CSE 489/589
Programming Assignment 2
Reliable Transport Protocols

VAISHNAVI KANNAN
50245763

I Have Read And Understood The Course Academic Integrity Policy.

1. Alternating Bit Protocol/ ABT

The following FSM was used as a basis for programming the ABT/RDT3.0 protocol (Reference: Computer Networking a Top Down Approach by Kurose Ross).

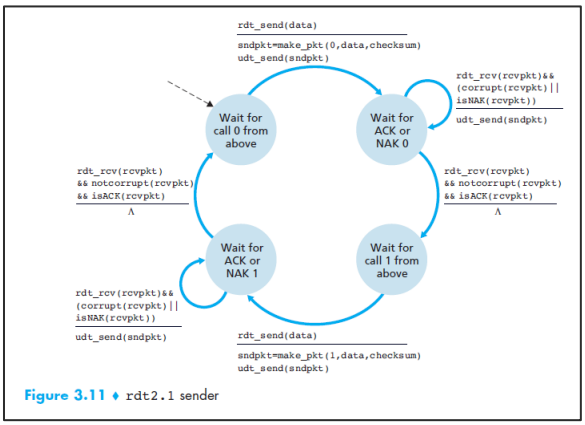

Figure 3.11 ♦ rdt2.1 sender

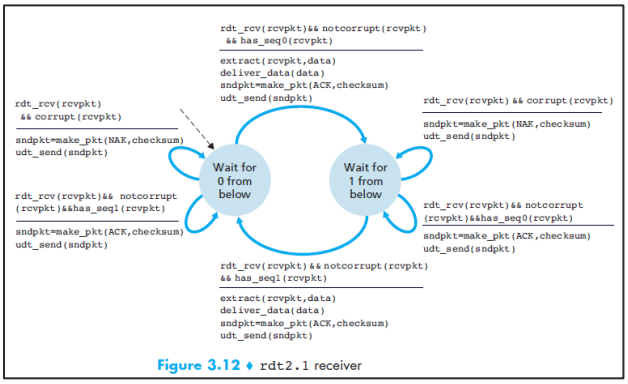

Figure 3.12 ♦ rdt2.1 receiver

Timer Implementation:

The timer was started using the function starttimer(0, TIMEOUT) whenever, a packet is sent and stopped using stoptimer(0) whenever an ACK arrives.

A static TIMEOUT value of 35.0 was chosen for the implementation. The chosen value gave the best throughput and provided sufficient time for the packets to travel from A to B and account for loss and delay in the network.

Go-back-N

The following FSM was used as a basis for programming the ABT/RDT3.0 protocol (Reference: Computer Networking a Top Down Approach by Kurose Ross).

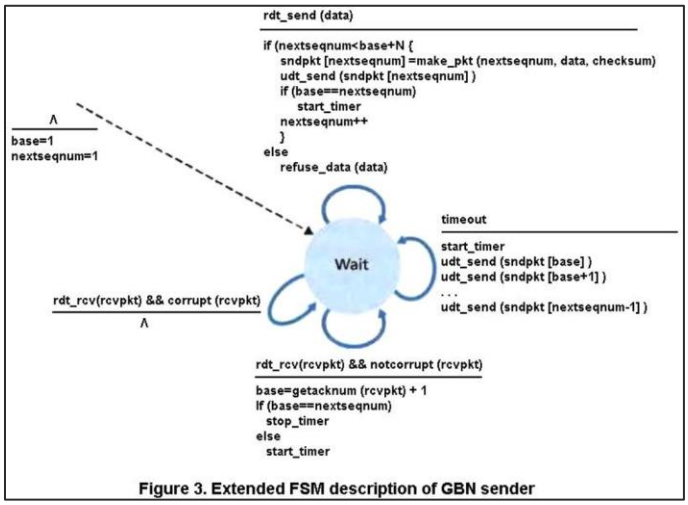

Figure 3. Extended FSM description of GBN sender

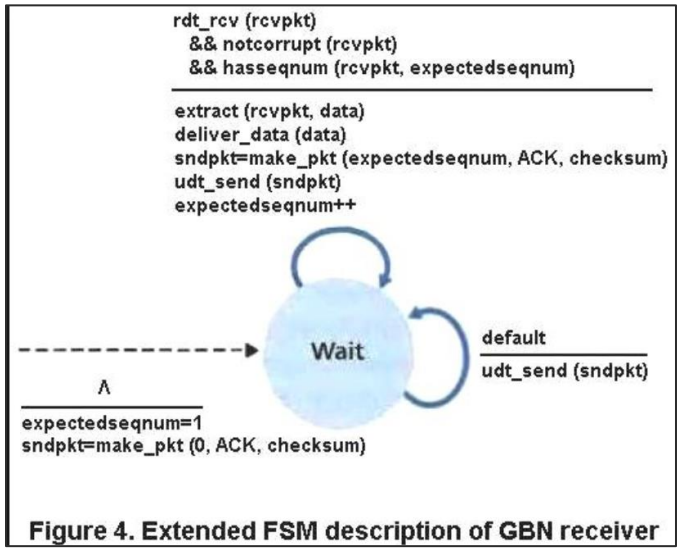

Figure 4. Extended FSM description of GBN receiver

Timer Implementation:

The timer was started for TIMEOUT whenever a base with sequence number equal to the sender's base was sent. The timer is stopped if an ACK is received for the sender_base packet. Else, when timeout occurs, the sender sends all previously sent but unacked packets.

To improve the throughput, different values for TIMEOUT were chosen based on the window size and the TIMEOUT value was further updated based on the loss occurrence during the running of the program. The TIMEOUT values are as given below:

```
if(window_size<=50)
    {TIMEOUT=11.5;
    }
  else if(window_size<=100)
    {TIMEOUT=15;
    }
```

```
  else if(window_size<=150)
    {TIMEOUT=18;
    }
  else if(window_size<=200)
    {TIMEOUT=20;
    }
  else if(window_size<=500)
    {TIMEOUT=23;
    }
  else
    {TIMEOUT=25;
    }
```

The loss probability was calculated as the 1 – (number of correct Acks / Number of packets sent). The TIMEOUT values were increased based on the following scheme:

```
 if(loss>=0.8)
 {
  TIMEOUT=40;
Return;
 }
 else if(loss>=0.6)
 {
  TIMEOUT=35;
 return;
 }
 else if(loss>=0.4)
 {
  a=0.050;
 }
 else if(loss>=0.2)
 {
  a=0.040;
 }
 else
 {
  a=0.030;
 }
 sample_rtt=x-y;
 estimated_rtt=(a*estimated_rtt)+((1-a)*sample_rtt);  /** 2 zeros**/
 difference=abs(sample_rtt - estimated_rtt);
 dev_rtt=(0.75*dev_rtt)+(0.25*difference); /** 3  zeros **/
 if(loss<=0.6)
{
TIMEOUT+=0.01*estimated_rtt+dev_rtt; /*0.004090*/
}
```

Where x=Time at which the last ack was received and y=time at which the ack was sent. The coefficients chosen for the calculation of estimated_rtt and dev_rtt differ from the recommended values and were chosen by hit and trial to give the best throughputs. The sample and estimated RTT plots are plotted in Fig 8 – 10.

Selective Repeat:

Timer Implementation:

1. An array 'Arrival_time' was maintained to store the time at which a packet with seqnum packet.seqnum was sent to layer 3 to be delivered to B. The array timer_values stores the timeout for different packets.
2. The float variable 'New_timeout' stores the time for which the timer needs to be started. Its initial value is set to 'TIMEOUT'.
3. The float variable 'timer_started_at' stores the time at which the timer was started and the variable 'timer_Stopped_at' stores the time at which the timer was stopped.
4. Whenever, a packet arrives and its seqnum==sender_base (i.e. in this case the timer should not be already running, since for this case, either the arriving packet is the first packet or all other previously sent packets have been acknowledged), then the timeout value of the packet is set to the value of 'TIMEOUT' and the packet is sent.
5. When a new packet arrives, and it's seqnum !=sender_base, the timer should already be running for some other packet. The timeout value of this packet is set to the value 'Timeout_value=TIMEOUT+(Arrival_time[packet.seqnum]-timer_started_at). For the packet, first the amount of time for which the timer has already been running is calculated. This amount is added to the 'TIMEOUT' so that each packet effectively is retransmitted after the 'TIMEOUT' value.
6. When the timer stops, the entries in the array timer_values are decreased by the 'New_timeout' amount. When a packet is retransmitted, its 'timer_value' array field is set to 'TIMEOUT'.
7. When an ack is received, the following two cases may take place:
   a) Packet.seqnum==sender_base. In this case, the timer is stopped and the packet is marked as ACKED. The time difference between the Arrival_time of the packet and the ack received time is calculated and the 'timer_values' array entries are decreased by this amount. The 'new_timeout' is set to the minimum entry in the 'timer_values' array.
   b) The packet is not the base packet. The timer continues to run and the packet is marked as ACKED. The entry corresponding to this packet is set to zero and is not considered for the 'New_timeout' value henceforth.

An example run is shown below:

Consider a window_size of 4 and TIMEOUT=20.0 and New_timeout=20.0.

The initial values for the packets In the window are:

Packet seqnum	Arrival_time	Timer_values
1	0	20
2	2.23	22.23
3	3.3	25.53
4	4.62	30.15

Case1: Timer is stopped for Packet 1. The updated values are:

Packet seqnum	Arrival_time	Timer_values	Time remaining for timeout

1	0	20	0
2	2.23	22.23	2.23
3	3.3	25.53	3.3
4	4.62	30.15	4.62

The timer is now started for the minimum entry of the timer_values array, which in this case is New_timeout = 2.23.

Case 2: An ACK for the packet 1 arrives at 11.23 time units. The updated values are:

Packet seqnum	Arrival_time	Timer_values	Time remaining for timeout
1	0	20	0
2	2.23	22.23	10.6
3	3.3	25.53	13.9
4	4.62	30.15	18.52

The timer will now be started for the value 10.6.

Case 3: When a new packet arrives at say 35.44 and the timer is already running for say, 6 time units i.e. timer_started_at=29.44.

The values are given as:

Packet seqnum	Arrival_time	Timer_values	Time remaining for timeout
1	0	20	0
2	2.23	22.23	10.6
3	3.3	25.53	13.9
4	4.62	30.15	18.52
5	35.44	26	26

The timer_value for the new packet (packet 5) is set to = TIMEOUT+(Arrival_time[packet.seqnum]-timer_started_at) =20+ (35.44 – 29.44) = 26.

The packets already ACKED are in green and others waiting Acknowledgements are in blue.

EXPERIMENTS

Experiment 1 and 2 were conducted for c=0.2, m=1000 and t=50.0.

Experiment 1

Performance Comparison:

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

For window size = 10

Table 1 summarizes the 'average' throughput obtained for the three protocols ABT, GBN and SR for a loss probability of 0.1, 0.2, 0.4, 0.6 and 0.8 for a window size of 10. The corresponding performance graph is shown in **Fig 1**.

LOSS	ABT	GBN	SR
0.1	0.0102396	0.0192352	0.0201653
0.2	0.0087324	0.0200396	0.0199456
0.4	0.004093	0.0173817	0.0180694
0.6	0.0026899	0.0140352	0.0141869
0.8	0.0007467	0.0048835	0.0084

Table1: Performance comparison of ABT, GBN and SR for window size =10.

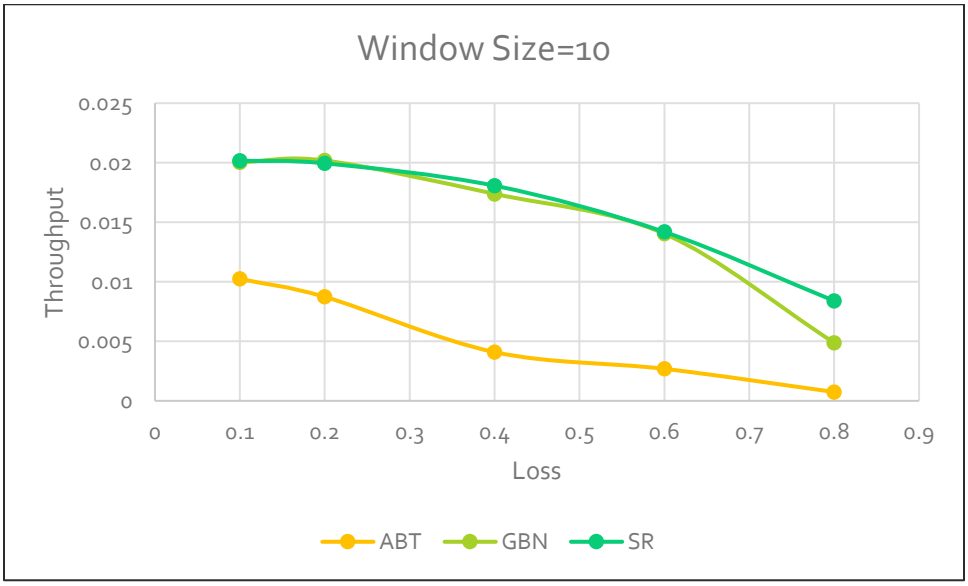

Fig 1: Graph showing the throughput for ABT, GBN and SR for window size =10.

From Fig 1, we can see that Selective Repeat protocol performs better than both ABT and GBN. The SR curve is closely followed by the GBN curve. This is expected since SR is known to have a better throughput in comparison to GBN and ABT, and GBN in turn outperforms ABT. The

efficiency/ throughput decreases with an increase in loss probability as the number of packets to be retransmitted increases and hence, congestion of the network also increases.

For window size = 50

Table 2 summarizes the 'average' throughput obtained for the three protocols ABT, GBN and SR for a loss probability of 0.1, 0.2, 0.4, 0.6 and 0.8 for a window size of 50. The corresponding performance graph is shown in **Fig 2**.

LOSS	ABT	GBN	SR
0.1	0.0102396	0.019289	0.019905
0.2	0.0087324	0.0200396	0.019995
0.4	0.0062528	0.0064049	0.0199981
0.6	0.0026899	0.0032467	0.0199439
0.8	0.0007467	0.0033498	0.0121725

Table 2: Performance comparison of ABT, GBN and SR for window size =50.

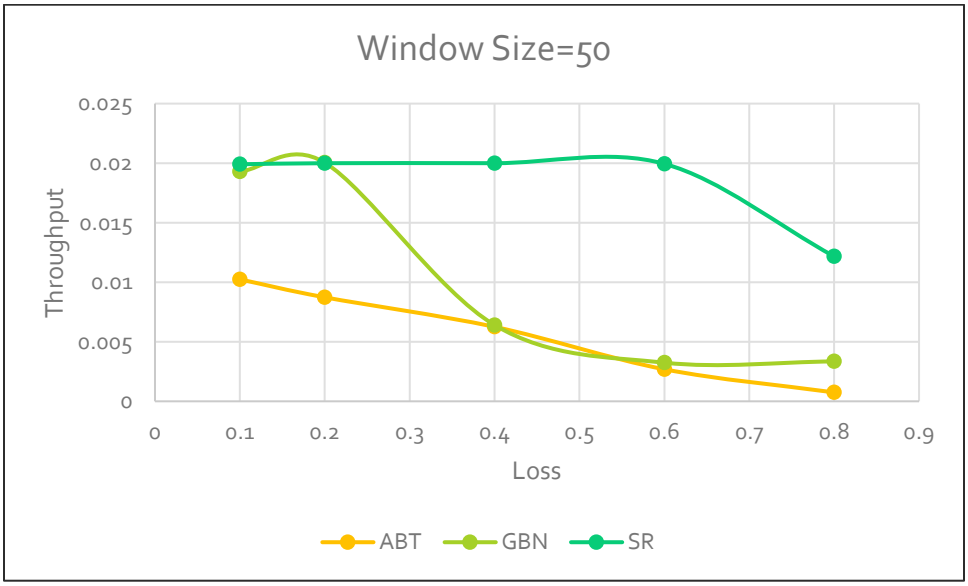

Fig 2: Graph showing the throughput for ABT, GBN and SR for window size =50.

From Fig2, we can see that again Selective Repeat protocol performs better than both ABT and GBN. However, unlike the previous case, the GBN curve shows a sharp decline in the throughput, which is expected with the increase in the window size. With a larger window size, the GBN protocol will have to transmit larger amount of packets in case of a loss and hence the congestion increases many-folds.SR, on the other hand is somewhat resistant to loss and hence shows a gradual decline. Further, we can see that the throughput of ABT does not change as it is independent of the window size and both the sender and receiver maintain a constant window size of 1.

Experiment 2

Performance Comparison:

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

Loss=0.2

Table 3 summarizes the 'average' throughput obtained for the three protocols ABT, GBN and SR for a window size of 10, 50, 100, 200 and 500 and a loss probability of 0.2. The corresponding performance graph is shown in **Fig 2**.

Window Size	ABT	GBN	SR	Time Taken for SR	No of Packets Delivered for SR
10	0.0087324	0.019958	0.019463	51328.88	999
50	0.0087324	0.0198941	0.019463	51328.88	999
100	0.0087324	0.0197557	0.019838	50358.95	999
200	0.0087324	0.0192798	0.020221	49354.66	998
500	0.0087324	0.0192798	0.021022	47521.4	999

Table 3: ABT, GBN and SR performance comparison for loss=0.2.

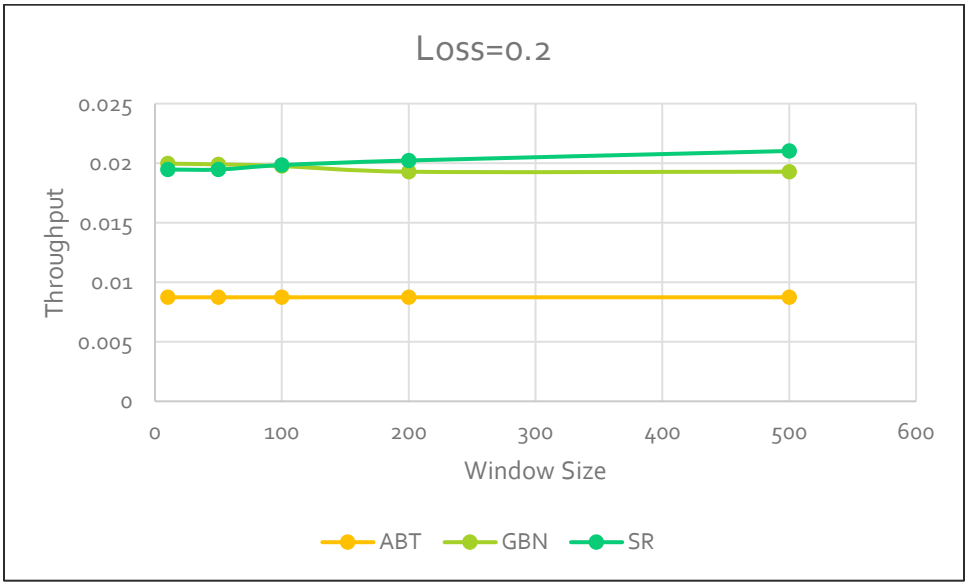

Fig 3: Graph showing the performance comparison for ABT, GBN and SR at loss=0.2.

From Fig 3, we see that the performance of the three protocols varies less with the window size when the loss=0.2. More specifically, ABT shows no difference in throughputs for different window sizes. This observation is in accordance with the fact that ABT uses a constant window size of 1 at both the sender and receiver side. Further, there is slight decrease in the performance of GBN for an increase in the window which shows that the performance of the protocol decreases due to possible increase in congestion in the network. However, since the loss probability is small, the number of packet retransmissions is small, and hence the performance is more or less maintained for different window sizes. Selective Repeat however, shows a gradual increase in the throughput. Since, SR is tolerant to loss and delay in the network, an increase in the window size increases the number of packets transmitted and the number of retransmissions is more or less unaffected. The observations for SR are also in accordance with the throughput formula :

$$\text{Throughput} = \text{window\_size}/\text{RTT}$$

Hence, SR shows an increase in the throughput, GBN shows a small decline and ABT remains constant.

Loss=0.5

Table 4 summarizes the 'average' throughput obtained for the three protocols ABT, GBN and SR for a window size of 10, 50, 100, 200 and 500 and a loss probability of 0.5. The corresponding performance graph is shown in **Fig 4**.

Window Size	ABT	GBN	SR
10	0.004582	0.019831	0.019268
50	0.004582	0.019831	0.019729
100	0.004582	0.012226	0.01973
200	0.004582	0.008063	0.019747
500	0.004582	0.005403	0.019768

Table 4: ABT, GBN and SR performance comparison for loss=0.5

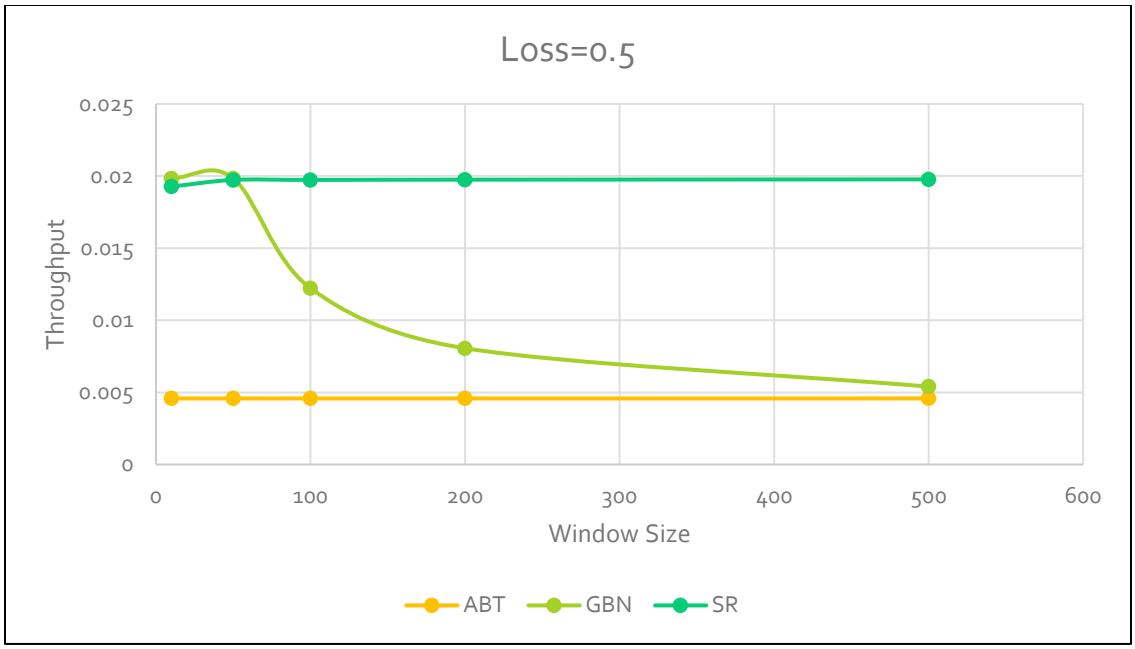

Fig 4: Graph showing the performance comparison for ABT, GBN and SR at loss=0.5.

From Fig 4 above, we see that SR's throughput is not much affected by window size although it does increase by a very small amount. Further, ABT does not show any change in throughput with the window size. GBN, however, shows a decrease in the efficiency. This is expected as the any loss/corruption now results in more number of packets being retransmitted than for a smaller window size, increasing the congestion and thus decreasing efficiency. Further, we can see that for a loss of 0.5, the GBN curve declines more rapidly than for the case of 0.2.

Loss=0.8

Table 5 summarizes the 'average' throughput obtained for the three protocols ABT, GBN and SR for a window size of 10, 50, 100, 200 and 500 and a loss probability of 0.8. The corresponding performance graph is shown in **Fig 5**.

WINDOW SIZE	ABT	GBN	SR
10	0.000944	0.004643	0.010737
50	0.000944	0.004081	0.00994
100	0.000944	0.002127	0.00994
200	0.000944	0.001347	0.009422
500	0.000944	0.001152	0.009397

Table 5: ABT, GBN and SR performance comparison for loss=0.8

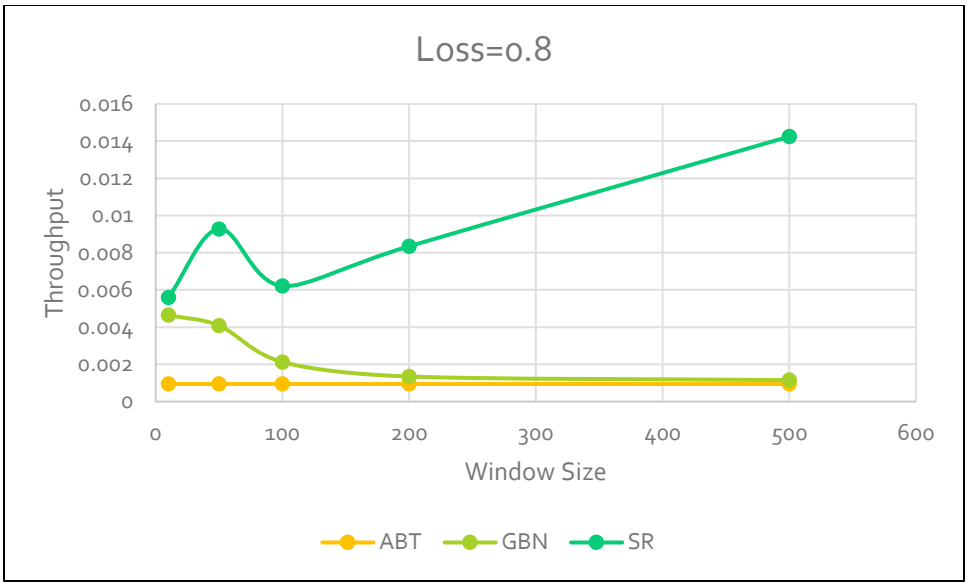

Fig 5: Graph showing the performance comparison for ABT, GBN and SR at loss=0.8.

For a high loss probability of 0.8, we see that both GBN and SR show a decrease in efficiency with an increase in window size. The corruption coefficient 'c' is also set to 0.2, indicating that the channel is highly unreliable and hence leads to a lot of congestion. Therefore, a decrease in throughput is expected. ABT, as expected, is unaffected by the window size.

Experiment 3: The 3 protocols were further tested for extreme values of the parameter t {=0.8, 100.0)

For t=0.8: The throughputs obtained for ABT, GBN and SR for t=0.8 are shown in **Table 6** and the graph is plotted in **Fig 6**.

Loss	ABT	GBN	SR
0.1	0.026244	0.034421	0.298601
0.2	0.014959	0.029434	0.29201
0.4	0.011422	0.024121	0.028402
0.6	0.002527	0.022885	0.028456
0.8	0.002523	0.011285	0.0237

Table 6: Table showing the performance comparison for ABT, GBN and SR at t=0.8.

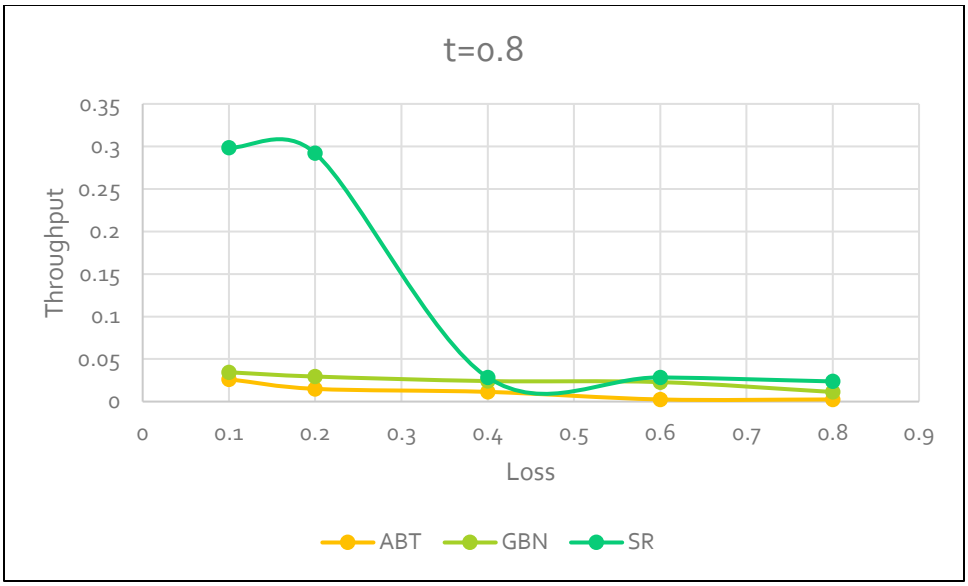

Fig 6: Graph showing the performance comparison for ABT, GBN and SR at t=0.8.

All the 3 protocols show their best throughput when t=0.8. The lower value of t, causes more packets to arrive at a faster rate and these packets are buffered. The sender sends these packets as soon as possible and does not need to wait for the packets to arrive from layer 5. The throughput for all the three protocol decreases with increase in loss, as expected.

For t=100: The throughputs obtained for ABT, GBN and SR for t=100 are shown in **Table 7** and the graph is plotted in **Fig 7**.

Loss	ABT	GBN	SR
0.1	0.008089	0.010154	0.01015
0.2	0.007479	0.01033	0.010066
0.4	0.00516	0.010017	0.00998
0.6	0.000916	0.010013	0.009881
0.8	0	0.005372	0.004645

Fig 7: Graph showing the performance comparison for ABT, GBN and SR at t=100.

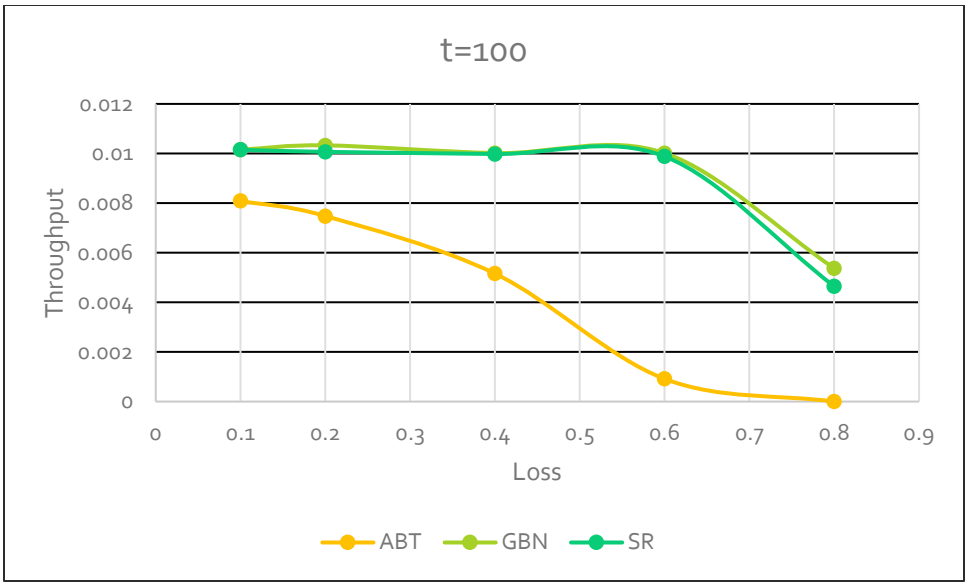

Fig 7: Graph showing the performance comparison for ABT, GBN and SR at t=100

The throughput for the 3 protocols is very low since the sender has to wait for long durations for the packet to arrive. The sender is thus idle for a long period of time and hence the throughput is low.

Experiment 4

The sample and expected RTT's obtained for the GBN protocol at various losses {0.1,0.2 and 0.6}.

Fig 8-10 show the graph between the Sample and Expected RTT for different loss probabilities.

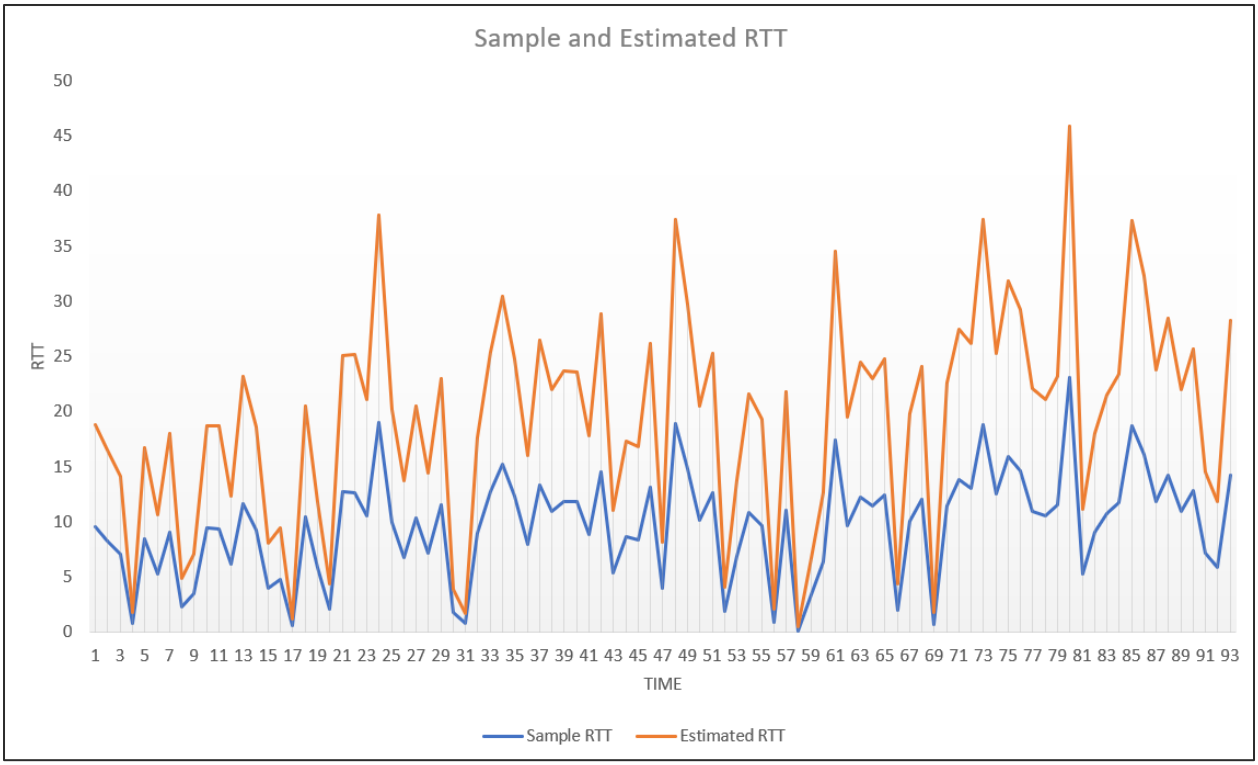

Fig 8: Sample and Expected RTT for loss =0.1.

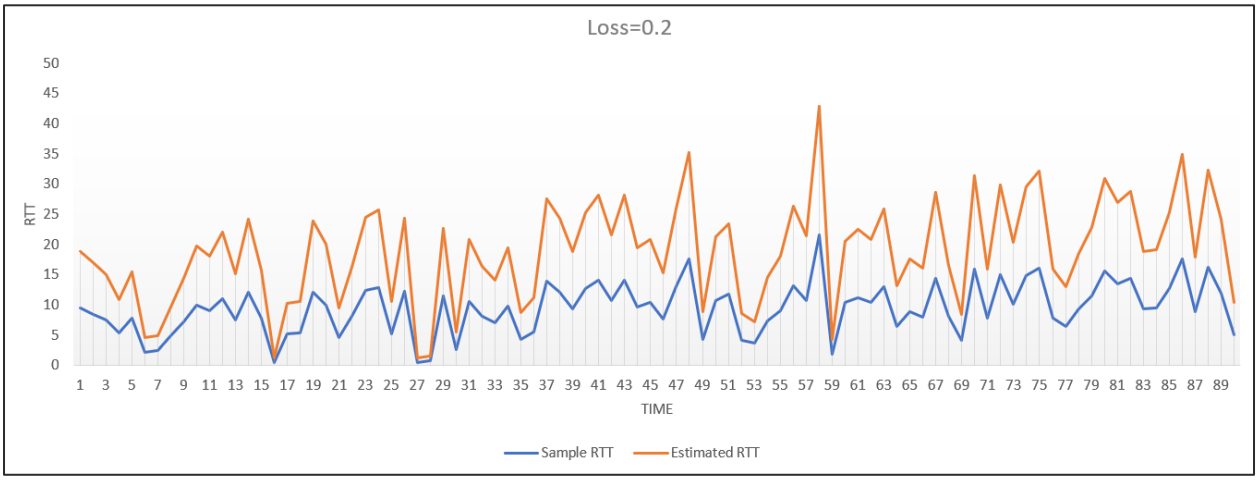

Fig 9: Sample and Expected RTT for loss =0.1.

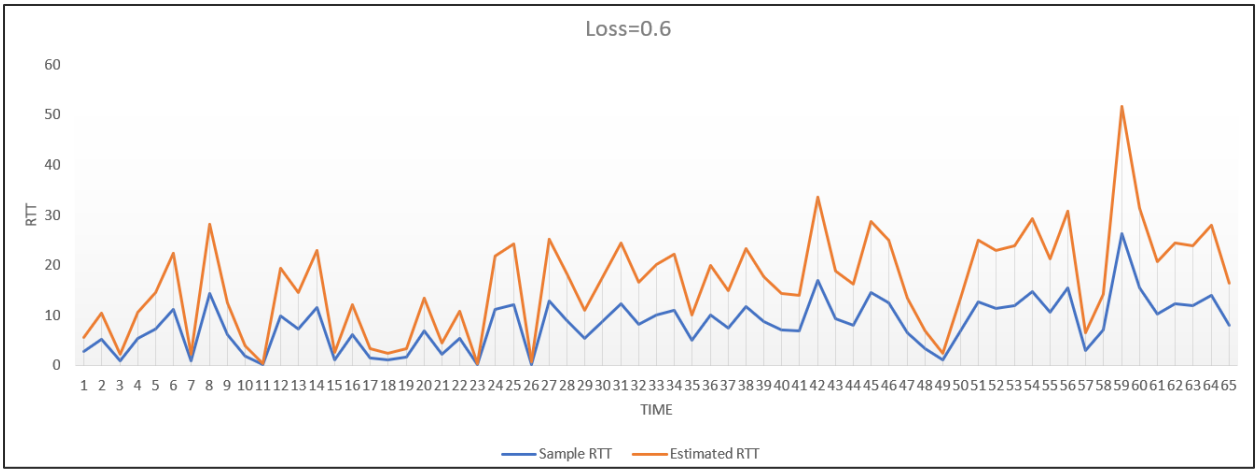

Fig 10: Sample and Expected RTT for loss =0.1.

Both the sample and estimated RTT show high variations. Ideally, the estimated rtt smooths out the variations in the sample rtt. However, our value of the parameters differ from the recommended values and hence, the graph above is different than what is usually expected.

Experiment 5:

Plot the throughput of ABT, GBN and SR for l=0.0 and c=0.0

Loss=0	ABT	GBN	SR
10	0.017492	0.01985	0.019691
50	0.017492	0.01985	0.019691
100	0.017492	0.01985	0.019484
200	0.017492	0.02067	0.019484
500	0.017492	0.02067	0.02067

Table 8: ABT, GBN and SR performance comparison for loss=0.0 and c=0.0

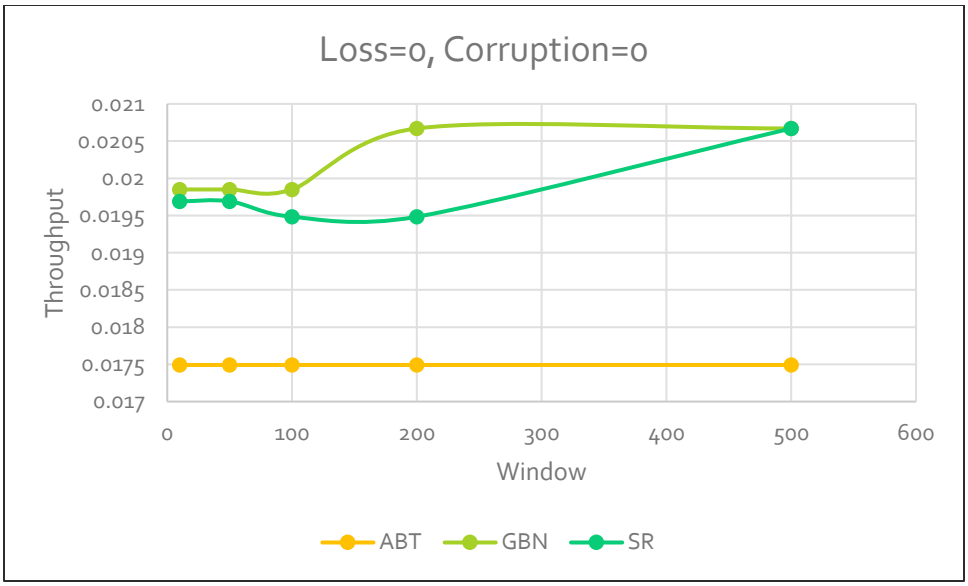

Fig 11: Throughput for ABT, GBN and SR for a reliable channel

For zero loss and corruption, the channel is completely reliable and hence the throughput so achieved should be the highest.

Experiment 6:

Plot the throughput of ABT, GBN and SR for l=0.0 and c={0.2, 0.4, 0.6 and 0.8}

corruption Rate	ABT	GBN	SR
0.2	0.014563	0.020417	0.019691
0.4	0.01044	0.001329	0.019484
0.6	0.003126	0.001201	0.019484
0.8	0.003074	0.001031	0.01067

Table 9: Throughput for ABT, GBN and SR for different corruption rates

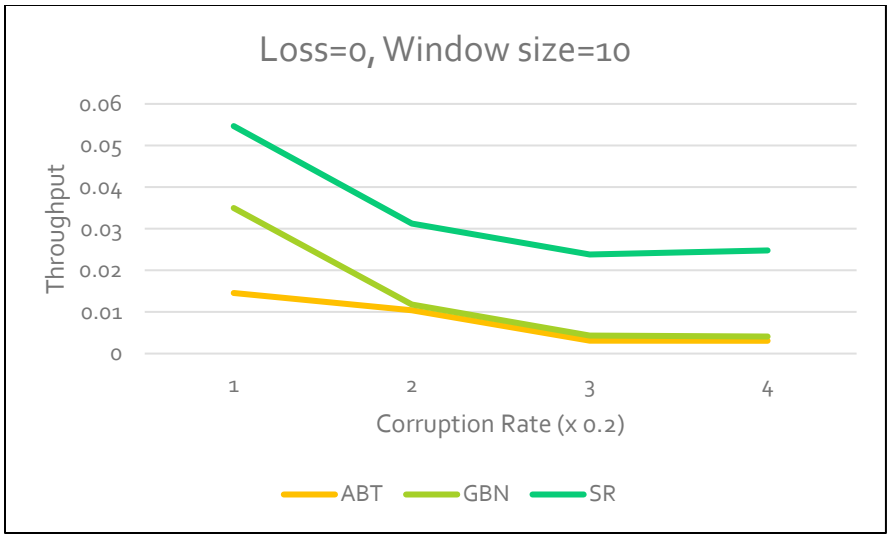

Fig 12: Throughput for ABT, GBN and SR for different corruption rates

From Table 9 and Fig 12, we can see that the throughput decreases as the corruption rate increases, for a loss of 0. This is in accordance with the fact that increase in corruption, increases congestion and hence decreases throughput.