CSE 589 Modern Networking Concepts (Spring '20)

PA2: Reliable Transport Protocols - Analysis, dt: 04/01/2020

Srisai Karthik Neelamraju (neelamra@buffalo.edu), 50316785

"I have read and understood the course academic integrity policy located under the link: http://www.cse.buffalo.edu/faculty/dimitrio/courses/cse4589_s20/index.html#integrity."

In this programming assignment, the following three reliable data transport protocols are implemented for facilitating data transfer between Hosts A and B in a simulator environment.

i. Alternating-Bit (ABT)
ii. Go-Back-N (GBN)
iii. Selective-Repeat (SR)

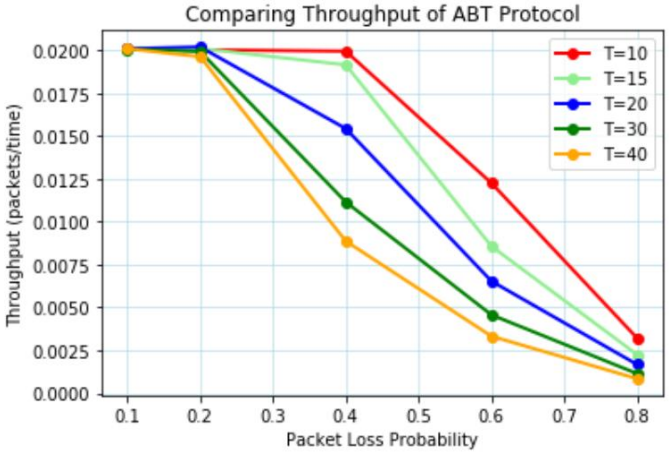
Later, the performance of each of these protocols is evaluated by performing experiments which use various combinations of simulator parameters like packet loss probability, packet corruption probability, sender window size, the number of messages sent and the average time between messages from the sender's application layer. Relevant observations and inferences are detailed.

For all the experiments described in this report, the following simulator parameters are in common:

i. Number of messages sent by Host A: 1000
ii. Mean time between messages from Host A's application layer: 50
iii. Probability of packet corruption: 0.2

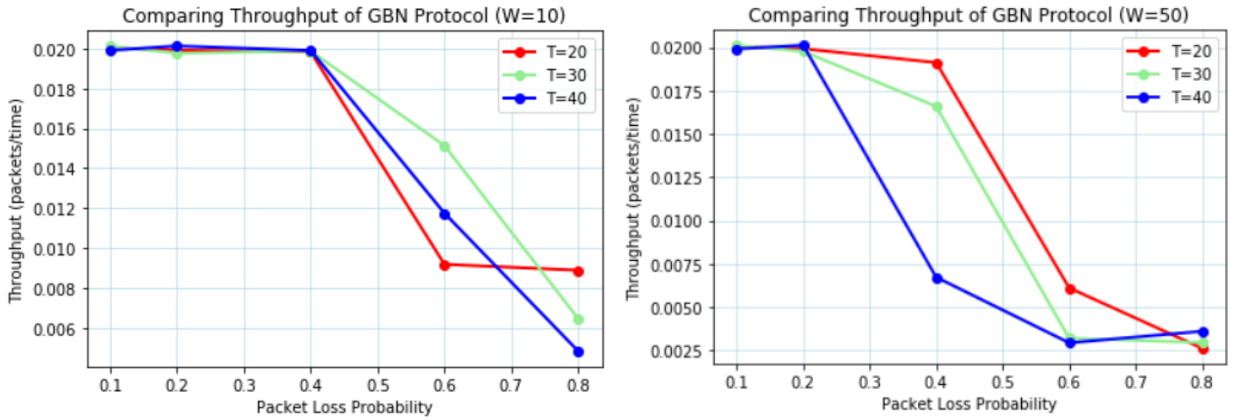
Timeout Schemes used in the Protocols

ABT: For deciding the timeout scheme in this protocol, multiple values of timeout (T) were tried. Specifically, for analyzing the throughput of the protocol, the timeout values 10, 15, 20, 30 and 40 were used, along with varying packet loss probabilities. The observed throughputs are as shown,



It is evident from the graph that using a timeout of 10 units produces the best throughput for the protocol, especially in scenarios with higher packet loss probability. Having a lower timeout value implies a greater number of retransmissions and thus, a higher chance of eventually not losing a packet. On the other hand, having a high timeout value in high packet loss scenarios means spending more time looking for an acknowledgement that has very low chance of being received. Consequently, the protocol gives better throughput when the timeout value is on the lower side. Therefore, a timeout value of **10 units** is used in the experiments for the ABT protocol.

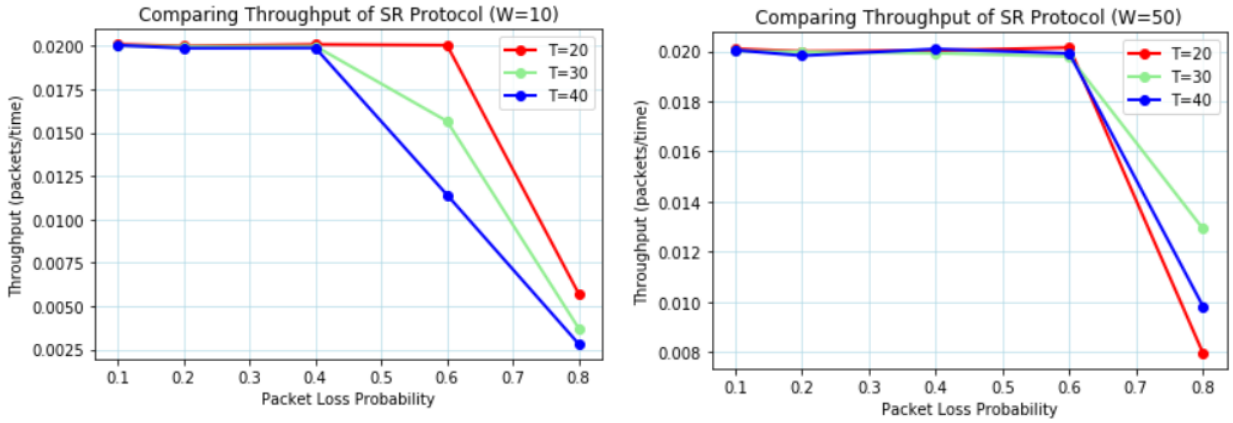
GBN: For deciding the timeout scheme in this protocol, once again multiple values of timeout (T) were tried. Specifically, since the average time required for a packet to reach the other side is 5 units when there is no other packet in the network, the timeout value 10 was not considered and only 20, 30 and 40 were evaluated. Moreover, the optimal timeout value in GBN protocol is not only dependent on the packet loss probability but also the window size. Thus, the performance was evaluated by varying both these parameters. The observed throughputs are as shown,



It is evident from the first graph that when the window size is 10, using a timeout of 30 units produces the best throughput for most of the packet loss probability values. However, a timeout of 20 units works best for the scenario where the loss probability is 0.8. On the other hand, when the window size is 50, a timeout of 20 units works better in all the situations. This could be possible because waiting longer for an acknowledgement means that more packets might be added in the meantime to the sender's window and any further loss in packets would lead to many more retransmissions. Besides, as can be observed from the graphs, different timeout values provided the best performance for different simulator parameters. For instance, for loss probability 0.6, window size of 10 and a timeout value of 30 units provided the highest throughput.

Since a relation between the packet loss probability, window size and the timeout value could not be obtained so as to develop an adaptive timeout scheme (discussed at the end), a constant value of **20 units** was chosen as the timeout for the GBN protocol. Besides, since RTT is approximately 10 units, using a value of 20 units as timeout seems alright as it also accounts for any deviations in RTT (as described in the lecture when discussing about determining TCP timeout interval).

SR: For deciding the timeout scheme in this protocol, once again multiple values of timeout (T) were tried. Similar to GBN protocol, the timeout value 10 was not considered and only 20, 30 and 40 were evaluated. Besides, since SR is also a sliding window protocol, the optimal timeout value is dependent on both the packet loss probability and the window size. Thus, the performance was evaluated by varying both these parameters. The observed throughputs are as shown,



It is evident from the first graph that when the window size is 10, using a timeout of 20 units produces the best throughput for all the packet loss probability values. On the other hand, when the window size is 50, a timeout of 20 units works better in all the situations except when the loss probability is 0.8. Besides, the throughput values in general follow a decreasing trend as the packet loss probability increases. Once again, different timeout values provided the best performance for different simulator parameters and an adaptive timeout scheme is necessary for the protocol to provide optimal performance in all possible scenarios. However, since a relation between the packet loss probability, window size and the timeout value could not be obtained, a constant value of **20 units** was chosen as the timeout in the experiments for the SR protocol.

Implementing Timers in the Protocols

ABT: For implementing this protocol, a single timer is required and it is started at the time of transmission of each packet. The timer is stopped only when an acknowledgement of the packet is correctly received by the sender. Further, if there any outstanding packets in the buffer, the oldest of them is transmitted by the sender and the timer is started again. If the timer expires or upon receiving a corrupt acknowledgement, the current packet is retransmitted and the timer is restarted.

GBN: As with ABT, implementing GBN protocol also requires the use of a single timer. The sender keeps a timer for the oldest unacknowledged packet, i.e., the base of the sender's window. If the sender receives an acknowledgement, the timer is stopped and it is started once again for the oldest unacknowledged packet (if any) in the shifted window. If the timer expires, all the packets in the sender's window are retransmitted and the timer for the base is restarted.

Upon receiving an acknowledgement for the base, in order to adjust the timer based on the now oldest unacknowledged packet, its transmission time has to be saved. For this purpose, the sender (host A) uses a buffer of the following struct type:

```
struct A_dtype{
        struct pkt packet;
        float start_time;
};
```

In the above struct, 'packet' corresponds to the packet that is being transmitted into the network and 'start_time' is the transmission time of the packet.

When the acknowledgement for base packet is received, the hardware timer is stopped and it is then restarted for duration equal to the difference between the actual timeout duration, and the difference between the current time and the start time of the new base, i.e.,

Timer Duration = Actual Timeout Duration – (Current Time – New Base Start Time)

SR: Unlike the previous two protocols, implementing SR protocol requires the use of multiple timers because the sender has to maintain a timer for each of the unacknowledged packets. However, the simulator provides only one hardware timer. Therefore, multiple logical timers are required to be implemented using this single hardware timer. For this purpose, the sender (host A) uses a buffer of the following struct type:

```
struct A_dtype{
        struct pkt packet;
        float start_time;
        int ACKed;
};
```

In the above struct, 'packet' corresponds to the packet that is being transmitted into the network, 'start_time' is the transmission time of the packet and 'ACKed' is a Boolean variable indicating whether the packet has received an acknowledgement yet.

Initially, the hardware timer is started for timeout duration when the first packet is transmitted and the start times of all subsequent packets are stored. If the sender receives an acknowledgement for a packet, its 'ACKed' field is set to 1. If this packet is the only unacknowledged packet in the sender's window, the hardware timer is stopped. If not, the hardware timer is stopped and then restarted for duration equal to the difference between the actual timeout duration, and the difference between the current time and the oldest start time among the unacknowledged packets (whose ACKed field is 0) in the sender window, i.e.,

Timer Duration = Actual Timeout Duration – (Current Time – Oldest Start Time)

If the hardware timer expires, the packet whose timer got expired is found by comparing the start time of each of the unacknowledged packets with the difference between the current time and the

timeout duration. For the packet whose timer got expired, these two quantities are equal. Once the packet is identified, it is retransmitted and the hardware timer is restarted using the same logic as above, for duration equal to the difference between the actual timeout duration, and the difference between the current time and the oldest start time among the unacknowledged packets.

Comparing Performance of the Three Protocols

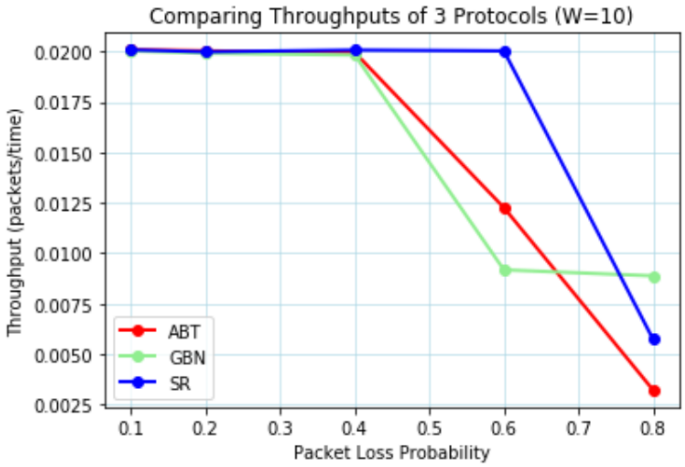
As mentioned previously, the following simulator parameters are in common for all experiments:

i. Number of messages sent by Host A: 1000
ii. Mean time between messages from Host A's application layer: 50
iii. Probability of packet corruption: 0.2

Since 1000 messages are sent from Host A with mean time between messages being 50, it takes around 50,000 units of time for Host A to send all the messages. In the best possible scenario, host B receives all the 1000 messages. This gives a throughput of about 0.02, which is near optimum.

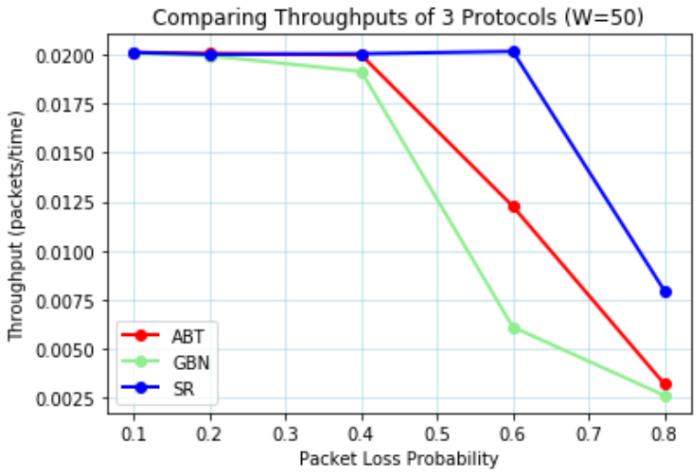
Experiment 1: Comparing the throughput for the three protocols with packet loss probabilities {0.1, 0.2, 0.4, 0.6, 0.8} and window sizes (for GBN and SR) {10, 50}

1. Window Size = 10



When the window size is 10, SR protocol gave the best performance among all the protocols for all loss probabilities except 0.8. GBN protocol gave the best throughput for this scenario with the highest packet loss. This could happen because excess packet loss in GBN means multiple retransmissions of packets and as a result, a higher chance of the packet correctly making it to the receiver. ABT protocol provided decent throughput except for the maximum loss case. Besides, the throughput of each protocol expectedly decreases as the packet loss probability increases. This behaviour is due to the extra transmissions of the lost packets. In addition, since the mean time between arriving messages is 50, a window size of 10 seems to be working just about fine. There is not much queueing at the sender, especially in scenarios with lower packet loss probability.

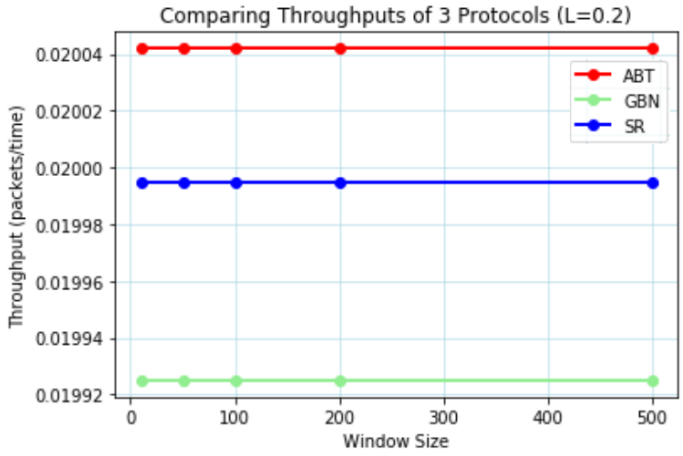
2. Window Size = 50



On the other hand, even when the window size is increased to 50, the performance of SR protocol is much better than the other two protocols. GBN fails particularly for scenarios with high packet loss probabilities and also falls behind ABT, because of how retransmissions work in the protocol. This is not the case with SR since only one packet is retransmitted each time. Once again, a general decreasing trend in the throughputs can be observed with an increase in chance of packet loss.

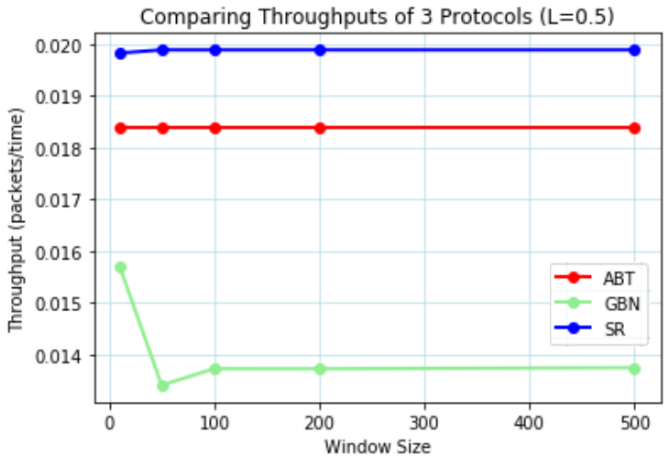
Experiment 2: Comparing the throughput for the three protocols with packet loss probabilities {0.2, 0.5, 0.8} and window sizes (for GBN and SR) {10, 50, 100, 200, 500}

1. Packet Loss Probability = 0.2



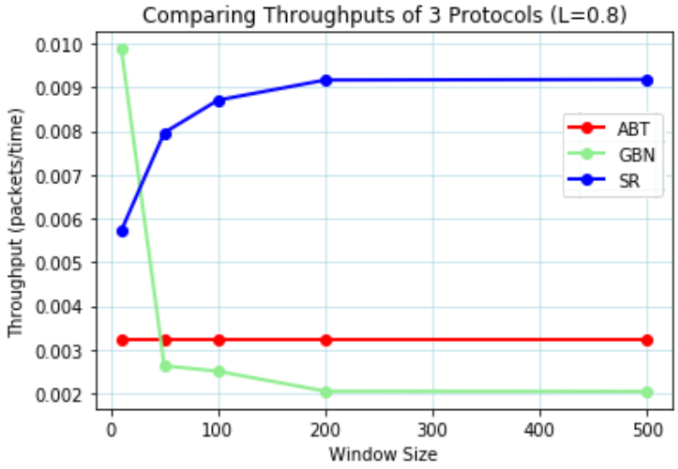
When the packet loss probability is 0.2, each of the three protocols gives similar performance and the throughputs are very close to the optimal value. Out of the three protocols, GBN gives the least throughput owing to the retransmissions of the whole window of packets in case of packet losses. ABT slightly edges SR because of the different timeout schemes in both the protocols. Besides, since the mean time between arriving messages is set to 50, there is not much improvement in the throughput of the sliding window protocols despite having bigger windows of sizes more than 50.

2. Packet Loss Probability = 0.5



When the packet loss probability is 0.5, ABT and SR protocols give similar performance and their throughputs are very close to the optimal value. Out of the three protocols, GBN once again gives the least throughput and it is very low as compared to the other two protocols. GBN performs moderately when the window size is 10 and the loss probability is 0.5. However, an increase in the window size has counterintuitively caused a drop in the protocol's throughput. As in the previous case, this could be attributed to the fact that packet losses using GBN protocol can cause excess retransmissions. Moreover, the number of retransmitted packets also grows with the window size. Thus, a higher window size value has caused the protocol to provide sub-optimal performance. SR protocol with window size 50 or more and a timeout of 20 units gave the best throughput.

3. Packet Loss Probability = 0.8



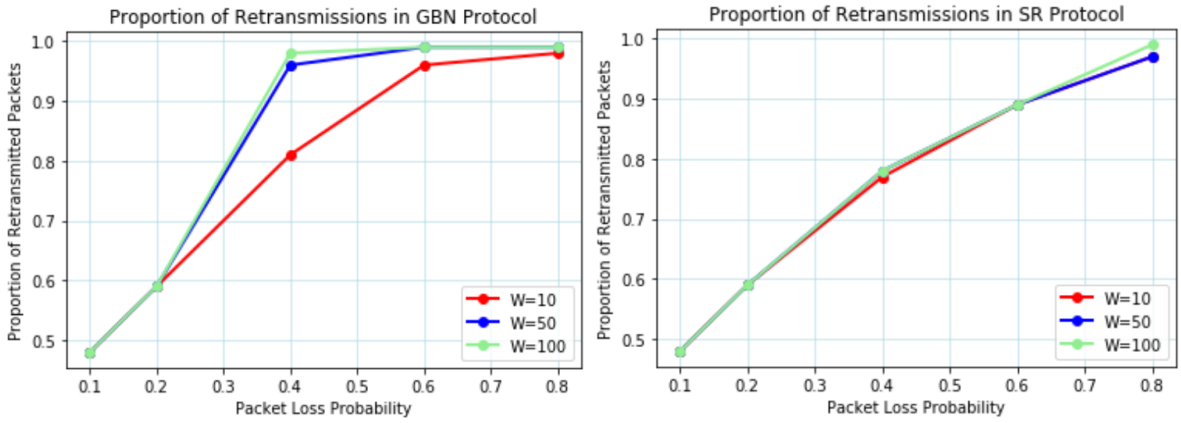
When the packet loss probability is 0.8, the throughputs of all the protocols are expectedly very less than the optimal value. Once again, GBN protocol gives the least throughput as its window size increases and ABT performs slightly better than GBN. This does not seem to be very intuitive as theory suggests that having a greater window size gives higher throughput, as happening with the SR protocol. It could certainly be possible that excess multiple retransmissions because of

higher window size values are impacting the throughput. On the other hand, SR protocol provides the best throughput among all the three protocols when the window size is 50 or more and unlike GBN protocol, its throughput increases with an increase in the window size. This shows that the retransmission scheme in GBN is making the protocol underperform. SR protocol with window size 500 and a timeout of 20 units gave the best throughput for this scenario.

An Attempt at using an Adaptive Timeout Scheme

Initially, an attempt was made to use an adaptive timeout scheme for the three protocols. The timeout value would be dependent on the packet loss probability in case of ABT and, in addition, on the window size in case of the sliding window protocols. The general idea was that having a lower timeout value would help in the scenarios with more packet losses and vice versa. Also, having a higher timeout value would help when the window size in the sliding protocols is bigger. However, since the sender would not know beforehand how the network behaves, it has to figure out a way to learn about the network parameters on the go and adjust the timeout value based on what it has learnt after a certain number of transmissions. The sender could also choose to periodically evaluate the timeout scheme based on the most recent behaviour of the network. For this purpose, the proportion of retransmitted messages at the sender was thought of as an indicator to the number of packets being lost by the network. However, it was not possible to determine how timeout could exactly be defined in terms of these values and how it varies. Thus, a relation involving the retransmission ratio, the window size and the timeout value could not be formulated.

By varying the loss probabilities and the window sizes for GBN and SR protocols, the ratio of the number of retransmitted packets to the total packets sent by the sender's transport layer was calculated and some intuitions were gained from the results summarized below.



Primarily, it was observed that GBN protocol had very high number of retransmissions when the window size is high despite the packet loss probability being low. For instance, for the scenario when loss probability is 0.4 and the window size is 50, this ratio is about 0.95 in the case of GBN protocol and about 0.75 for the SR protocol. Further, it was also inferred that window size had little effect on the ratio of retransmissions in the SR protocol. This behavior of the protocols is expected because of how retransmissions in the protocols are defined.