Anush Ravindra Shetty
5024 7204

Reliable Transport Protocols

I have read and understood the course academic integrity policy.

Modern Networking Concepts

Programming Assignment 2

Index:

1. Key Concepts :-

Timeout selection:-
In the selection of timeout for the protocols, I considered the time for a packet to travel from A_output to B_input. After having careful observation, the time duration for arrival of packet at B is in between 1 time unit and 10 time units. Thus for half roundtrip time the duration is at the maximum 10 time units. By this observation the time required for the packet's acknowledgment to be received is at most 20 time units. Thus the timeout value I have kept is equal to 20 time units.

Buffer Data:
In the case when multiple messages are sent from application layer in the sender, I maintain a buffer for each protocol.
ABT:- 1. For ABT, I use a queue based buffering mechanism which stores every message sent by layer 5 of sender regardless of the sender's state.
2. On every correctly received acknowledgment sends next message within this buffer.

GBN and SR:-
1. For GBN, I maintain a array of packets, every message sent from the application layer of the sender is first buffered in this array.
2. On correct receipt of acknowledgment, for every window shift, packets are taken from this array of packets and sent to the receiver.

Multiple Logical Timers for single hardware timer:-
For Selective Repeat we need to have timer for each packet which is not possible with using single hardware timer.
<u>Timer Implementation:-</u>
1. We first set the timer for each packet its absolute value of timeout. That is for 3 packets started at time 0, 1, 2 respectively we set the timeout to be 20, 21, 22 respectively in a list resembling to packet list.
2. Further when an acknowledgment of a packet is received we remove that packet from the timeouts list and add the consecutive timeout list with time remaining from the timeout of the packet acknowledged and the time when it actually received. Similarly for every packet in the buffer I apply this strategy.
3. During a timeout, the timer for all the consecutive packets except are subtracted from the current simulation time and the timed out packet is set to a timer for next timeout values.
4. Further this packet is resent with the timeout value set to it in the timers list.

Idea of Adaptive timeout:-
1. For an adaptive timeout in GBN and SR we need to maintain a timer list in the sender side and check for the transmission time values in the timer list till the window size .
2. Check for a maximum transmission time within this window.
3. Update this value as Timeout value for next transmission window.

2. Implementation Details:

This section explains the implementation of the three protocols.
Below is the explanation of seven routines which describes the key function for packet transmission.

2.1 Alternating Bit Protocol:
1. Function A_output():
a) Buffer all the incoming messages in a queue.
b) Check for the state of A whether it is currently busy(i.e. just now sent the packet) or not. If the state is free, then this message can be sent to the receiver end.
c) Change the state of A to busy in this course of duration.
d) Make the packet and send to layer3.
e) Start the timer.

2. Function B_input():
a) Check the checksum field of the packet and compare this by calculating the packet’s checksum.
b) If the sequence number is not the one expected simply send back an ACK.
c) If the sequence number is the one which is expected, send data to layer5 and then send back an ACK.

3. Function A_input():
a) Receive packet from layer3, if the packed is corrupted the drop the packet
b) If the acknowledgment number of ACK packet is the same as expected, stop the timer and send the next packet in buffer to layer3.

4. Function A_timerinterrupt():
a)If we do not receive ACK of last packet, then retransmit the same packet and start timer again

2.2 GoBack-N protocol:
1. Function A_output():
a) Buffer all the message incoming in a array of packets.
b) If the current window is full, do nothing.
c) Retrieve the message of nextseqnum to be passed.
d) Make the packet and send the packet to layer3.
e) If base is same as nextseqnum start the timer for that packet.

2. Function B_input():
a) Check for corruption by looking into the checksum of the packet, and compare this by calculating the packet’s checksum.
b) If the seqnum is same as expected seqnum, correct packet is received send the message to layer5 and send an ack to sender side.
c) Discard any out of order OR corrupted packets.

3. Function A_input():
a) Receive packet from layer3, if the packet has not corrupted and is within the window.

b) Shift the window from last received correct acknowledged number and resend the packets which are remaining in the buffer.
c) For the sent packet start the timer.

4.Function A_timerinterrupt():
a) Retransmit all the packets that are not ACKed yet.
b) Start the timer again.

2.3 Selective Repeat protocol:
1. Function A_output():
a) Buffer all the message incoming in an array of packets.
b) if sender’s current window is not full, retrieve the message of nextseqnum to be passed.
c) Make a new packet and send the the packet to layer 3.
d) Start the timer only if base is equal to sequence number. Note this packet’s timer as get_sim_time() + TIMEOUT.

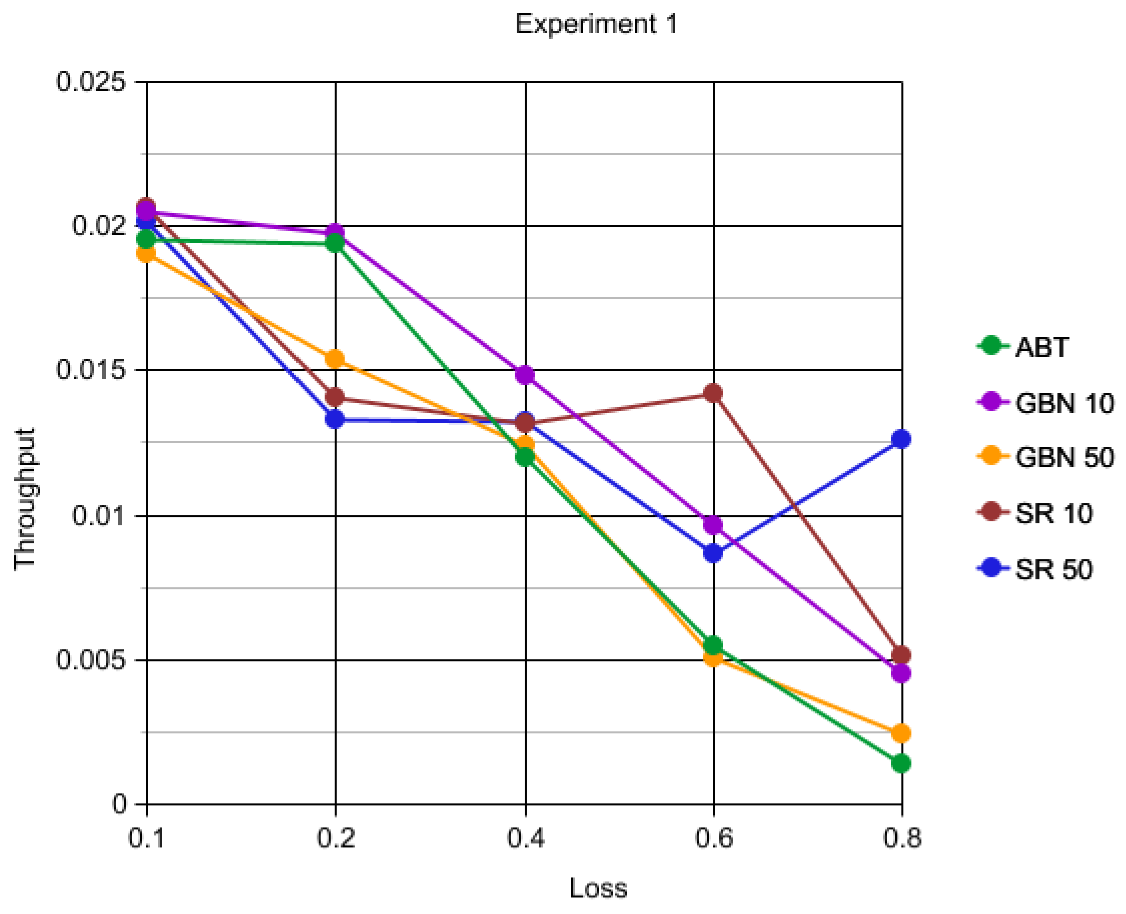
2. Function B_input():
a) Maintain a list of received packets in the receiver end.
b) If the packet is not corrupted and sequence number is within receiver’s window, make an acknowledgment packet and mark B_side state of that packet as received.
c) Send this acknowledgment to the sender side and shift the window appropriately.
d) If a packet’s sequence number is out of order we first check whether the packet has a received state in our list. If yes then send back an acknowledgment for the same packet.
3. Function A_input():
a) if the packet is not corrupted and the acknowledgment number received is within the sender’s window then stop the timer.
b) By shifting the window, send another packet.
c) Relatively update the timer of the successive packets in the buffer and wait for an acknowledgment.
d) If a packet is in buffer and was not sent or acked then send the packet to layer5. Start the timer for this packet by calculating the timeout + earlier timeout of the packet – get_sim_time().

4. Function A_timerinterrupt():
a) For the packet which has timed out, update the timer and send it again to receiver side.
b) Further update the value of consecutive packet within the window.
c) Start the timer for the sent packet according to the timer logic.

3. Observation and Analysis:

3.1 Experiment 1:(Loss Probabilities : 0.1, 0.2, 0.4, 0.6, 0.8)

Observation:



I. GBN and SR protocols are almost faster than ABT as window loss rate increases.
II. Throughput decreases for all three protocols as loss rate increases.

Analysis:

ABT:-
Considering the ABT protocol sends packet only after the acknowledgment for previously sent packet. It has to retransmit every lost packet. From the above graph if we consider the loss rate of 0.1 and loss rate of 0.8, a significant decrease in the throughput can be seen. This means that the protocol retransmits almost more than double rate of the transmitted packets.
GBN:-

Now considering GBN which has a window size of say N. If we transmit N packets, we can send them at once and wait for acknowledgment of all the packets within this window. Thus the time required would be RTT + N*Rate of (last packet sent – first packet sent)

SR:-

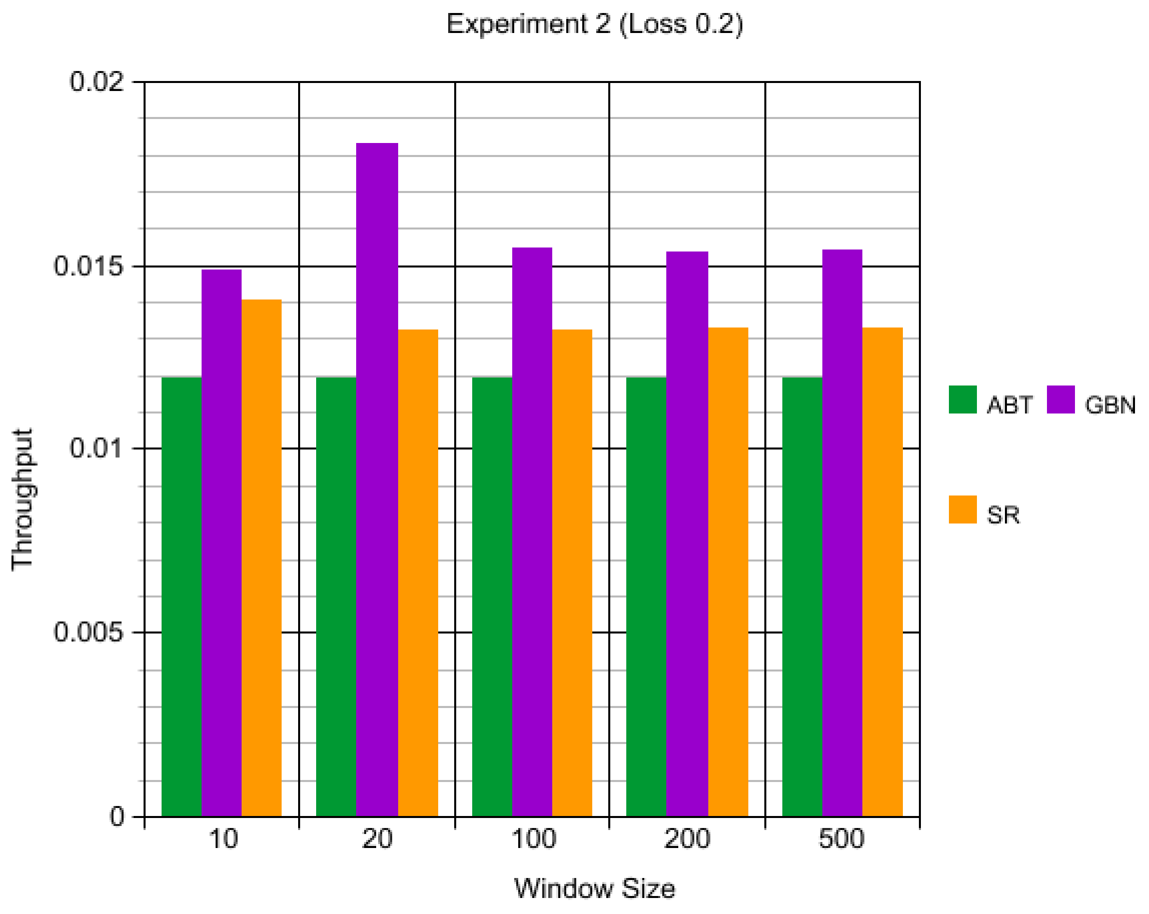
Similarly considering SR protocol of window size N. We get the same time required as that of GBN i.e. RTT + N*Rate of (last packet sent – first packet sent)

Comparison:-

1. GBN and SR are comparatively faster than ABT.
2. Also say if A packets lost rate and B packets corrupted, then probability that packets will be transmitted is (1-A)(1-B) from sender to receiver, thus as loss rate increases, this transmission probability decreases.
3. Hence in the above graph, the throughput rate decreases as the loss rate increases.

3.2 <u>Experiment 2:-</u>

3.2.1 Experiment 2.1 (Loss 0.2)



Observation:

I) Under conditions of low loss, GBN and SR protocol get maximum throughput even with a small window size.
II) Variations among the window size doesnt make much difference for less loss.

Analysis:

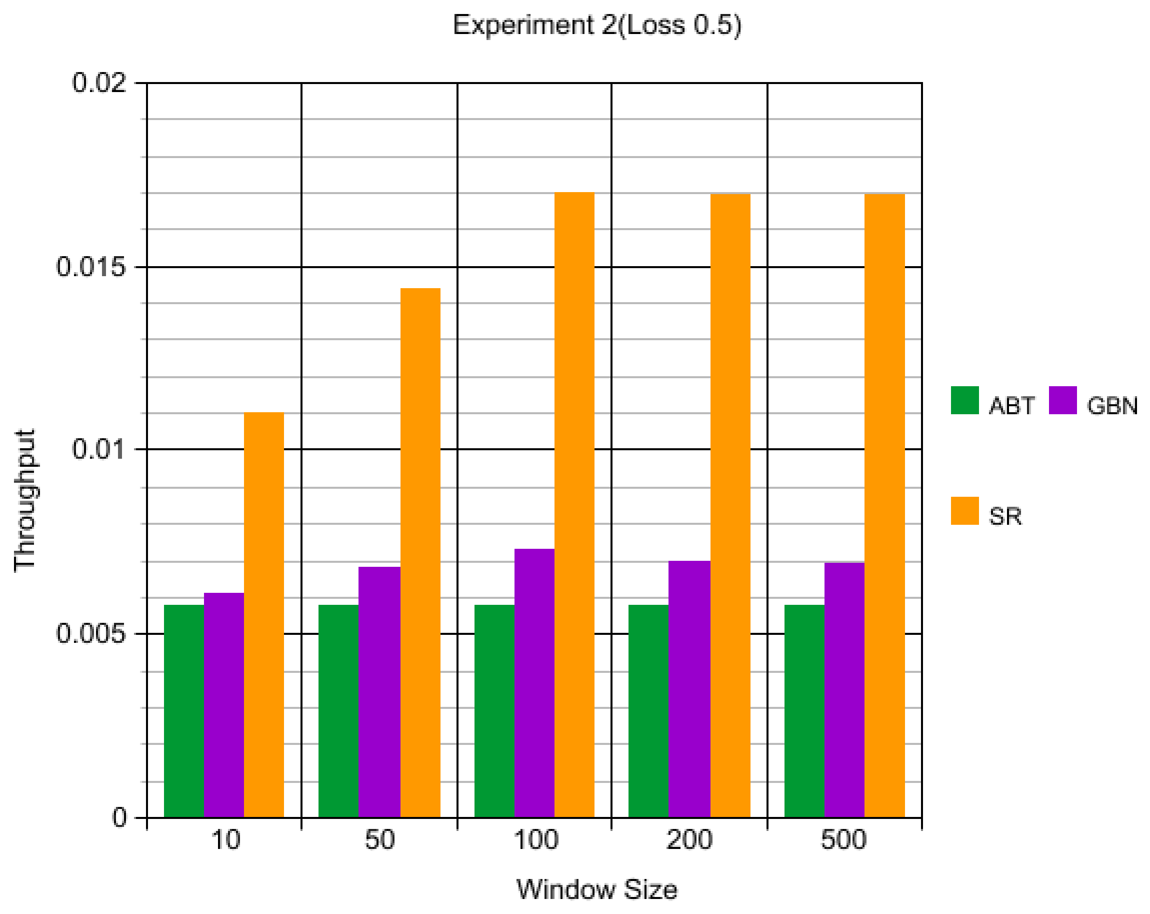
ABT:-
1) In this as window size is 1, we can say that from the above graph throughput is inversely proportional to product of loss rate and corruption rate.
2) Also by comparing all the values, this is the maximum throughput with loss rate of 0.2

GBN:-
1)With the window size of 10 and a timeout of 20, as discussed earlier for sending data across the layer3, end to end transmission would take around 10*20 =500 time units. Thus if simulator runs with a loss rate of 0.2(0.8 of packets are not lost) and corruption rate of 0.2(0.8 of packets not corrupted), then minimum of 0.64 times the packets sent should be arrived at least on the receiver side(inclusive of duplicate ACKs or retransmission).

SR:-
Similarly with the case of SR, the graph doesn’t show much improvement with increase in window size.

Thus the throughput will not increase given a limited message generation speed by senders application layer.

3.2.2 Experiment 2(Loss 0.5)
Observation:



<u>Analysis:-</u>

ABT:

1) In ABT, as window size is 1 throughput is relatively low as compared to the throughput with loss 0.2

2) Also as window size is 1, for every packet lost we have to transmit twice the number of packets sent. This reduces the overall throughput of ABT.

GBN:

1) With different window sizes, the rate at which the packets are received is less than the throughput with loss 0.2.

2) Also the graph specifies that this is not the maximum throughput which can be achieved.

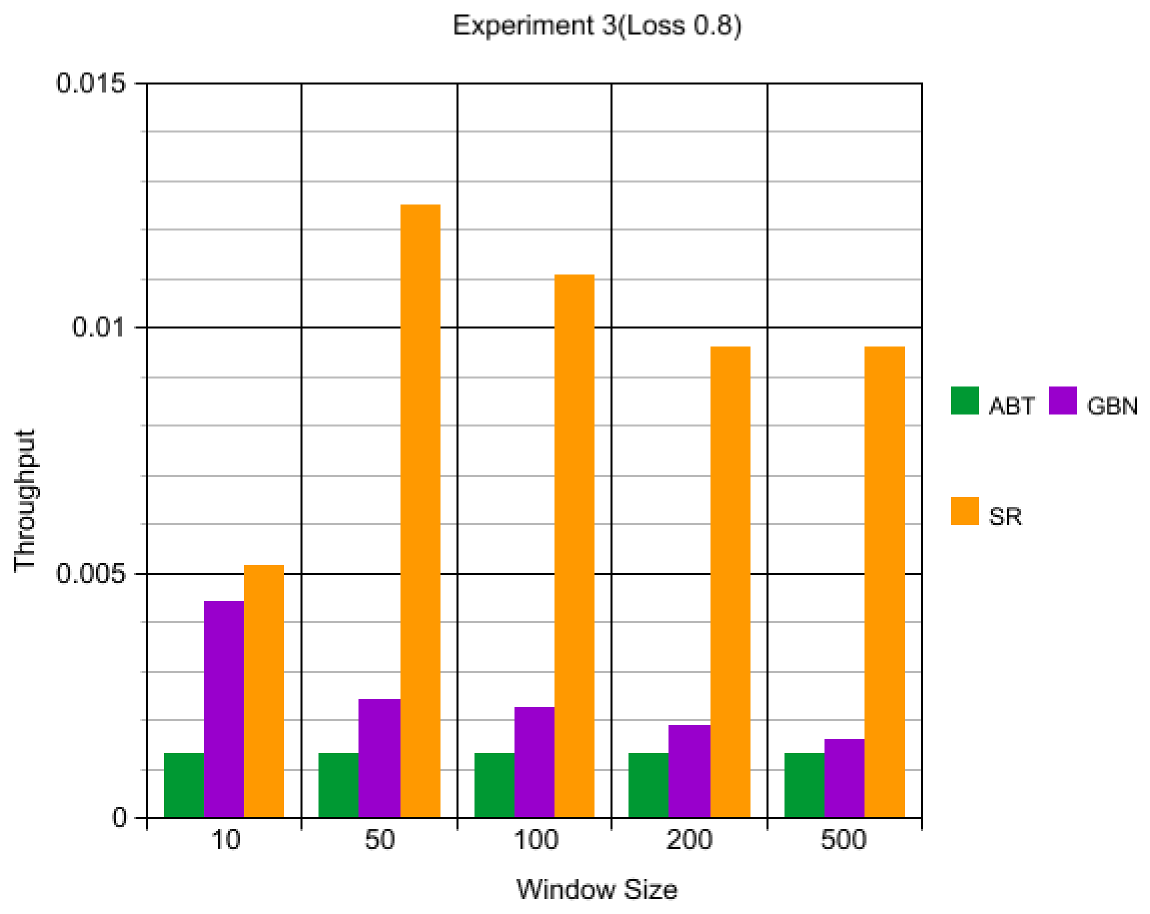
SR:-

1) In the case of SR as the window size increases, the throughput almost remains same. This maybe due to independent acknowledgement. As when the window size increases there are only retransmissions of the unacked packet within the window.

2) Also in the graph, after increase in window size, the throughput increases to maximum throughput.

Thus by increasing window size we get the maximum throughput during relatively high loss.

3.2.3 Experiment 3(Loss 0.8)
Observation:



<u>**Analysis:-**</u>
Throughput is relatively less compared to loss rate of 0.2 and 0.5
ABT:-
1) With a window size of 1, we get an extremely low throughput.
2) This is because of high loss. For every transmission of packet a minimum of (1-0.8)(1-0.2)=0.16 times the packets are sent from sender again to the receiver and similarly from the receiver to the sender as the loss and corruption can be during the sending of acknowledgment.

GBN:-
1) In the above graph, the rate of throughput decreases with increase in window. This is because retransmission occurs of all the packets from the last acknowledged packet.
2) Considering the observation in Experiment 1 the duration will be RTT+ Window size*Rate of(last sent packet – first sent packet) and thus for smaller value of window size we get a better throughput which reduces as we increase the window size.

SR:-

1) In case of SR protocol, the transmission gets better as we go further the window size.
2) This is because of single unacked packet retransmission in the event of timeout.

4. References:

1. http://www.ccs-labs.org/teaching/rn/animations/gbn_sr
2. R. Brown, "Calendar queues: A fast O(1) priority queue implementation
for the simulation event set problem," Commun. ACM, vol. 31, no. 10,
pp. 1220–1227, Oct. 1988.. Calendar Queues: A Fast O(1) Priority Queue Implementation for the Simulation Event Set Problem
3. https://stackoverflow.com/questions/40113972/simulate-multiple-virtual-timers-with-one-physical-timer