Reliable Transport Protocol

PA2 ANALYSIS AND REPORT

SACHIN KUMAR KUPPAYYA
PERSON NO: 50248933

UNIVERSITY AT BUFFALO

CSE 589 – Modern Networking Concepts

I have read and understood the course academic integrity policy.

INTRODUCTION

In this Programming assignment, I have designed three Reliable data transfer protocols

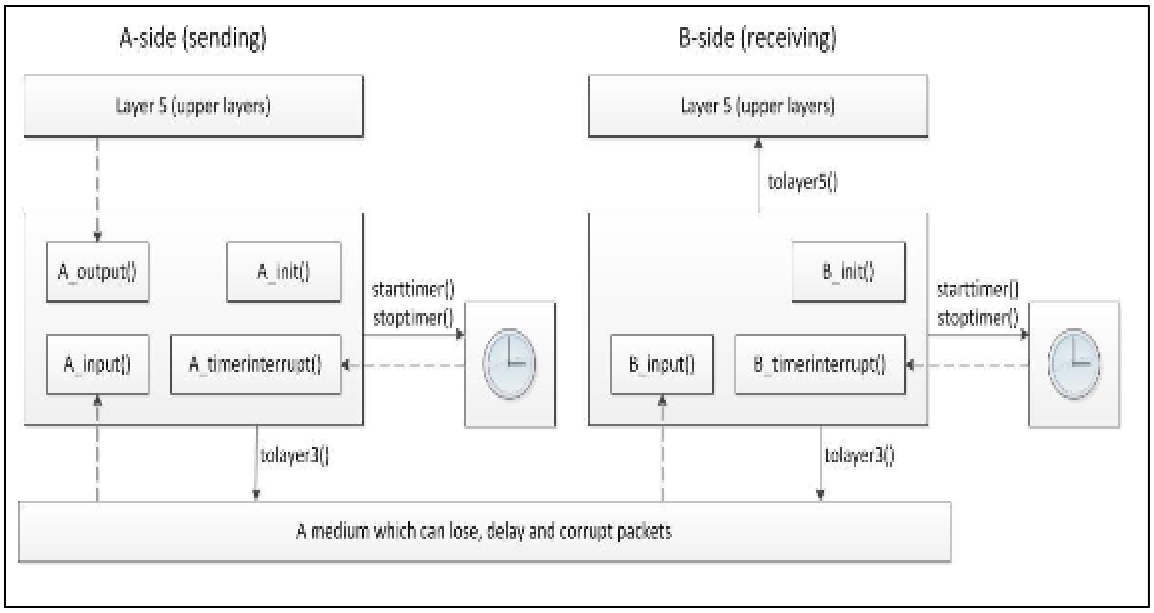
- *Alternating-Bit (ABT)*
- *Go-Back-N (GBN)*
- *and Selective-Repeat (SR).*

For a given simulator as shown below, I have written routines for A_output, A_init, A_input , A_timerinterrupt in sender and B_input, B_timerinterrupt in receiver for the three protocol.

After successfully implementing the three protocols two experiments were run and this report contains the detailed observation and analysis made after running those experiment

- Experiment 1 consists of comparing throughput of ABT, GBN and SR protocol by varying the loss probability.
- Experiment 2 consists of comparing throughput of ABT, GBN and SR protocol by varying window size with a fixed loss probability

Simulator:



Sachin Kumar Kuppayya
50248933

TIMEOUT SCHEME

- ***Alternating-Bit-Protocol(ABT)***

  This protocol uses a timer which has a fixed timeout value for a message. Every time a message is sent with a sequence number, the timer will start. Once the ACK for that message is received the timer will stop. If the timeout occurs, then the same message will be retransmitted and timer will start again. Here I have used a timeout value as 20 because for Loss = 0 and Corruption = 0, no retransmission was seen for any value greater than or equal to 20 .

- ***Go-Back-N Protocol (GBN)***

  For GBN protocol for a particular window size timeout is set based on throughput value for zero loss and zero corruption where there is no retransmission. This timeout value can be changed measuring the best throughput for varying window size and set accordingly

- ***Selective-Repeat Protocol (SR)***

  For SR protocol the implementation is mainly dependent on the timeout value because we have to implement multiple timers using a single timer. In my implementation the timeout value is dynamic. I have used the logic where every time a timer interrupt or a ACK is received the timeout value is set based on the packet whose timer is going to expire first. A detailed description of the logic is given below

  Data Structures:

```
 struct ack_pack {

   struct pkt A_pack;
   int ACK_received;
   float timestamp;

}
```

  The above structure consists of three variables.

  1. struct pkt A_pack: This is just the packet created from the message
  2. int ACK_received : This parameter is used to mark the packet for which ACK has been received
  3. timestamp: This parameter is used to store the time at which packet has been transmitted. Using this parameter, we calculate the timeout value

Pseudocode:

Below is the description of the two main routines of SR protocol A_input and A_timerinterrupt

A_input :

1. Check if the ACK received is within the window .
2. If it is, then Mark ACK_received = 1
3. Check whether sequence no = base seqnum
4. If it is true, then move the base until you find a packet where ACK_received = 0
5. Now we have changed the base, but now we to update the timeout value based on the new base
6. If new base = nextseqnum, implies there is no more data. So we stop the timer
7. Else timeout= timer – [ACK_receivedtime – minimum timestamp in the window]
8. Start timer(0,timeout)
9. The above calculation makes sure that the timer interrupt happens for the packet which has least timestamp

A_timerinterrupt:

1. Take the packets which has not yet received ACK along with timestamp
2. Find the packet with minimum timestamp in this list of packets
3. Retransmit the packet associated with the minimum timestamp
4. Update the new time stamp of this packet
5. Find the new minimum from the list of packets
6. Calculate the new timeout value as below
   Timeout = timer – [retransmitted time – time stamp of the new minimum
7. Start timer(0, timeout)
8. The above calculation makes sure that the timer interrupt happens for the packet which has least timestamp

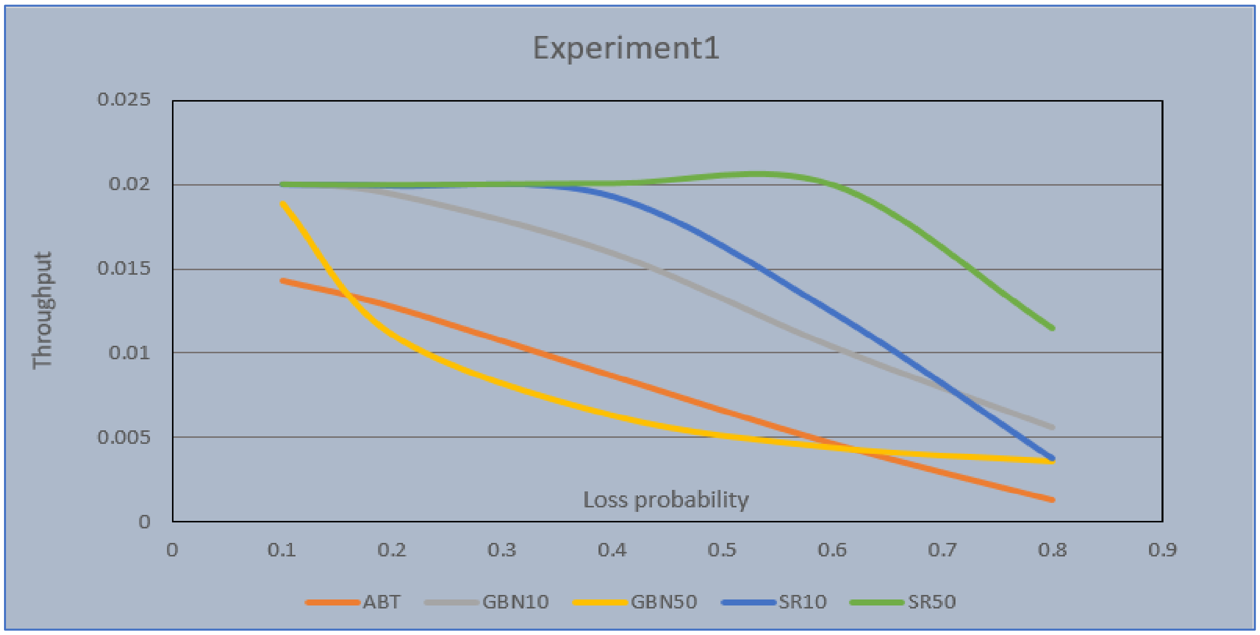
EXPERIMENT 1

The experiment is carried out for ABT , GBN and SR with the below experimental values

Experimental Values:

Protocol	Message(-m)	Loss(-l)	Corruption(-c)	time between message(-t)	Window (-w)
ABT	1000	0.1 ,0.2,0.4,0.6,0.8	0.2	50	-
GBN	1000	0.1 ,0.2,0.4,0.6,0.8	0.2	50	10,50
SR	1000	0.1 ,0.2,0.4,0.6,0.8	0.2	50	10,50

Loss vs Throughput Graph :



Throughput Values :

Loss Probability	ABT	GBN10	GBN50	SR10	SR50
0.1	0.0143422	0.0200329	0.0188488	0.0200043	0.019998
0.2	0.0127784	0.0193955	0.0110818	0.019923	0.019958
0.4	0.0086873	0.0159042	0.0063118	0.0192926	0.0200507

0.6	0.0046725	0.0103465	0.0044249	0.0124159	0.0199667
0.8	0.0013136	0.0055581	0.0035929	0.0037151	0.0114487

Observations:

- ABT
  ABT has the worst performance because as the above graph shows, with the increase in loss probability throughput decreases gradually

- GBN
  For GBN10 throughput remains constant from 0.1 to 0.2 loss. After that we can see decrease in throughput , however still the performance of GBN 10 is better than ABT

  For GBN50, initially throughput is higher than ABT but as loss increase we can see that throughput decreases significantly

- SR
  For SR10 throughput remains constant up to a loss of 0.4 .After that we can see decrease in throughput however it still performs better than GBN and ABT for most of the period

  For SR50 throughput remains constant up to a loss of 0.6. After that we can see decrease in throughput however it still performs better than GBN and ABT for most of the period. Also we can observe that with increased window size we see better performance during high loss.

Conclusion:

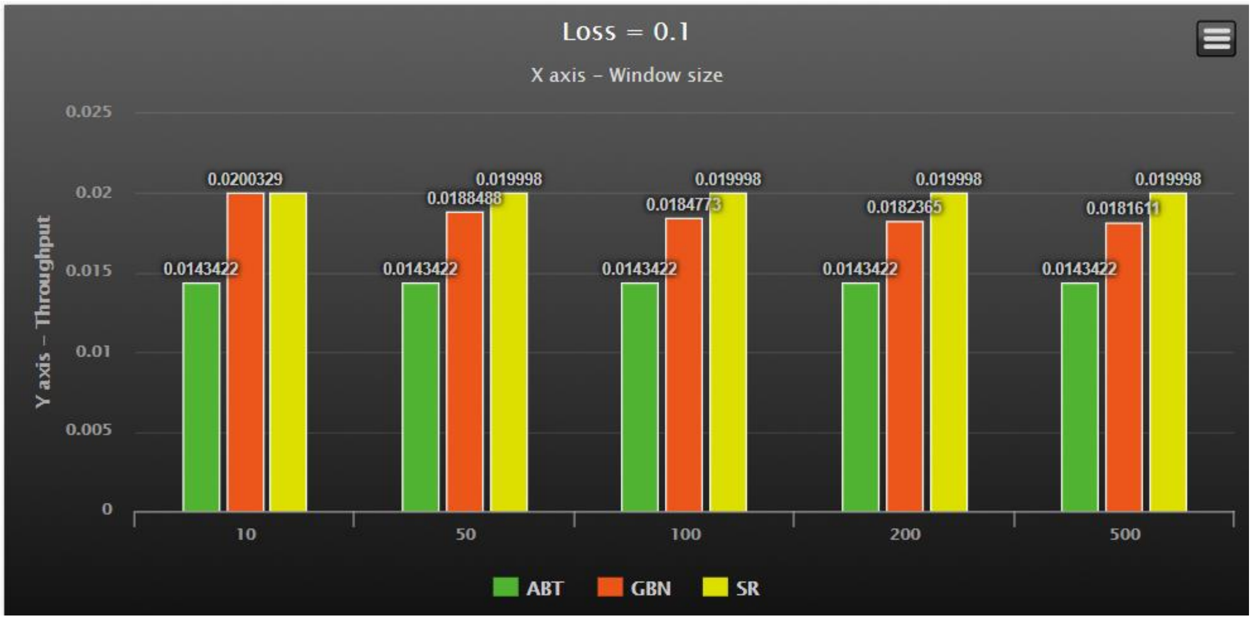
- Performance of GBN and SR is better than of ABT
- SR is the best performing protocol overall both at very high loss probability
- ABT is the worst performing protocol and shouldn't be used in real world conditions where loss and corruption are high
- GBN can be used when loss is less as it gives comparable performance to SR

‘

EXPERIMENT 2

Case 1→ Loss : 0.1

Throughput vs Window size graph:



Throughput vs Window size Values:

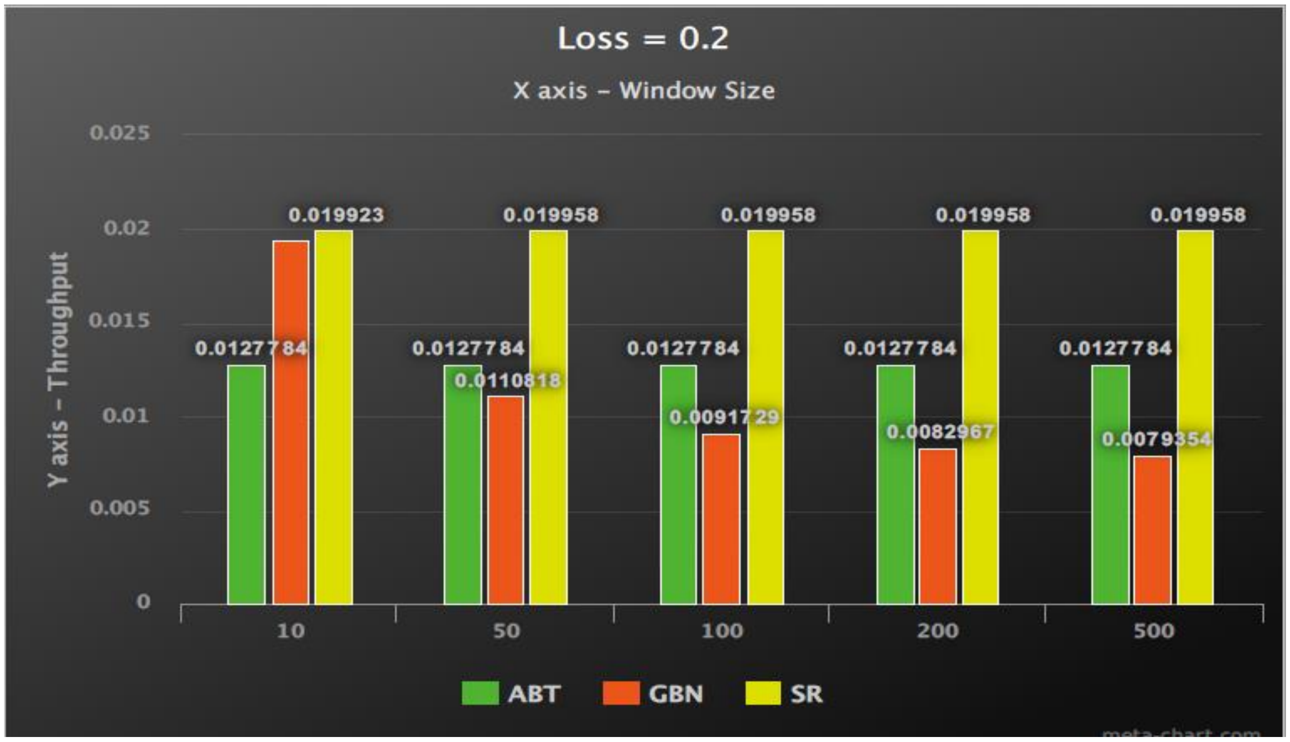
Window Size Vs Throughput					
Protocol	10	50	100	200	500
ABT	0.0143422	0.0143422	0.0143422	0.0143422	0.0143422
GBN	0.0200329	0.0188488	0.0184773	0.0182365	0.0181611
SR	0.0200043	0.019998	0.019998	0.019998	0.019998

Observation:

- In all three protocols when loss is less with the increase in window size no major change in throughput is seen
- Throughput of GBN and SR are in the same range for any window size

Case 2→ Loss : 0.2

Throughput vs Window size graph:



Throughput vs Window size Values:

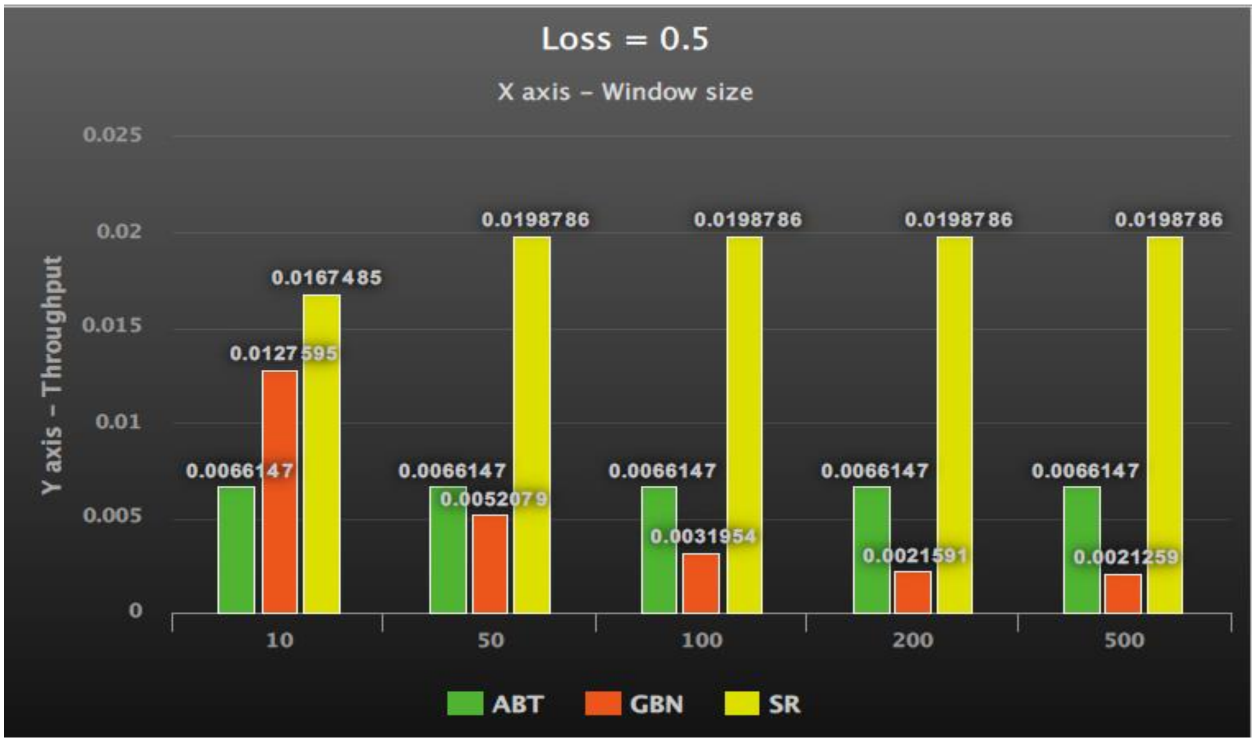
Window Size Vs Throughput					
Protocols	10	50	100	200	500
ABT	0.0127784	0.0127784	0.0127784	0.0127784	0.0127784
GBN	0.0193955	0.0110818	0.0091729	0.0082967	0.0079354
SR	0.019923	0.019958	0.019958	0.019958	0.019958

Observation:

- For GBN it is observed that with increase in window size the throughput decreases
- For SR it is observed that with increase in window size no change in throughput is seen

Case 3→ <u>Loss = 0.5</u>

<u>Throughput vs Window size graph:</u>



<u>Throughput vs Window size Values:</u>

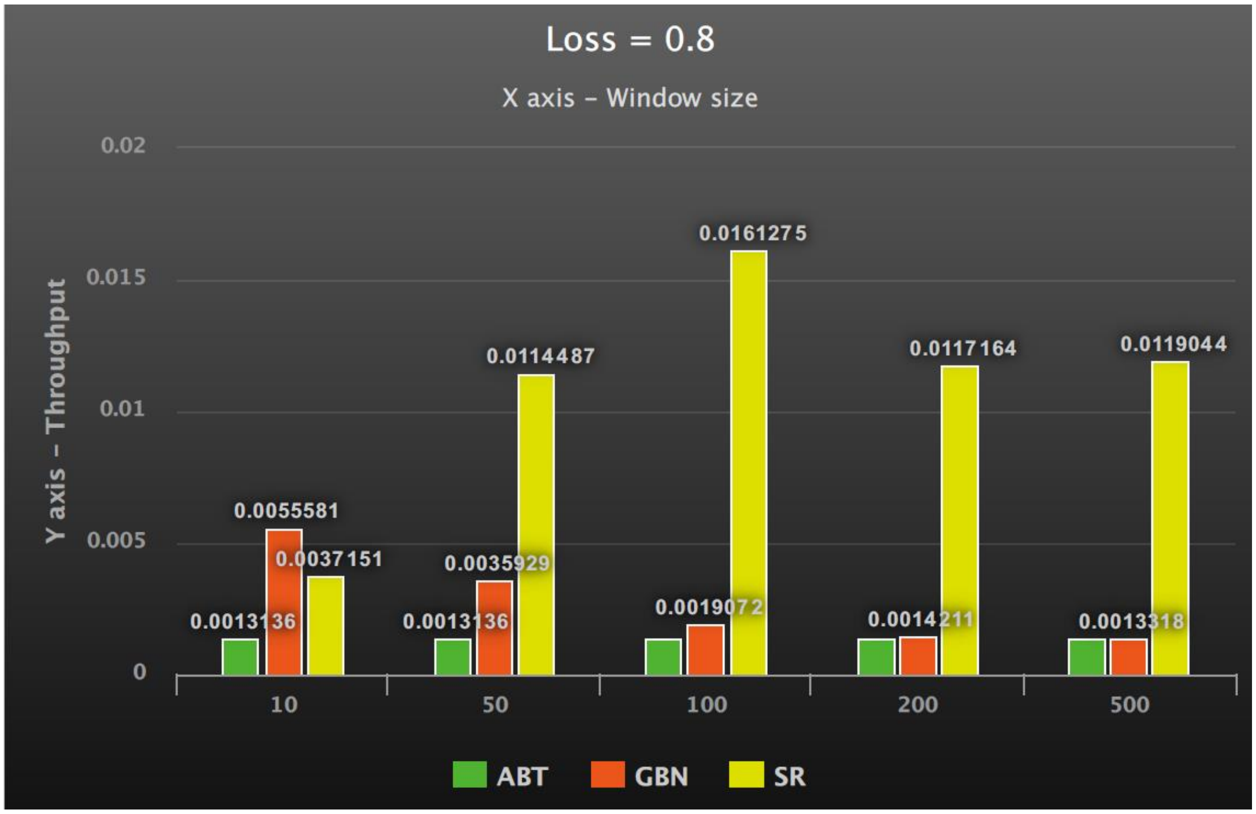
Window Size Vs Throughput					
	10	50	100	200	500
ABT	0.0066147	0.0066147	0.0066147	0.0066147	0.0066147
GBN	0.0127595	0.0052079	0.0031954	0.0021591	0.0021259
SR	0.0167485	0.0198786	0.0198786	0.0198786	0.0198786

<u>Observation:</u>

- For GBN it is observed that with increase in window size the throughput decreases
- For SR it is observed that with increase in window from 10 to 50 we see an increase in throughput but after that throughput remains constant

Case 4 → Loss = 0.8

Throughput vs Window size graph:



Throughput vs Window size Values:

Window Size Vs Throughput					
	10	50	100	200	500
ABT	0.0013136	0.0013136	0.0013136	0.0013136	0.001314
GBN	0.0055581	0.0035929	0.0019072	0.0014211	0.001332
SR	0.0037151	0.0114487	0.0161275	0.0117164	0.011904

Observation:

- At high loss, for GBN it is observed that with increase in window size the throughput decreases
- At high loss, for SR it is observed that increase in window size improves the throughput significantly

Conclusion

- ABT is the worst performer and is not practical
- During high loss, SR performs the best when window size is large.
- At very low loss SR window size doesn’t matter
- GBN performs better with lesser window size and it performance suffers when window size is large
- At very low loss throughput of SR and GBN are comparable

Reference

- Computer Netowrking – A top down approach by kurose
- http://stackoverflow.com
- Guidance from instructors and TA of CSE589