PROGRAMMING ASSIGNMENT 2

Implementing Reliable Transport Protocols

"I have read and understood the course academic integrity policy located under this link: http://www.cse.buffalo.edu/faculty/dimitrio/courses/cse4589_f14/index.html#integrity". Your submission will NOT be graded without this statement.

TIMEOUT SCHEME USED :

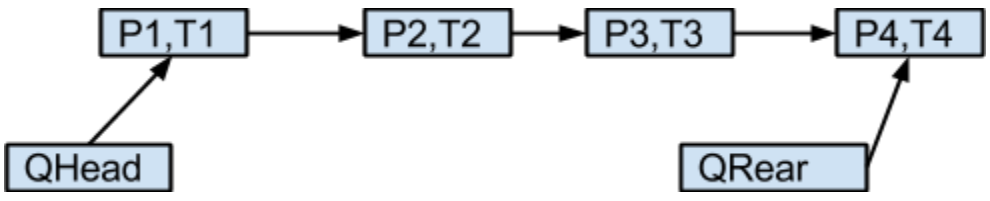
In alternating bit protocol the timeout used is of value - 25 .Since the alternating bit protocol is a wait and stop protocol and we cannot achieve much significant improvements by modifying the timeout value.By running the experiments for different settings of error and corruption.

In Go Back N ,the timeout value used is a constant value of 25 for the window sizes less than 100,for the window size > 100 , i have used a timeout value of 50.

TIMER IMPLEMENTATION IN SELECTIVE REPEAT :

In selective repeat I have implemented the multiple timers by employing a logic that uses the timer functions provided in the program.The logic makes use of the functions start timer and *stoptimer()* and a **queue data structure** where I store the packets and the time at which the packet arrived(**time variable**). I have made use of FIFO property of the queue.So the packets as they arrive are pushed into a queue along with their time.So the packet that will be moving out of the queue first will be the one that came earlier than the others.

Initially when the program starts we call the starttimer() method and as the packets arrive they are queued in the queue with the timestamp value of the packet(arrival time) .Initial starttimer() call acts as the timer for the first packet.



When the time out is reached after timeout interval,the first packet will be removed from the queue ,the packet will be retransmitted and then added back to the rear of the queue.Then we

calculate the latest time out value for setting the new timeout (timeout for the next packet in the queue)and starting the timer for the packet next inline.The timer then runs for the next packet in the queue. The new timeout is calculated as ***T2 - T + TIMEOUT.***

When we receive the ack for a packet we remove the packet from the queue.If the ack is received for the packet in the front of the queue we remove it and calculate the timeout for the second packet and start the timer with new time value.

Data Structures -

queuehead -> points to the head of the queue
queuehead->front : points to first packet in the queue. (used for dequeing)
queuehead->rear : points to the last packet.(used for enquing)

Queue implemented is of type :

```
typedef struct pkt_timer{
        float s_time ;
        int send_base_seq;
       struct pkt  st_packet;
       struct pkt_timer *next;
     }pkt_timer;
```

-- this houses the packet and its arrival time

Logic Used :

On start:

1. Initially the timer will be started for the first packet with a timeout value **T.**
2. Packets as they arrive are packaged to the struct type -***pkt_timer*** with their arrival timestamp and pushed to the queue.The first packet will be the one in front and which will be pushed out first with dequeue operation.The packets in the queue can be assumed as the packets for which timers have been started.

On timeout :

1. In A_interrupt (when timeout occurs for the first packet) at time- ***t***,the first packet **P1** will be dequeued ,the timestamp **T2** of next packet **P2** will be taken and we will calculate the new timeout value for the next packet in the queue with equation -$\boldsymbol{timeout_{t2}}$**= T2 - T + t .**
2. Then the the ***stoptimer()*** is called followed by the call of ***starttimer(***$\boldsymbol{timeout_{t2}}$***) -*** *this is like we have started a timer for packet 2.* ***General formula of timeout can be given as*** $\boldsymbol{timeout_{ti} = T_i - T_{i-1} + t.}$
3. The packet **P1** that is dequeued is then added to the rear of the queue.

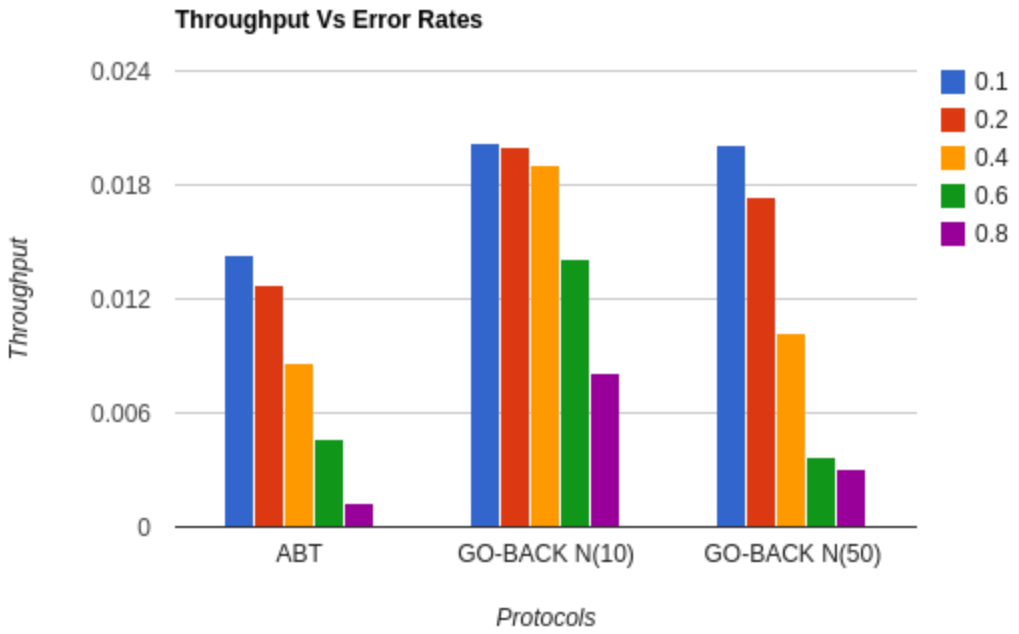
On Receiving acknowledgement:

1. In A_input on receiving an acknowledgement we have to stop the timer for that particular packet.This is done by removing the packet from the queue.If the packet is the one in the front of the queue we have to calculate the timeout for the next packet in line and start the timer for the packet as mentioned in previous step -2 -(***On timeout-2)*** remove the packet.
2. If it is not in the front of the queue we can remove the packet from the queue.

PERFORMANCE COMPARISON :

EXPERIMENT 1.

The Alternating bit protocol in comparison with the Go Back N has a poor throughput and also the throughput reduces considerably with the increase in the loss probability.
The reason for the reduced throughput can be attributed to its stop and wait behaviour.The protocol just rejects the application level packets when it is in transmission or retransmission mode,so the number of packets that are send or resend in a given time decreases as the loss rate increases(retransmissions increases).
Graph : --



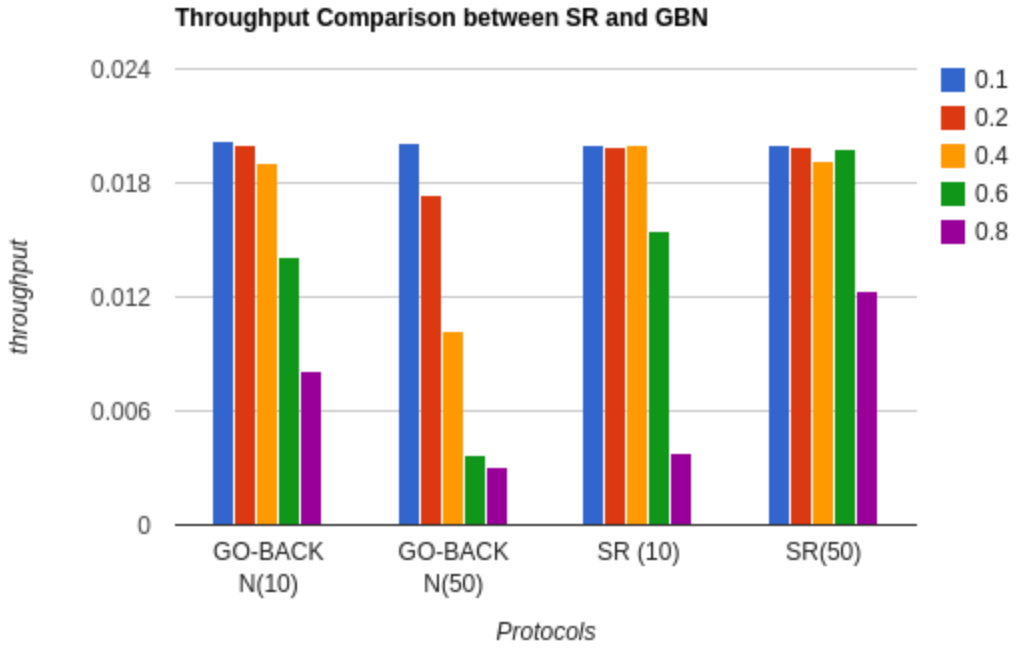
GBN VS ABT:

The GBN protocol's performance is better than the ABT,as we can see that as the loss rate increases the throughput of ABT reduces steeply whereas for GBN it reduces very gradually with a very small change in throughput for the loss rates of 0.1,0.2,0.4.

The behaviour of the GBN at high loss rate of 0.8 results in a lower throughput as in this scenario GBN resorts to lot of retransmissions of the packets within its window .

Readings :

Loss Probability	ABT Throughput	Go Back N (20) Throughput	Go Back N(50) Throughput	SR(10) Throughput	SR(50) Throughput
0.1	0.0143	0.02017	0.02011	0.01998	0.0199818
0.2	0.01277	0.01998	0.01733	0.01991	0.0199262
0.4	0.0086	0.01902	0.0102	0.02	0.0191166
0.6	0.0046725	0.01408	0.0037	0.01549	0.0197469
0.8	0.001.31	0.008	0.003005	0.0038207	0.0123025



GBN VS SR :

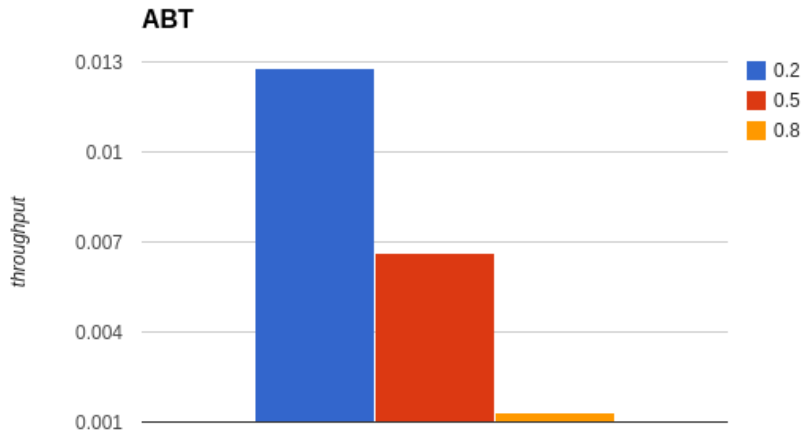
SR has better performance than GBN when we see the figures in the chart.

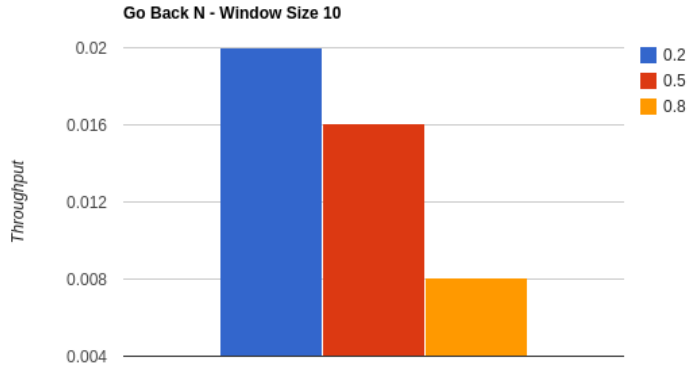
- SR gives and GBN at SR gives a higher throughput for higher loss rates when compared to the GBN.
- For low loss rates GBN and SR have matching throughputs,but for higher error rates SR has better performance.
- As the window size increases for SR the performance increases for higher error rates as compared to the smaller window size.
  SR window - 10 Loss 0.8 Throughput :**0.0038207**
  SR window - 50 Loss 0.8 Throughput :**0.0123025**

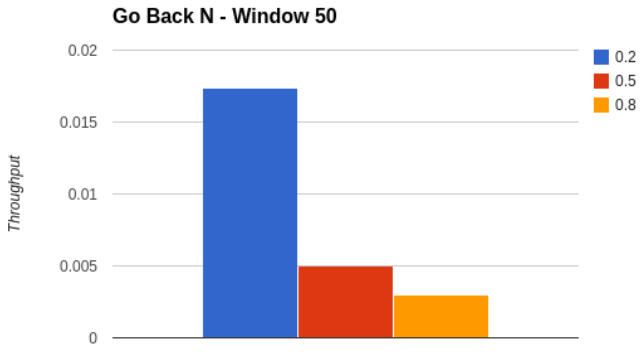
The better performance of the SR compared to GBN at higher error rates can be attributed to the the lower retransmission of packets on loss when compared to the GBN where we retransmit all the packets that falls within the window which results in the congestion in the **simulator event queue**.

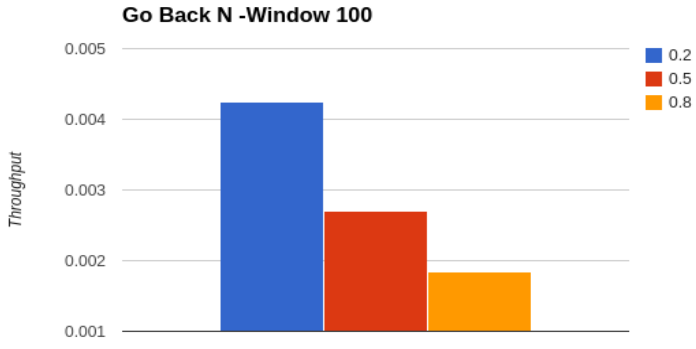
Experiment 2 :

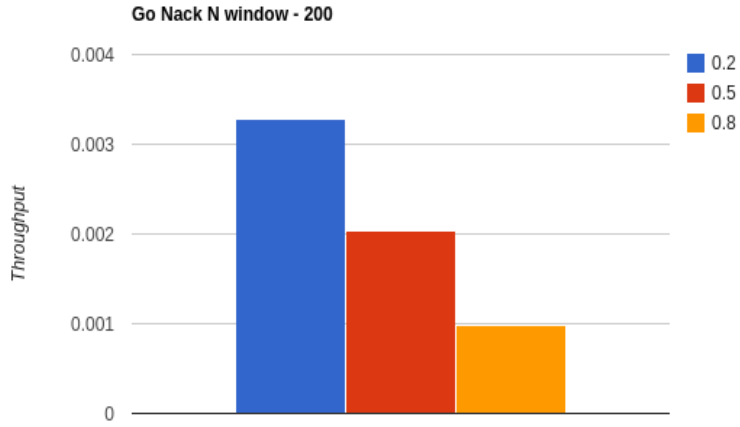
ABT:

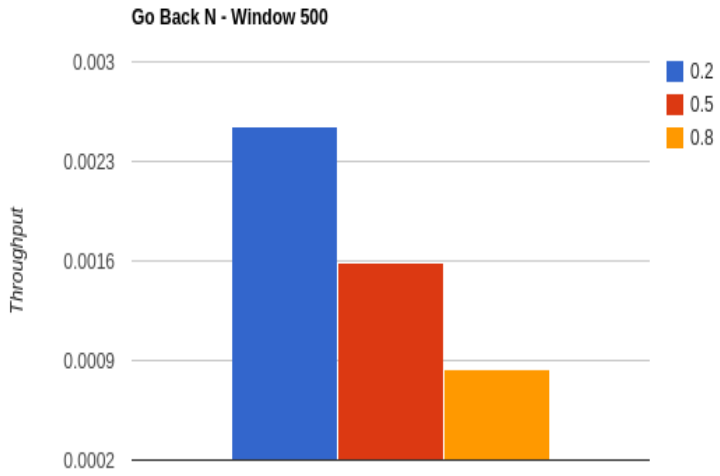





Go Back N - Window 50
0.02
0.015
0.01
0.005
0
Throughput
0.2
0.5
0.8


Go Back N -Window 100
0.005
0.004
0.003
0.002
0.001
Throughput
0.2
0.5
0.8


Go Nack N window - 200
0.004
0.003
0.002
0.001
0
Throughput
0.2
0.5
0.8

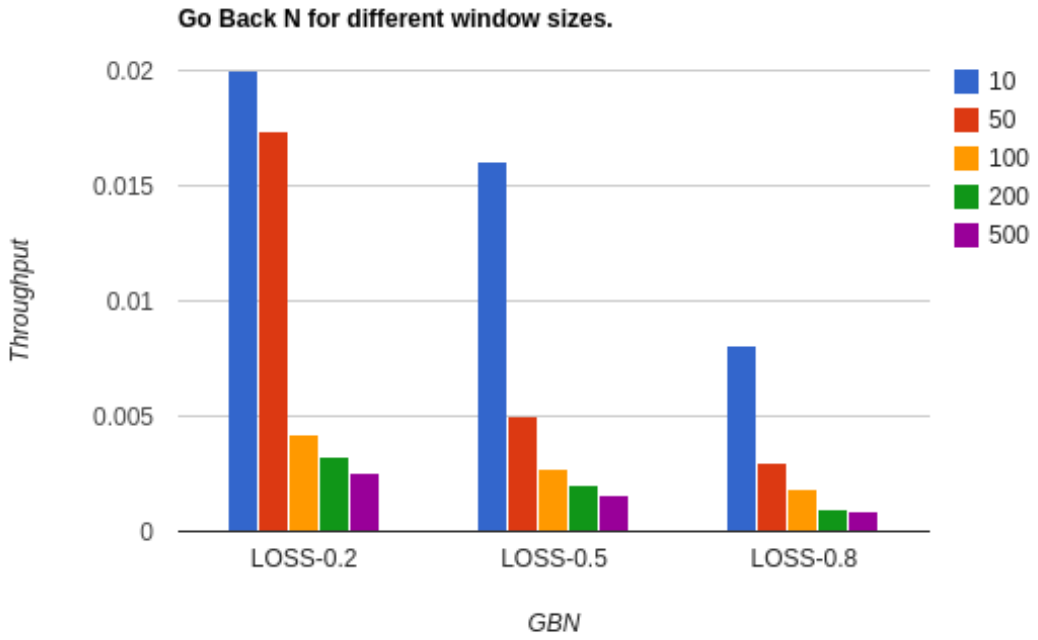


Readings :

GBN for windows -50,100,200,500 And ABT

Loss	GBN WINDOW 10 Throughput	GBN WINDOW-50 Throughput	GBN WINDOW 100 throughput	GBN WINDOW 200 throughput	GBN WINDOW 500 throughput	ABT
0.2	0.01998	0.01733	0.00425	0.00328	0.00255	0.01277
0.5	0.016054	0.0050193	0.002689	0.002028	0.001589	0.00661
0.8	0.0081	0.003005	0.00184	0.000991	0.0008402	0.00131

Please note - Timeout value is set as 25 for window sizes 10 amd 50.The timeout value is set to 50 for window sizes 100 and above.As the window size increases the throughput reduces as the we will be transmitting a large number of packets when a packet loss is encountered.



Comparison of Go Back N for different window sizes for different loss rates

The performance of the Go Back N reduces as the window size increases and the loss rate increases.For higher loss rate when a packet is lost the Go Back N protocol sends in all the packet that falls in the window and as the window size increase this number (number of packets to be retransmitted also increase) also increases, resulting in a lower throughput.

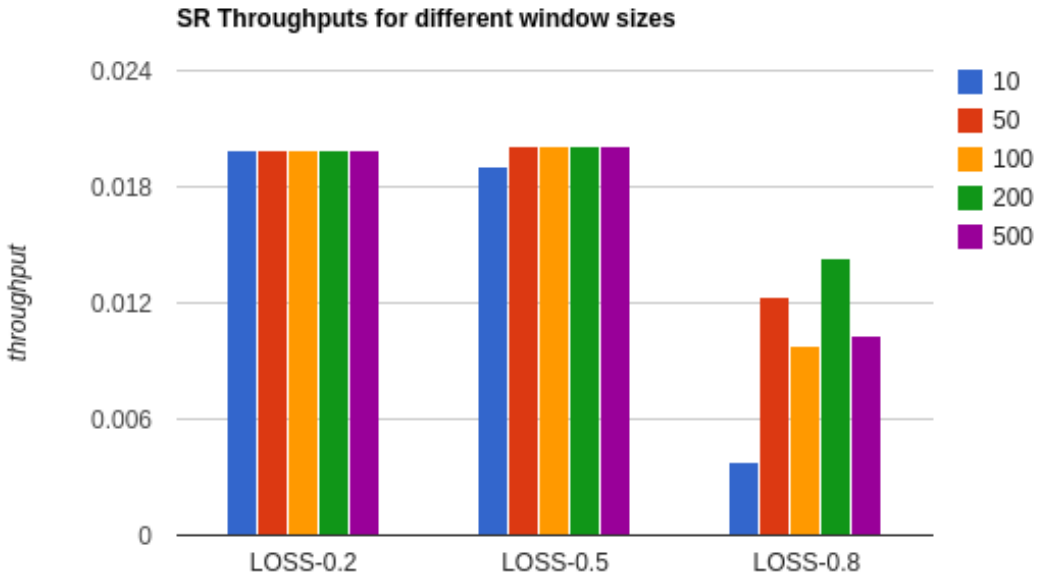
Readings

SR for windows -50,100,200,500

Loss	SR WINDOW -10	SR WINDOW-50	SR WINDOW 100	SR WINDOW 200	SR WINDOW 500
0.2	0.01991	0.0199262	0.0199262	0.0199262	0.0199262
0.5	0.0190026	0.0201059	0.0201059	0.0201059	0.0201059
0.8	0.0038207	0.0123025	0.0098215	0.014324	0.010299

Please note :
Throughput Readings for window 500, Loss 0.8 was taken for seeds - 3333,4444,9999 and averaged .
Throughput for window 200 , Loss 0.8 was taken for seeds - 1234,4444,9999
Throughput for window 100 , Loss 0.8 was taken for seeds -1234,1111,3333,2222

please note :*legend -blue for window size 10*
red for window size 50
yellow for window size 100
green for window size 200
violet for window size 500



The selective repeat protocol (SR) has a consistent performance for different window sizes unlike the GBN protocol where it varies for different window sizes.
The throughput for the higher loss rate is less for a smaller window size ,but it increases as the window size increases.For the timeout value of 30.

Conclusion :

From the experiments 1 and 2 I can conclude that the performance of the SR is the best among the other three protocols.SR provides the best throughput compared to the GBN when the window size is larger and the loss rate is high.