CSE 4/589 - MODERN NETWORKING CONCEPTS
PROGRAMMING ASSIGNMENT 2

➢ REPORT

➢ Academic integrity statement:

I have read and understood the course academic integrity policy located under this link: "http://www.cse.buffalo.edu/faculty/dimitrio/courses/cse4589_f17/index.html#integrity"

➢ Time-out Scheme:

ABT:

- ❏ For alternating bit protocol, I have used a constant time out of 8 time units for both the window sizes because it was delivering the desired results.

GBN:

- ❏ For GBN, I used different time-out values for window sizes 10 and 50 depending upon running multiple experiments. I chose the values that were producing desired amount of throughput.

SR:

- ❏ For Selective repeat, I observed that the performance especially throughput is highly dependent on the Time-out values. I wanted to calculate the expected loss using number of packets sent vs. number of packets received at a particular time interval and vary the timeout depending upon it and depending on the experiments, the throughput has changed consistently with changing timeout for different loss rates.

➢ Implementation of multiple software timers using single hardware timer:

- I created a vector to push the details of the packet that is attempting to start the timer.
- The first packet starts the timer with a value equal to the timeout value I originally set.
- For the rest of the packets, I calculated the duration which has passed from the time I originally wanted the timer corresponding to each packet to run.
- // Duration = present_time - beginnning_time
- Present_time is the local simulator time and beginning time is the time at which the packet had to start its own timer
- So, after the timer corresponding to the previous packet is stopped, the pointer moves to the next element in the queue and the timer is started with a timeout equal to the difference of original time out and duration

6.1 PERFORMANCE COMPARISON:

ALTERNATING BIT PROTOCOL:

From what I understood both theoretically and practically about ABT, it is the most inefficient of all the protocols under consideration. Because of its stop and wait nature there is a lot of necessity to queue outgoing messages which results in an overall delay. As the loss and corruption values increase, its stop and wait nature gets extremely tough to handle and the throughput decreases unpleasantly and I think this difficulty is one of the main reasons for why it is not used in a day-to-day real-world scenario.

GO BACK N PROTOCOL:

Observations for GBN:

GBN as a protocol has an implementation that is definitely better than that of ABT because of the betterment from the stop and wait nature of ABT in GBN. There is data - replication on the sender side in order to resend data when lost, and also according to implementation instead of the refuse data in FSM, we have an option to buffer data when the value of next sequence number is greater than or equal to the sum of the base and window size. Also an effective improvement in the GBN would be implementing an option to buffer acknowledgements on the receiving end and order them and send the most recent acknowledgement to the sender because this can be done with receiver not having to discard out-of-order acknowledgements. The concept of cumulative acknowledgement helps reasonably in handling sending and receiving of data.

For the implementation, I used two queues one for data whose next sequence number is less than the sum of the window size and base(replication queue - the name of the variable is referenceQueue) and the other queue to buffer data that is otherwise refused according to FSM(amessageQueue). Whenever I am popping messages from the first queue, I am checking for space and filling them with the messages from second queue. I think the smaller window size has a better performance metric because of the confinement on the number of messages being sent, these conclusions may vary for higher number of messages under different conditions.

For results, I think at a higher loss rate a smaller window size would perform better to avoid the pain of resending the whole queue because of the frequent data loss. So having a window size that can change according to loss values would help more.

SELECTIVE REPEAT PROTOCOL:

For selective repeat, I observed that the throughput is highly dependent on the loss of the packets and also the timeout time. For higher loss rates, it is better to consider smaller timeouts (ranging from 5 - 10) and for lower loss rates it is advisable to consider bigger timeouts for the protocol to run smoothly.

For the implementation, I used two queues mainly, one to store the data generated by the simulator and other to store the details of the packet that corresponds to its own logical timer to run.

Experiment 1:

Graph #1:

Window size:10

X - Axis : Loss Probability (0.1,0.2,0.4,0.6,0.8)

Y - Axis: Throughput (ABT, GBN, SR)

Experiment 1:

Graph #2:

Window size:50

X - Axis : Loss Probability (0.1,0.2,0.4,0.6,0.8)

Y - Axis: Throughput (ABT, GBN, SR)

From the window size 10 and 50 graphs, we can see that selective repeat performs reasonably better for a bigger window than on a smaller window, i.e.., the throughput difference is more for selective repeat, it is less for GBN because the whole window has to be transmitted once there is a timeout.

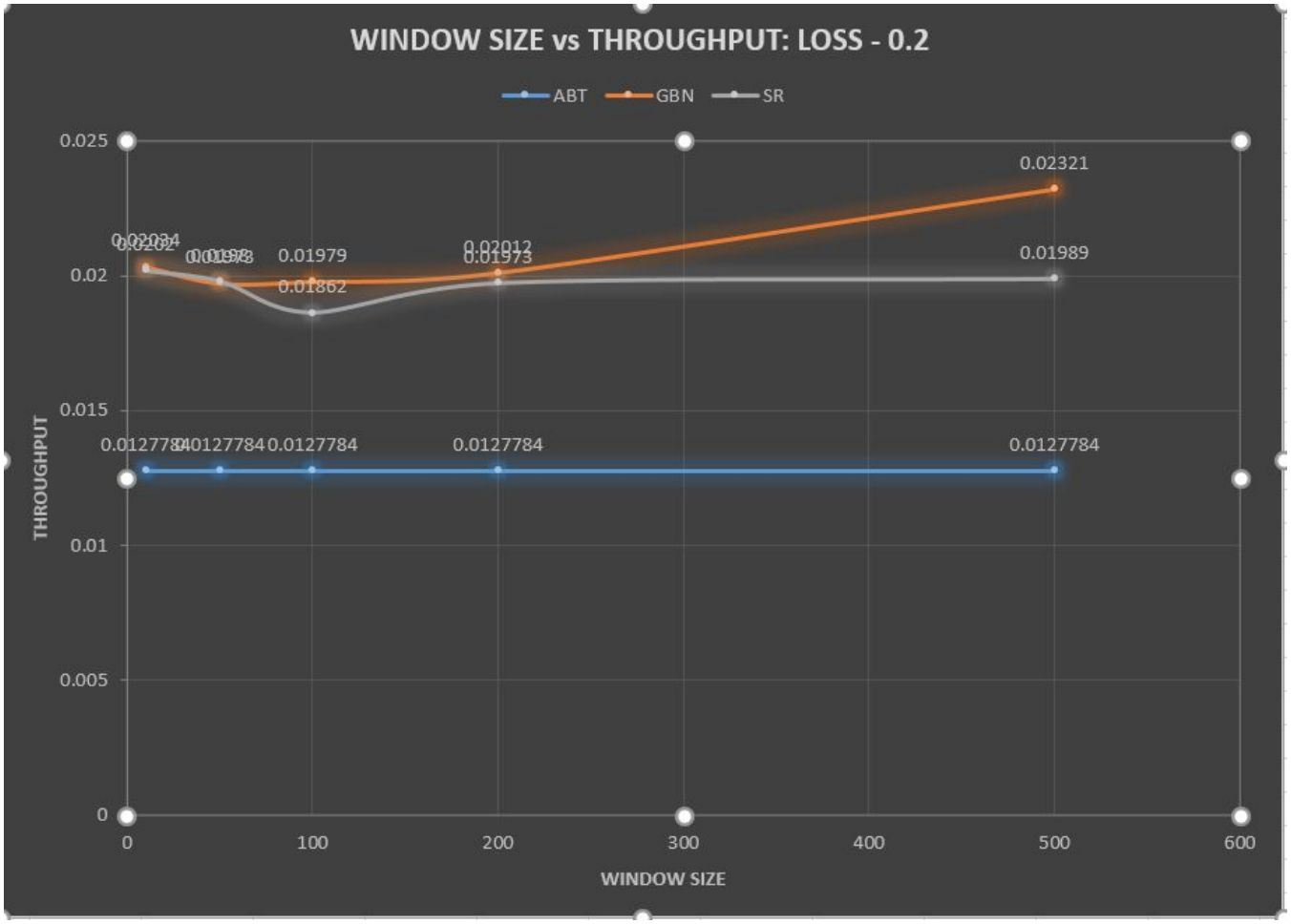
Experiment 2:

Graph #1:

Loss Probability: 0.2

X - Axis : Window sizes (10,20,50,100,500)

Y - Axis: Throughput (ABT, GBN, SR)



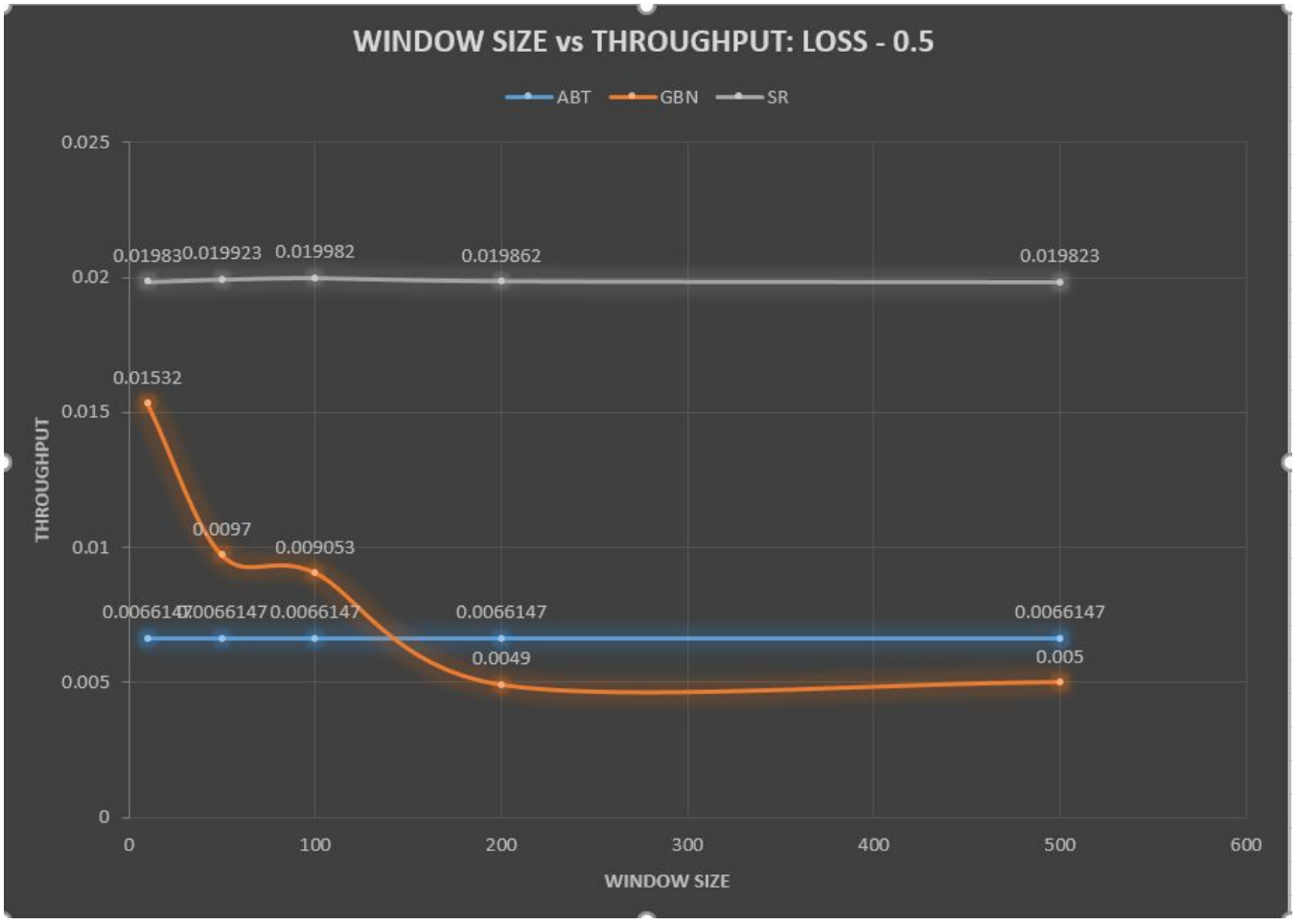
Experiment 2:

Graph #2:

Loss Probability: 0.5

X - Axis : Window sizes (10,20,50,100,500)

Y - Axis: Throughput (ABT, GBN, SR)



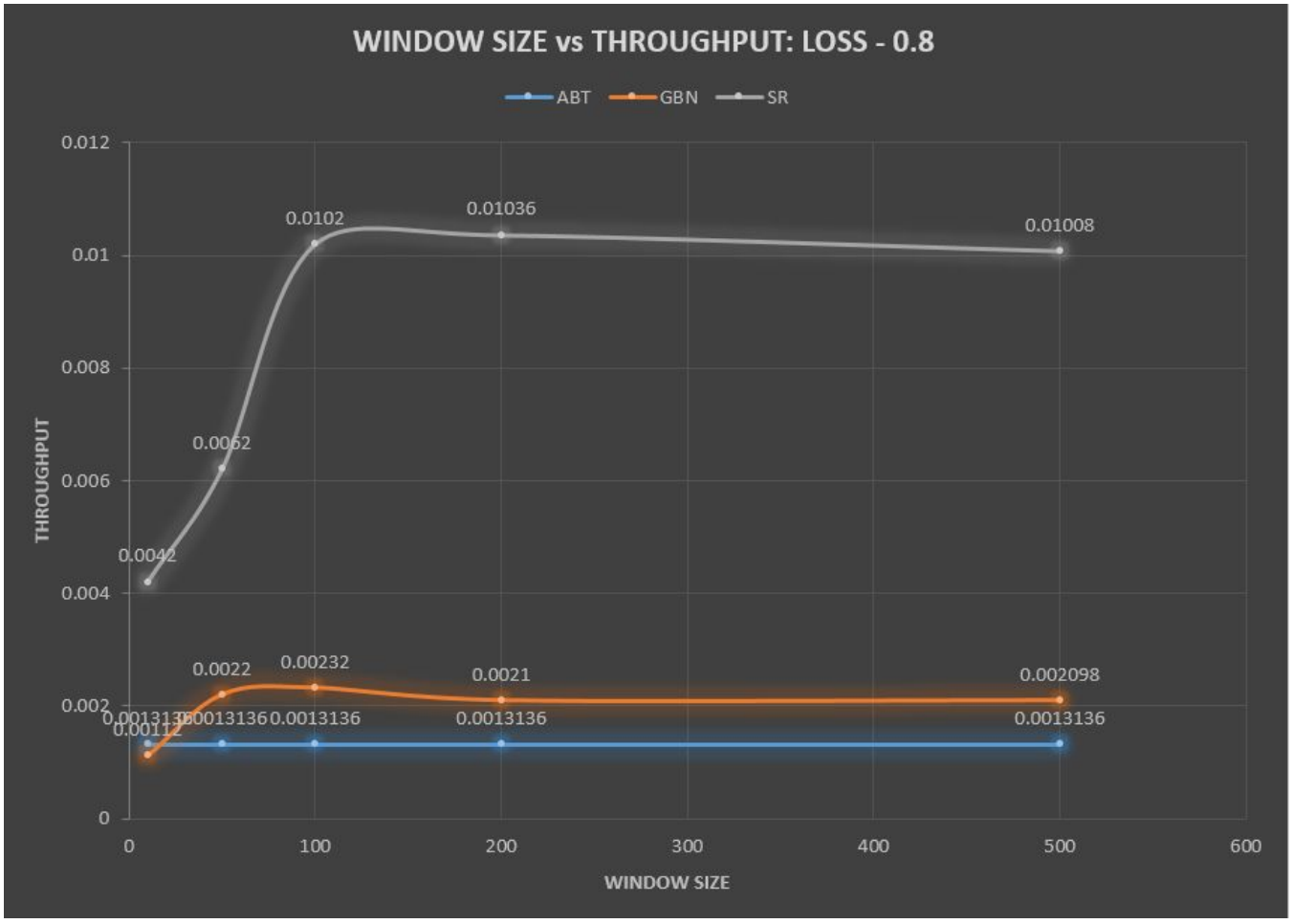
Experiment 2:

Graph #3:

Loss Probability: 0.8

X - Axis : Window sizes (10,20,50,100,500)

Y - Axis: Throughput (ABT, GBN, SR)



Selective repeat performs better for high loss rate because of the same reason that only one packet has to be retransmitted in the case of a timeout, and the number of packets to be transferred in the case of a high loss rate is going to be very high.