CSE 589
Modern Networking Concepts

Programming Assignment 2 - Report
Reliable Transport Protocols

Hemant Koti
UB Name: hemantko
UB ID: 50338187

1. Academic Integrity

I have read and understood the course academic integrity policy.

2. Timeout Scheme

a. Alternating Bit Protocol

A timeout value of *20* is used in this scheme which is derived from trial and error, experimentation. Initially, I started with 10 and increased till 15. A timer value of 15 gave issues for a higher number of messages and a large window size. Also, considering the time for the packet to reach the other side, i.e., 5 units, I further increased the timeout to 20 which turned out to be an optimal value.

b. Go Back N Protocol

A timeout value of *30* is used in this scheme. Initially, I started with 20 but the retransmission rate got higher due to early timeouts in advanced test cases thereby increasing the program runtime. Hence, I increased the values gradually and selected an optimal value of 30.

c. Selective Repeat Protocol

A timeout value of *30* is used in this scheme. In the case of selective repeat protocol, we have to maintain a timer for every packet with a constraint of having only one hardware timer. To implement multiple software timers, I invoke the timer (starttimer function in the simulator) for every 1-unit interval. By doing this I essentially count the number of unit intervals that have passed in the system (this is also stored in the variable **current_timer**).

I also maintain a structure/class declared as **SR** for every packet which contains a member instance - **timeover**. The timeover value for a packet is current_timer + timeout.

When a timer interrupt occurs in the system, I check if the timeover value for a particular packet is less than the current_time, if yes, then I send the packet to B.

The timeout value of 30 is derived from experimentation. Initially, I started with 20 and gradually increased it to 30 which was the optimal value I found.

3. Buffer and Checksum

a. Buffer

Buffering is required when we cannot serve a message that comes from layer 5. This happens when the sender might be waiting for an earlier packet acknowledgment from the receiver or the sender's window is full.

To implement the buffer mechanism, I am using a queue data structure (variable

name in code: **buffer**) provided by the C++ STL library. This is an optimal data structure as the queue follows the FIFO (first in first out) property. The earliest message in the queue is popped first and sent to the other side B.

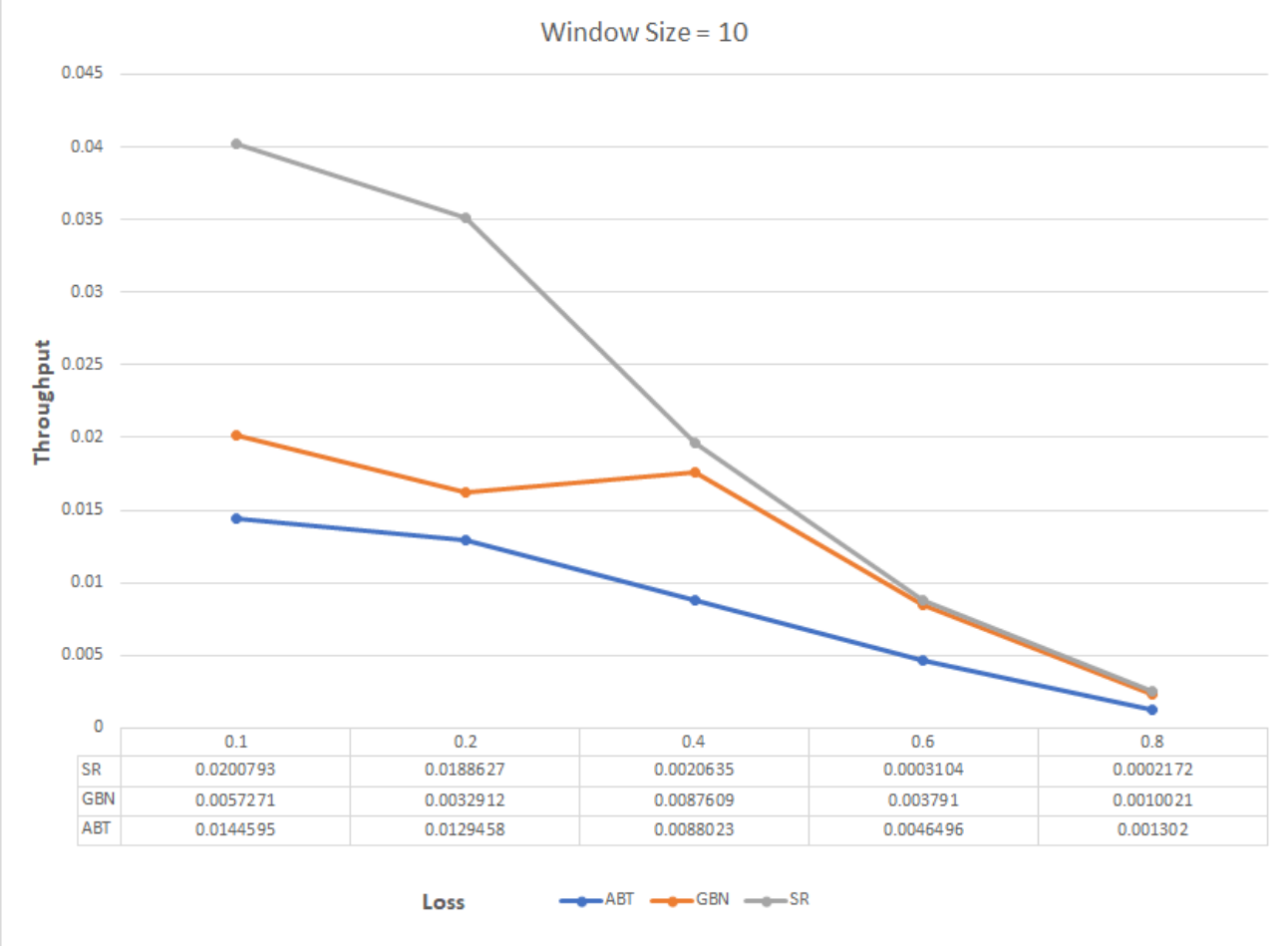
b. Checksum

I used a method *getPacketChecksum()* in all the three protocols which calculates the checksum for a given packet. I add the payload, sequence number and acknowledgment number components of a packet to calculate the checksum and assign it to the **checksum** field before sending it to the other side.

4. Performance Comparison between protocols

a. Experiment 1

i. Window Size 10

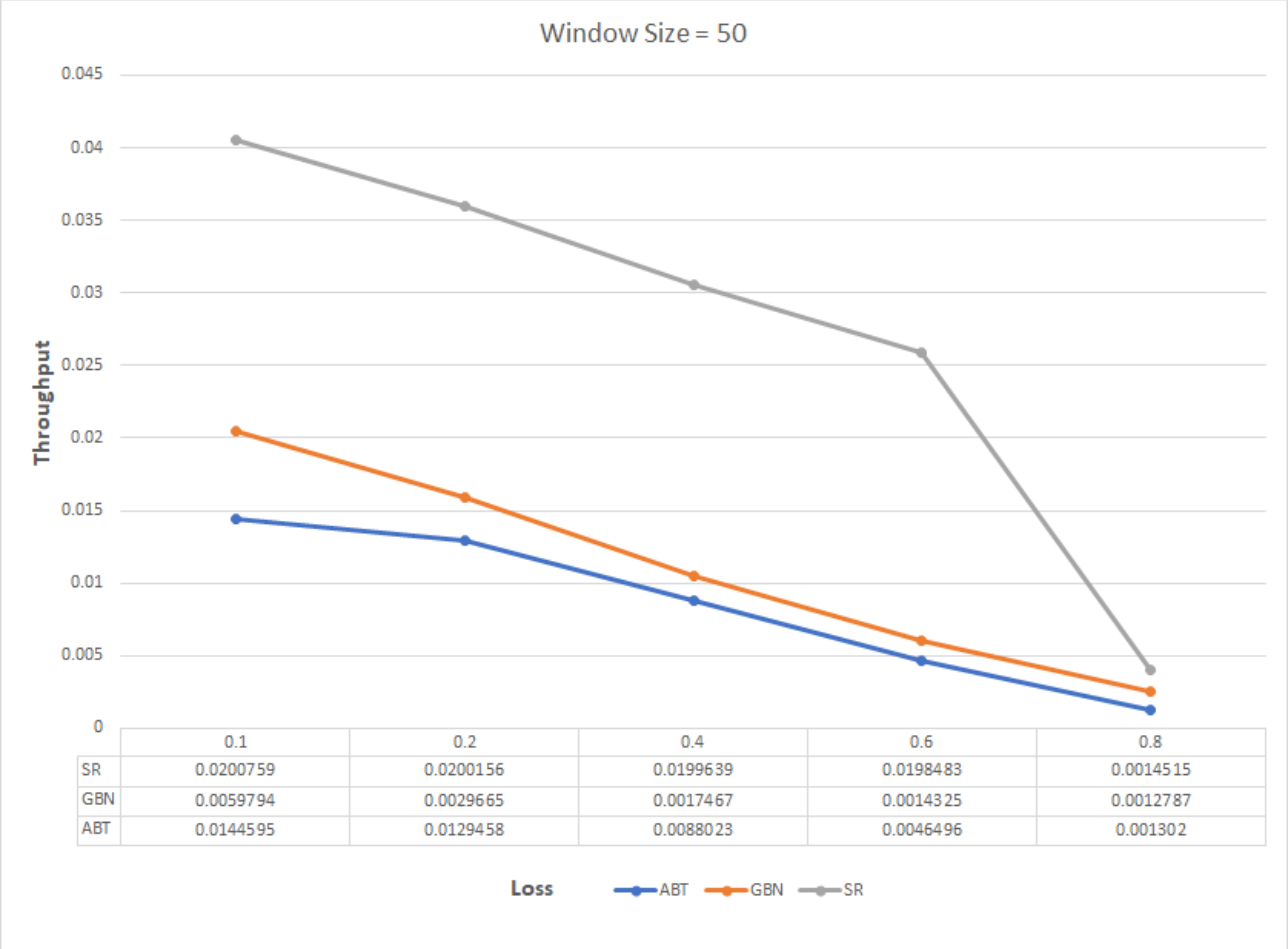


	0.1	0.2	0.4	0.6	0.8
SR	0.0200793	0.0188627	0.0020635	0.0003104	0.0002172
GBN	0.0057271	0.0032912	0.0087609	0.003791	0.0010021
ABT	0.0144595	0.0129458	0.0088023	0.0046496	0.001302

Description: In experiment 1, on the X-axis we have the loss values – 0.1, 0.2, 0.4, 0.6, 0.8 respectively. On the Y-axis we have the throughput values.

Analysis: From the above graph we can see that the throughput of SR protocol is better than GBN and ABT, as loss increases the number of transmissions also increases, that is why we see close graph line between SR and GBN. When we increase the loss value it reduces to *stop and wait* protocol, hence all the lines in the graph converge at 0.8 loss.

ii. Window Size 50

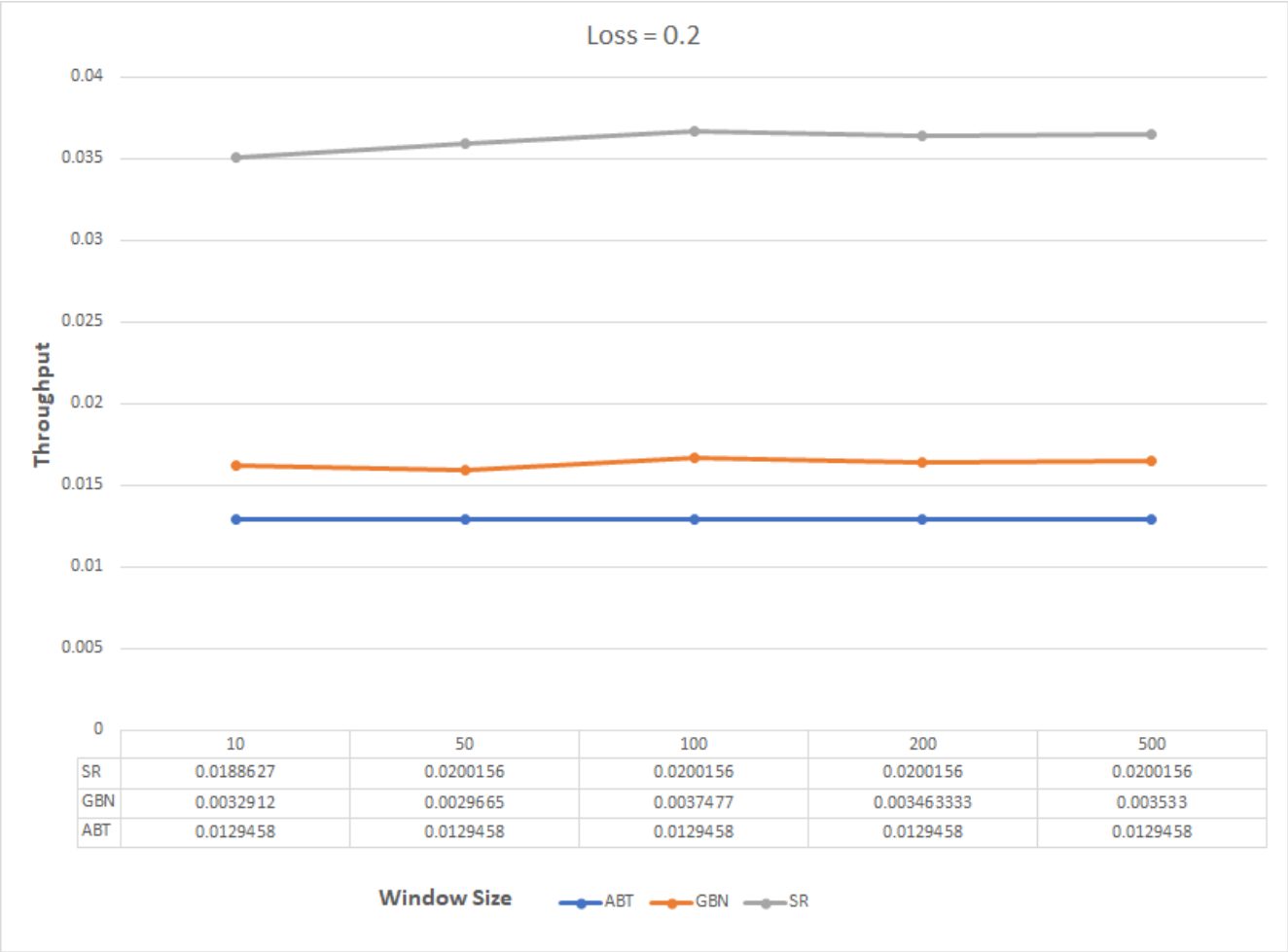


	0.1	0.2	0.4	0.6	0.8
SR	0.0200759	0.0200156	0.0199639	0.0198483	0.0014515
GBN	0.0059794	0.0029665	0.0017467	0.0014325	0.0012787
ABT	0.0144595	0.0129458	0.0088023	0.0046496	0.001302

Analysis: From the above graph we can see that GBN throughput decreases a lot as we increase the loss probability. Also, the window size is 50 and we have to send a lot of packets to B which decreases the throughput. In the case of ABT and SR protocols, the performance does not differ much from window size 10, except that we observe a steep decline in throughput for SR starting from loss probability 0.6. This is due to the high window size and an increase in loss probability value.

b. Experiment 2

i. Loss Probability 0.2

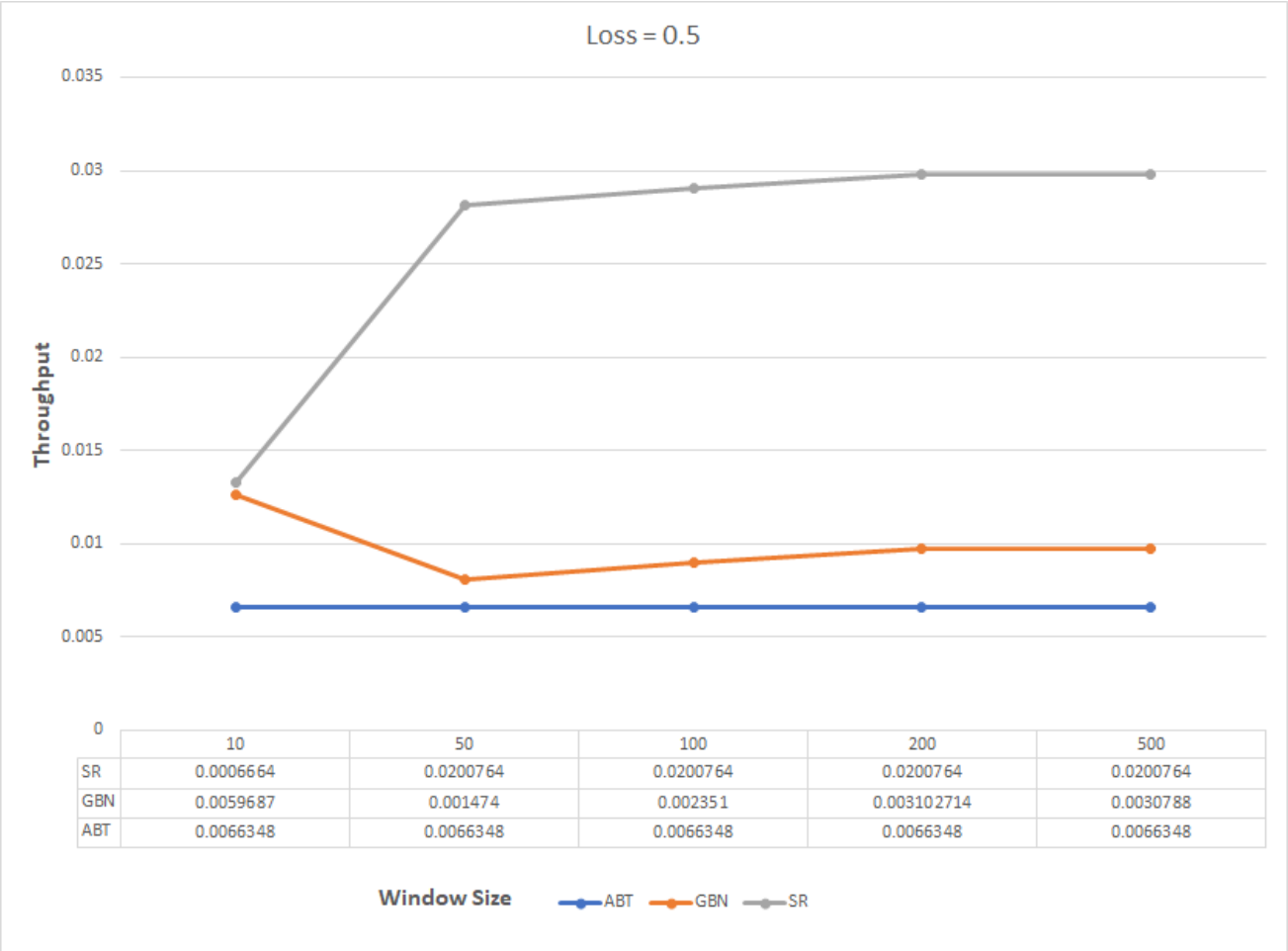


	10	50	100	200	500
SR	0.0188627	0.0200156	0.0200156	0.0200156	0.0200156
GBN	0.0032912	0.0029665	0.0037477	0.003463333	0.003533
ABT	0.0129458	0.0129458	0.0129458	0.0129458	0.0129458

Description: In experiment 2, on the X-axis we have the window size values – 10, 50, 100, 200, 500 respectively. On the Y-axis we have the throughput values.

Analysis: From the above graph we can see that at loss probability 0.2 we have the same variation in throughput for different window sizes for all the protocols. As expected, the SR protocol performs better than GBN and ABT even in this case. Also, we can state that the window sizes at a low rate of loss probability do not affect the throughput much.

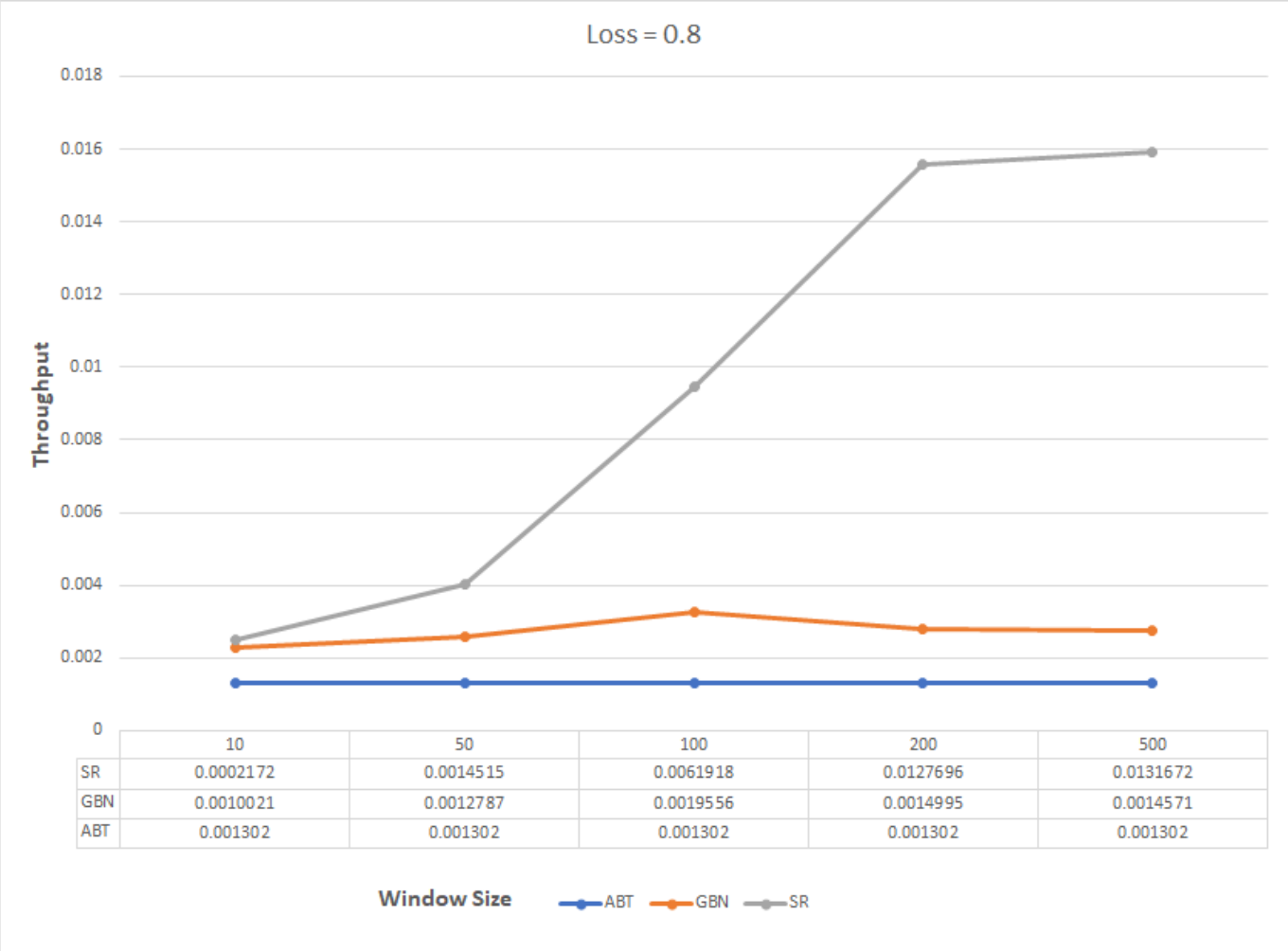
ii. Loss Probability 0.5



	10	50	100	200	500
SR	0.0006664	0.0200764	0.0200764	0.0200764	0.0200764
GBN	0.0059687	0.001474	0.002351	0.003102714	0.0030788
ABT	0.0066348	0.0066348	0.0066348	0.0066348	0.0066348

Analysis: From the above graph can see at loss probability 0.5 we have a slight variation in throughput in case of SR and GBN for initial window sizes. For larger window sizes the variation in throughput remains the same for all three protocols. Also, we can see due to a large number of retransmissions the throughput of the GBN is decreased compared to the previous loss probability value.

iii. Loss Probability 0.8



	10	50	100	200	500
SR	0.0002172	0.0014515	0.0061918	0.0127696	0.0131672
GBN	0.0010021	0.0012787	0.0019556	0.0014995	0.0014571
ABT	0.001302	0.001302	0.001302	0.001302	0.001302

Analysis: From the above graph we can see at loss probability 0.8 the performance of GBN protocol decreased a lot due to multiple packet drops and retransmissions. The SR protocol initially due to high probability gives low throughput eventually catches on as we increase the window size. ADT is not varying much from previous loss values.