RELIABLE DATA TRANSFER PROTOCOLS

NAME: Vaishnavi Rajendran

I have read and understood the course academic integrity policy.

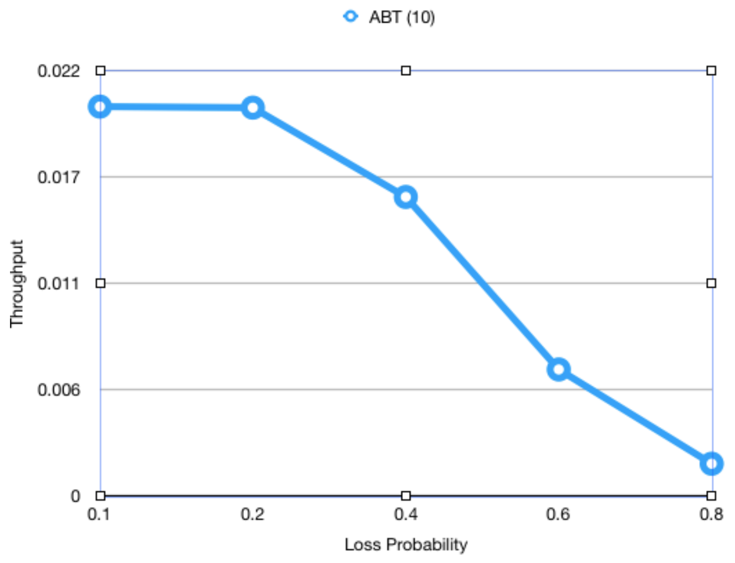
CHAPTER 1
IMPLEMENTATION AND TIMEOUT SCHEME

ALTERNATING BIT PROTOCOL:

- It is a connection-less protocol, where the window size is 1 and the sequence numbers are altered between 0 and 1 (thus it's name). It is also called stop and wait protocol.
- **IMPLEMENTATION:**
  - In my implementation, the sender and receiver are programmed according to the FSM diagram.
  - The sender will either send a message for sequence number (0 or 1) or wait for the acknowledgement of the sent packet.
  - If the sender is waiting for an acknowledgement, it will buffer the next incoming packet and would send it only after the acknowledgement for the previous packet is received as there can be only 1 unacknowledged packet in the pipeline at any given time.
  - The receiver checks the packet checksum and sends either an acknowledgement of the received packet sequence number or the opposite sequence number (NAK).
- **TIMEOUT SCHEME:** In ABT, a constant time has been used. After examining the throughput behavior for different timeout values.
- **THROUGHPUT ANALYSIS:** The figure shows the throughput of Alternating Bit protocol for different loss probabilities.

ABT Throughput Analysis

LOSS PROBABILITY	ABT (10)
0.1	0.0201144
0.2	0.0200599
0.4	0.0154483
0.6	0.0065401
0.8	0.001672



- **Observation:** As seen from the figure above, the throughput of ABT decreases as the loss probability is increased. This can be due to the fact that ABT will keep on retransmitting the same packet unless it is correctly received.

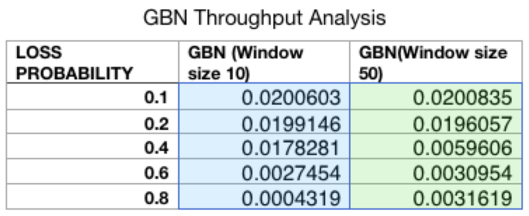
GO-BACK-N:

- Go-Back-N is a sliding window protocol where the window size N. In this the sender is allowed to transmit multiple packets (whenever available) without waiting for acknowledgement but the number of unacknowledged packets in the pipeline cannot be greater than N. The receiver only sends cumulative acknowledgements, and the sender maintains a timer for the oldest unacknowledged packet.
- **IMPLEMENTATION:**
  - The implementation is done like the FSM of GBN, except that the packets are buffered instead of being discarded.
  - Sender:
    1. Packet is sent if it lies within the window size, otherwise it is buffered.
    2. If timeout occurs, all the packets from the current base till the next available sequence number or till the window size (from current base) are sent.

3. In GBN, we get cumulative acknowledgements, thus if we get an acknowledgement, we check if it is less than the current updated base and update the base with received acknowledgement number +1.

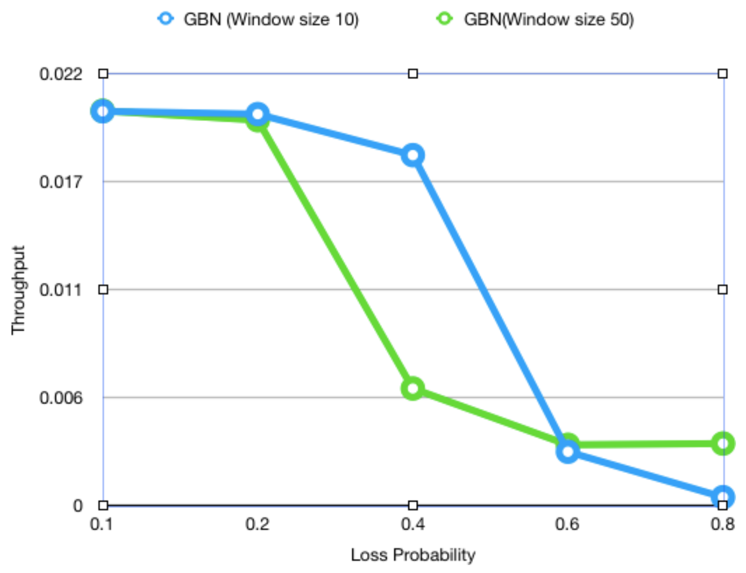
- Receiver:
  1. When the packet arrives at B, the sequence number and checksum of the packet is checked and an acknowledgement is sent. Acknowledgement is sent only for the correctly received packet with the highest sequence number. (Cumulative ack).
- In case of loss or corruption, timerinterrupt is called to retransmit the unacknowledged packets.

- **TIMEOUT SCHEME:** In GBN, I have used an adaptive timer according to the window size. The timer value was determined after a lot of experiments with different timeout values to find the best possible timeout value.
  - If the window size is <= 50, The timeout value is 40.
  - Otherwise the timeout value is 50.
- **THROUGHPUT ANALYSIS:** The figure shows the throughput of Go-Back-N protocol for different loss probabilities.



GBN Throughput Analysis

LOSS PROBABILITY	GBN (Window size 10)	GBN(Window size 50)
0.1	0.0200603	0.0200835
0.2	0.0199146	0.0196057
0.4	0.0178281	0.0059606
0.6	0.0027454	0.0030954
0.8	0.0004319	0.0031619

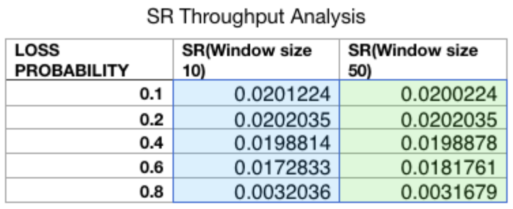


  - **Observation:** As seen from the above figure, the throughput of GBN is better when the window size is 10. In case of lower loss probabilities, the throughput of GBN is not affected by window size.

SELECTIVE REPEAT (SR):

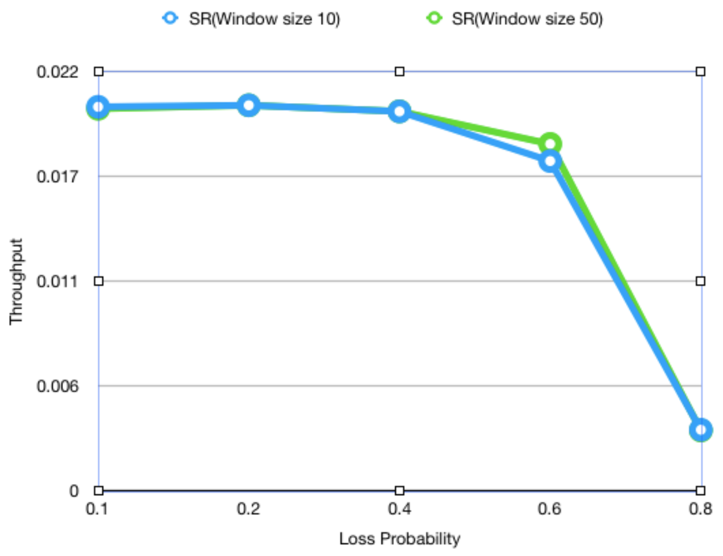
- Selective repeat is also a sliding window protocol,
- In GBN, even a single packet error can lead to a lot of retransmissions. This is avoided in selective repeat.
- **IMPLEMENTATION:**
    - Sender:
        1. Packet is sent if the next available sequence number is in the window and timer is started if the current sequence number is equal to the base.
        2. If the acknowledgement number of the received packet from B lies in the window, the packet is marked acknowledge and the base is advanced to the next acknowledged sequence.
        3. In case of SR, the whole timer implementation is taken care with each subsequent timeout. Timer is started according to the timeout scheme discussed below.
    - Receiver:
        1. The sequence number of the packet is checked if it lies within the window starting from receive base and is not corrupted. Then, if it is equal to the expected sequence number, it is sent to the application layer, otherwise it is stored in the receive buffer and an acknowledgement is sent for the receiver packet.
        2. The receive base is increase to the next expected sequence number.
        3. Just like in GBN, for loss and corruption scenarios, the timer interrupt takes care of retransmission. In SR, packets are selectively retransmitted on the basis of timeout.
- **TIMEOUT SCHEME:**
    - For Selective repeat, we maintain timers for each individual packet (unlike in GBN where the timer is only associated with the current window base), and we have only a single hardware timer which has been used to mimic the operation of multiple logical timers.
    - The logic used is described below:
        1. Each packet has an associated start_time and a relative_timeout associated to it. The start time is the time when the packet is sent, and the relative_timeout is the difference between the start time of the current packet P and the start time of base packet for the Pth packet.
        2. The timer is initially started for the base packet of the window, and then the next time at timeout, timer will be started for the packet which has not been acknowledged and has the smallest timeout.
        3. In case of retransmissions, the start_time of the packet P is changed to ensure that the next timer should be started for the packet which was sent before packet P.
        4. This timing scheme also ensures that packets are selectively repeated, thus better than Go-Back-N where at any timeout, all the unacknowledged packets in the window are resent.

- **THROUGHPUT ANALYSIS:** The figure shows the throughput of Go-Back-N protocol for different loss probabilities.



SR Throughput Analysis

LOSS PROBABILITY	SR(Window size 10)	SR(Window size 50)
0.1	0.0201224	0.0200224
0.2	0.0202035	0.0202035
0.4	0.0198814	0.0198878
0.6	0.0172833	0.0181761
0.8	0.0032036	0.0031679



  - **Observation:** The throughput of SR is almost comparable for both window sizes and it is not much affected by the loss probability. The performance of SR is almost comparable for all loss probabilities from 0.1 to 0.6 and then it decreases when the loss probability is 0.8 and corruption is 0.2.

CHAPTER 2
OBSERVATION AND ANALYSIS

EXPERIMENT 1

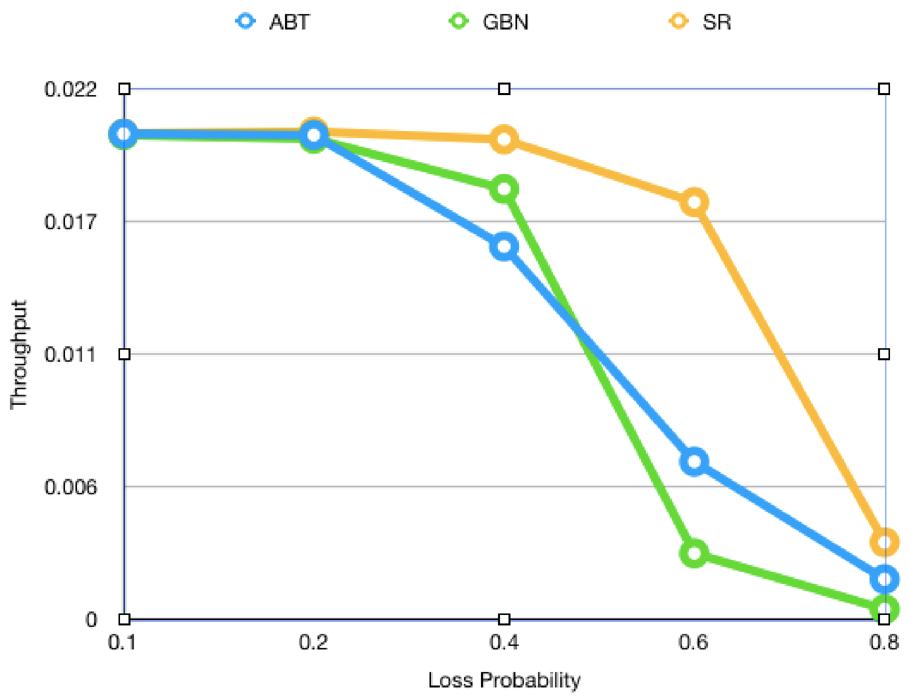
Aim: To compare the 3 protocols' throughputs at the application layer of receiver B with loss probabilities: {0.1, 0.2, 0.4, 0.6, 0.8} and 2 window sizes: {10, 50} for the Go-Back-N version and the Selective-Repeat Version.

a) For Window Size 10:

The observed results are shown below:

Exp1: Window Size 10

LOSS PROBABILITY	ABT	GBN	SR
0.1	0.0201144	0.0200603	0.0201224
0.2	0.0200599	0.0199146	0.0202035
0.4	0.0154483	0.0178281	0.0198814
0.6	0.0065401	0.0027454	0.0172833
0.8	0.001672	0.0004319	0.0032036



Observations:

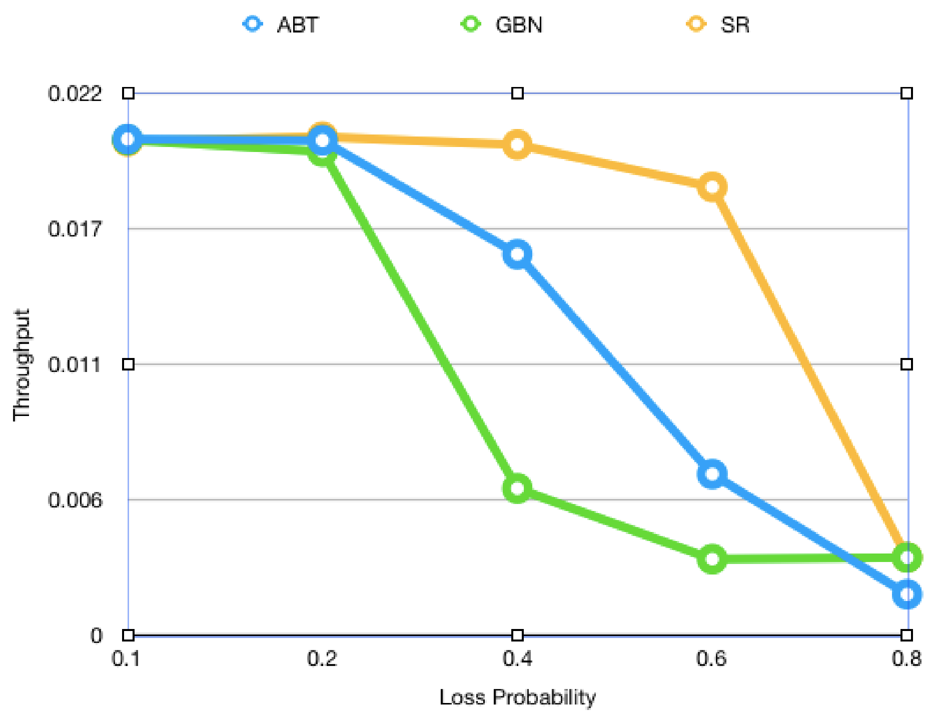
1. All the three protocols have almost the same throughput when loss probability is 0.1 or 0.2, and GBN is better than ABT for loss probability 0.4 but less efficient than SR.
2. At higher loss probabilities, ABT is performing better than GBN. This can be due to the fact that in GBN, as the loss probability increases, the number of retransmissions also increase exponentially since the whole window of unacknowledged packets are resent.
3. We can conclude from the above graph that for window size 10, SR has the best performance with respect to all the loss probabilities.

b) For Window Size 50:

The observed results are shown below:

Exp 1: Window Size 50

LOSS PROBABILITY	ABT	GBN	SR
0.1	0.0201144	0.0200835	0.0200224
0.2	0.0200599	0.0196057	0.0202035
0.4	0.0154483	0.0059606	0.0198878
0.6	0.0065401	0.0030954	0.0181761
0.8	0.001672	0.0031619	0.0031679



Observations:

1. For window size 50 also the initial throughput for all the three protocols for loss probability 0.1 and 0.2 is almost same.

2. As the loss probability increases, ABT has better throughput than GBN. In the above case, window size was 10, thus the number of retransmissions in a loss scenario would be max 10, and in this case the window size is 50, which means that whenever loss occurs, there is a possibility that the number of unacknowledged packets which would have to be resent is more. Even if one loss occurs, all the packets are resent as there is no receive buffer like in the case of SR. Thus, this can be a reason of a decrease in throughput as the time taken to send 1000 messages from A will be more than that taken by ABT.
3. SR has the best throughput for window size 50 as well.

Conclusions:

From experiment 1, we can conclude that:

1. SR is the most efficient of the three protocols.
2. GBN is efficient when the window size is less and is better than ABT for window size 10 with lower loss probabilities.
3. The performance of GBN decreases with an increase in window size and loss probability.

EXPERIMENT 2:

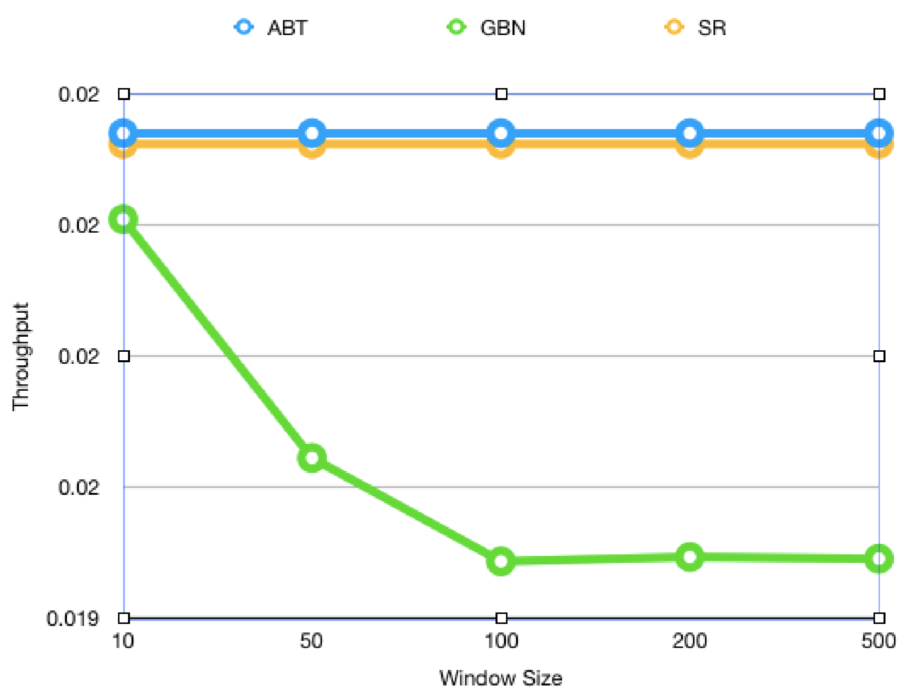
Aim: To compare the 3 protocols' throughputs at the application layer of receiver B with window sizes: {10, 50, 100, 200, 500} for GBN and SR, and 3 loss probabilities: {0.2, 0.5, 0.8} for all 3 protocols.

a) For Loss Probability 0.2

The observed results are shown below:

Exp 2: Loss Probability 0.2

WINDOW SIZE	ABT	GBN	SR
10	0.0202243	0.0200603	0.0202035
50	0.0202243	0.0196057	0.0202035
100	0.0202243	0.0194088	0.0202035
200	0.0202243	0.019417	0.0202035
500	0.0202243	0.0194137	0.0202035



Observations:

1. Efficiency of SR and ABT is almost similar when the loss probability is 0.2 and as we can see, it is unaffected by the change in window size.
2. In case of GBN, as the window size increases, the throughput at B also decreases.

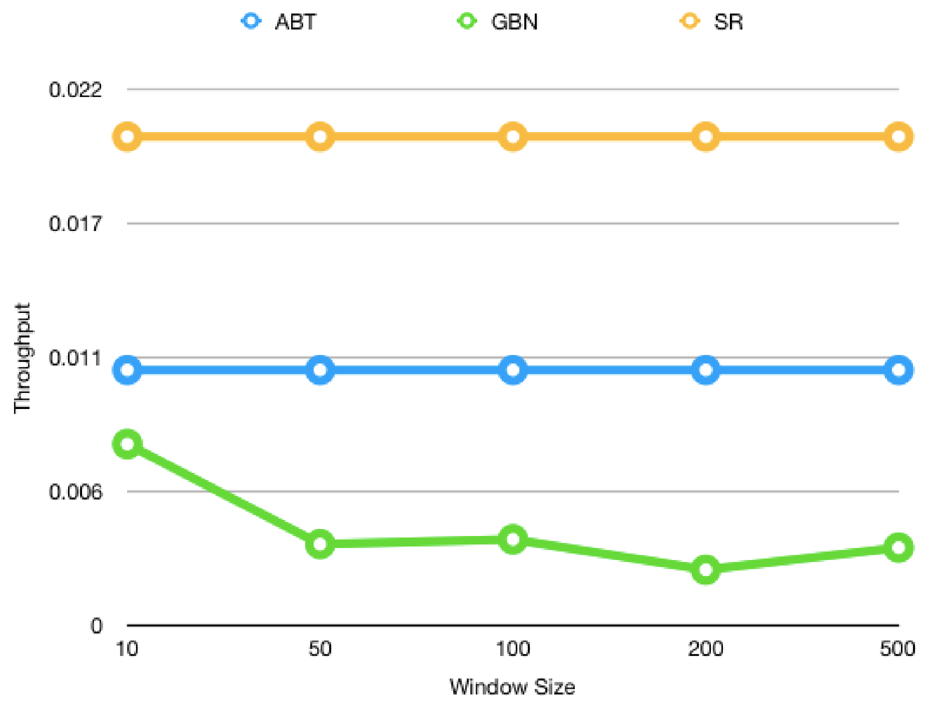
3. Thus, when loss probability is 0.2, SR and ABT both perform good.

b) For loss probability 0.5

The observed results are shown below:

Exp 2: Loss Probability 0.5

WINDOW SIZE	ABT	GBN	SR
10	0.0104798	0.0074471	0.0200353
50	0.0104798	0.0033476	0.0200353
100	0.0104798	0.0035431	0.0200353
200	0.0104798	0.0022945	0.0200353
500	0.0104798	0.003204857	0.0200353



Observations:

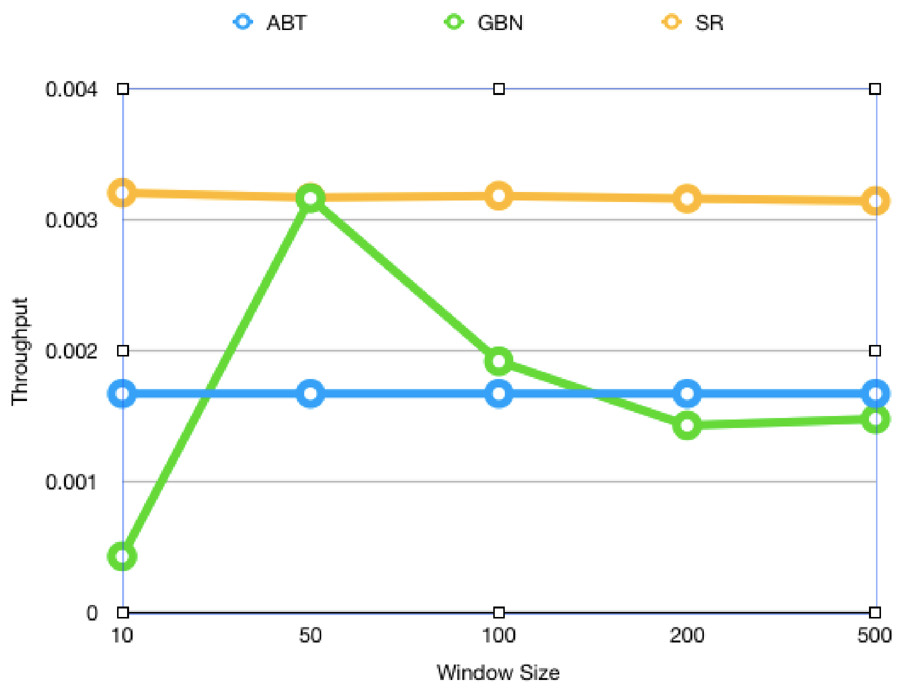
1. The efficiency of SR and ABT is not affected by the change in window size.
2. The change in window size does not affect the throughput of SR, as it selectively repeats in case of loss or corruption.
3. We can clearly see that ABT performs the best out of the three protocols.
4. Throughput of GBN is maximum when the window size is 10, and then shows a decrease in throughput as the window size increases.
5. SR is the best choice of protocol when the loss probability is 0.5.

c) For Loss Probability 0.8

The observed results are shown below:

Exp 2: Loss Probability 0.8

WINDOW SIZE	ABT	GBN	SR
10	0.001672	0.0004319	0.0032036
50	0.001672	0.0031619	0.0031679
100	0.001672	0.0019186	0.0031813
200	0.001672	0.0014295	0.0031588
500	0.001672	0.001478	0.0031411



Observations:

1. For loss probability 0.8 also, the throughput measured at B for SR is the best as compared to the other two protocols.
2. The throughput for SR decreases by a very small value as the window size is increased.
3. GBN shows a better performance, when window size is 50, and is better than ABT for a few window values and less efficient than ABT for others.

Conclusions:

1. SR has the best performance in all cases.
2. The throughput for GBN decreases as the window size and loss probability both increase. To improve this, a buffer can be implemented at the receiver side like in SR.
3. ABT is efficient when the loss probability is less.