 units in ABT to allow some delayed acknowledgements. Similarly, In GobackN and Selective repeat, I have increased the cushion time to additional 20 units and 10 units respectively, because these protocols pipeline multiple packets into the medium, which can increase congestion and delay.

## Brief description of the implemented multiple software timers in SR using a single hardware timer.

I am approaching this problem by using the time difference between consecutive packets sent, and starting the timer for that time duration.

Start the timer for the first packet sent with increment (RTT + some additional cushion time units). I am also storing the send time for each packet and also maintaining a queue of all the packets sent.

If timer interrupt occurs -

Pop the value from the queue and push the value at the end of the queue(value here represents seq no of the packet). A will resend the packet and start the timer with the following value-

*increment - (get_sim_time() - sendtime[queue.front()])*

If A receives an acknowledgement, A will stop the timer, it will pop all elements from the queue whose ack has been received and then start a timer by calculating value as described above. Note that the front of the queue will always have the element with smallest starttime or whose timer will expire first.

*increment* is the delay we want to give to our packet

*get_sim_time* will give the current sim time

*sendtime[queue.front()]* - It will give the send time of the packet whose timer will expires first

For example, If our packet was sent at time 20 and current sim time is 25 and increment(delay) is 10, then the calculation will give -

10 - (25-20) = 5

Another Example - If you have a timeout of 5 units and send packets p1, p2, p3, at t1=2, t2=5, t3=6. How do you set the timer at t1? What happens when the timer expires? What happens if an ACK arrives for p1 before the timer expires?

Timer will start at t1=2 for 5 units time

We have also stored the send time of p1, p2 and p3, and put them in queue in this order- p1,p2,p3

If timer expires, i.e. it's 7, then p1 will be retransmitted, queue will be p2,p3,p1 and the timeout will occur after 5-(7-5) = 3, i.e. 10 units and similarly next timeout will occur at 5-(9-6)=2 at 11

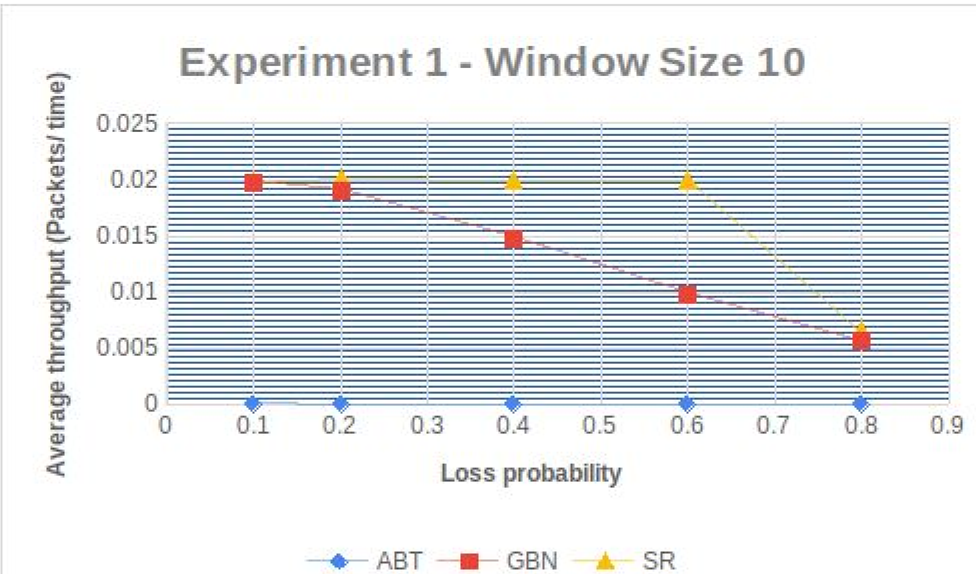
If we receive ack for p1, it is popped from the queue, and timer is stopped and its restarted with new timer value and the queue will be p2,p3.

*In the code*, I am starting the timer whenever *nextseqnum == sendbase for 20 time units,* then if interrupt occurs I am resending the packet and calculating the interval for next timer cycle. If a acknowledgement is received, I stop the timer whenever *base==nextseqnum*, i.e., there are no elements in the queue (waiting for acknowledgement).

# ***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.

## Window Size 10



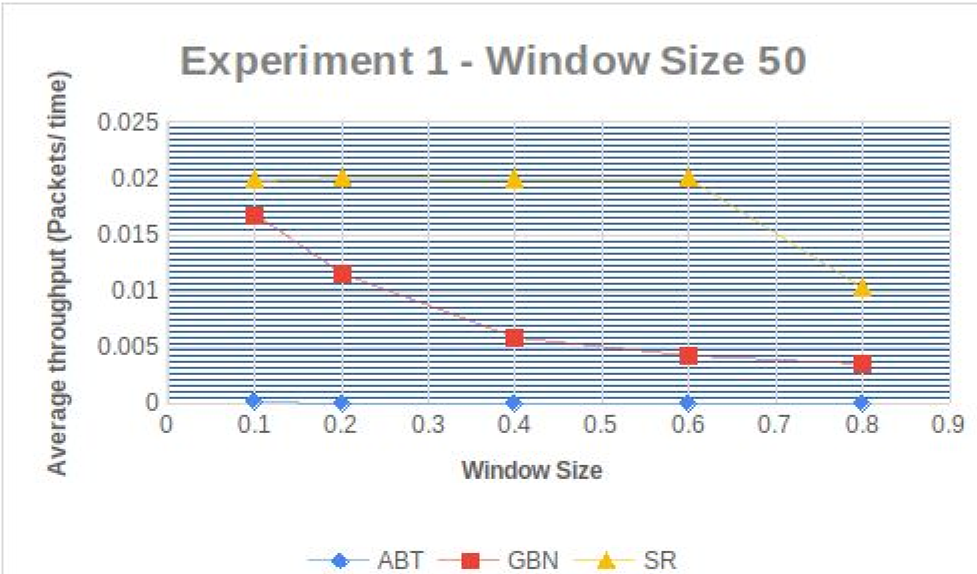

### Observations

As the loss probability increases, the throughput of GoBackN and Selective Repeat decreases whereas ABT is giving the same throughput. Also, Selective repeat has the highest average throughput, but its performance also decreases with high loss probability.

### Expectations

The results were expected to be similar, but throughput expectation from selective repeat was slightly higher than observed.

## Window Size 50




### Observations

This graph has a similar pattern as the previous one, but with more clear separations between GoBackN and Selective Repeat's throughput performance. ABT is still consistent, but Selective Repeat is the clear winner with better overall throughput performance. This experiment shows that as the window size increases, GoBack N performance degrades.

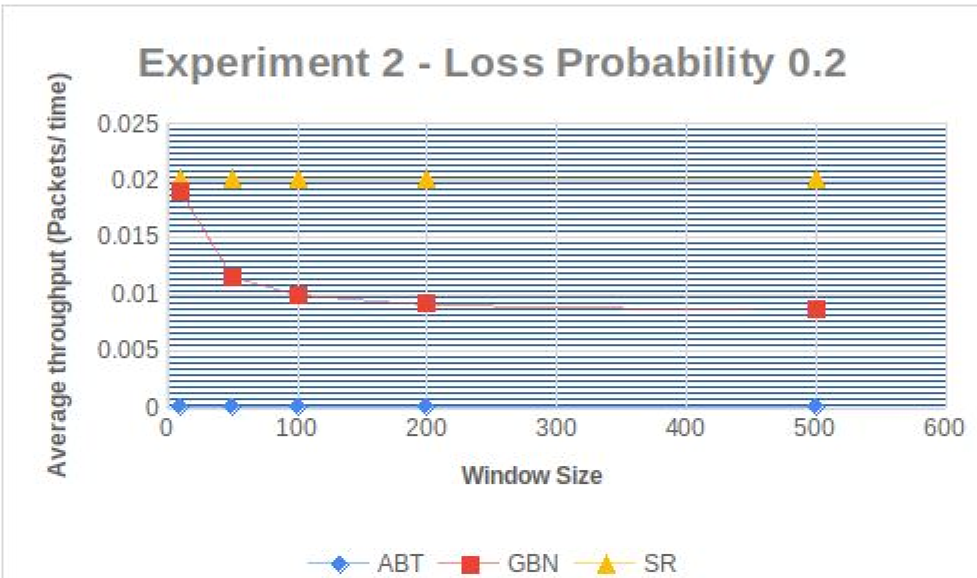
### Expectations

This experiment resulted as expected as selective repeat turned out to have the best throughput and GoBackN faces some throughput degradation.

# *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.

## Loss Probability 0.2



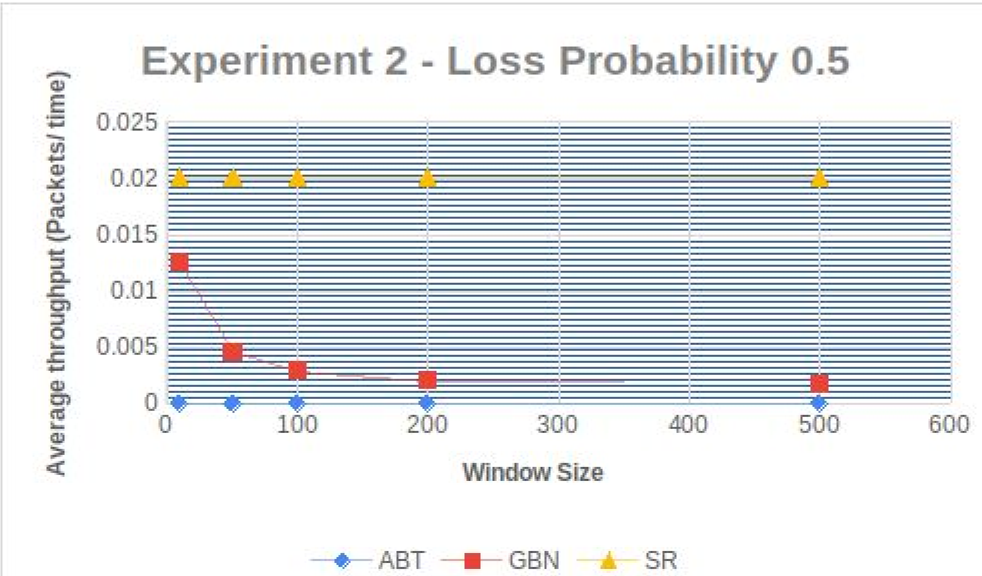

#### Observations

This experiment highlights the previous experiments results and shows the performance degradation of GoBackN as the window size increases even with low loss.

#### Expectations

Results are as expected for ABT and GBN , but the performance of selected repeat was expected to slightly increase.

## Loss Probability 0.5



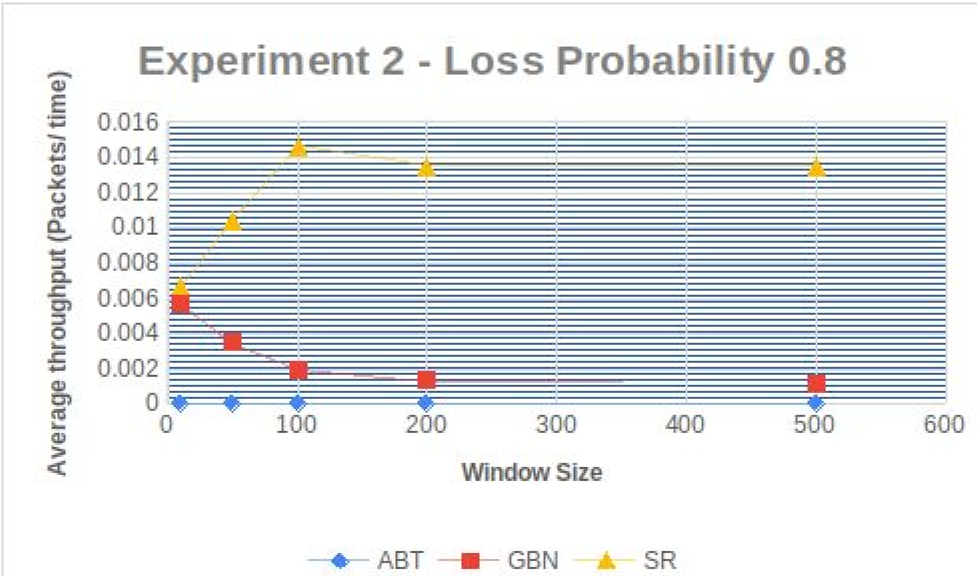

### Observations

This experiment highlights the previous experiments results and shows the performance degradation of GoBackN as the window size increases and as loss increases, it degrades further..

### Expectations

Results are as expected for ABT and GBN , but the performance of selected repeat was expected to slightly increase.

## Loss Probability 0.8




### Observations

This experiment highlights the previous experiments results and shows the performance degradation of GoBackN as the window size increases and as loss increases, it degrades further. This time, we also noticed slight increase in performance of Selective Repeat, which was expected.

### Expectations

Results are as expected for ABT and GBN , as well as for Selective Repeat, because it's throughput has increased with increased window size.

# Conclusion

In these set of experiments we observed that the Average throughput for Selective Repeat Protocol is maximum among all 3 protocols and the performance of GoBackN degrades with increasing window size. The ABT has no affect of change in window size as expected and all 3 protocols face through decrease with high loss in most cases.