EECE7374 Programming assignment2

Reliable transport Protocols

Fan Fei, Wenzhe Li

li.wenzhe1@northeastern.edu, fei.fa@northeastern.edu

Academic Integrity

We have read and understood the course academic integrity policy.

Contents

1. Implementation

1.1 ABT

This is the basic protocol, the go back n protocol and selective repeat protocol will continue to implement based on this protocol. Therefore, we clarify the whole transportation process in this protocol.

For this protocol implementation, we define some functions to process more directly and conveniently.

make_packet: we use the function to make packet from message

get_checksum: this function is aimed to calculate checksum for the packet

Here we will simply describe our program, and for more details you can find the comment in the code file.

Since the interface is given, what we should do is to implement layer4 which is transport layer. Firstly, a packet needs to send to layer3 from layer4, A_output function needs to finish this task. Since it is a wait-and-stop protocol, each time it just sends one packet. So, if the message comes when A waits for ACK, the sender needs to wait and buffer this message. For that reason, we create two states (wait layer5 or wait layer3) and buffer to handle the wait-and-stop situation.

And after A receive the ACK packet, we should define this behavior in A_input function, the logic is that if we get right ACK, just change the state, and then send the message in the buffer, otherwise wait until timeout. When timeout happens, the current packet needs to resend.

For receiver B, it is easy to implement, if the packet is not loss or corrupted, send ack back to A. and then check for expected sequence number, if the receive packet is right, send the message to application layer.

We can find that ABT protocol waste bandwidth because of stop-and-wait mechanism. Therefore, pipeline technology is used in GBN and SR protocol, which improves bandwidth utilization.

1.2 GBN

For go back n protocol, it uses the pipeline technology. We just discuss the difference between go back n protocol and ABT protocol.

Since the go back n protocol use pipeline method, the message can be sent without wait, but there is a limit for no-wait transmission which is called window. The window contains the unacked packet and unsent packet. If the packet outside the window we should buffer the message. To send packet in order, we should send the buffer packet first since their sequence number is lower than current packet sequence number.

For go back n protocol, we set a timer for the whole window, every time sender A got ACK packet, just update the window base which is also called sliding the window.

When the timeout happened, we should resend the packet in the window to make sure the send packet is in right order.

For receiver, there is cumulative ACK, which means current packet and before packet is right received which means in order. Use this cumulative ACK number to sender, the sender can update the base to the new window.

Even if go back n uses the pipeline method to improve the bandwidth utilization, there is still a problem in go back n protocol. Here is one situation, if the window is larger, one packet

corruption or loss will cause to retransmit all the packets in the window. Since many packets are unnecessary retransmission. The performance will be affected.

1.3 SR

Selective repeat protocol only resends the wrong packet which is lost or corrupted in receiver. So, the main difference is in the receiver, in the receiver, whether the packet is right or wrong, just buffer the packet. And when the receiver receives the expected in order packet, deliver all the in-order packet to the application layer. It is obviously that, we should also set a window in the receiver part. And for the sender, we only resend the wrong packet which means we should set a timer for each packet instead of use one timer for the whole window.

Here we discuss more about how to implement multiple software timers in SR using a single hardware timer.

When a packet is sent from host A to host B, the time will be recorded in a list and the sequence number of the packet is also recorded in a list with FIFO order. The timer will be started if there is only one packet in the packet sequence number order list. When host A receive the acknowledgement, if the acknowledgement number matches the expected sequence number in the packet order list, it will advance to the next unACKed sequence number. Then, the timer will be restarted with the time remaining. The time remaining can be obtained by subtracting the time elapsed since the packet was sent from the set time to live. The time elapsed can be simply calculated by subtracting the current time with the time the packet was sent. Similar in the timer interrupted situation, the timer will be restarted in the same manner.

2. Timeout scheme

From the instruction file, we know that a packet sent into the network takes an average of 5-time units to arrive at the other side when there are no other messages in the medium. Therefore, we can get RTT = 10-time units which means roundtrip time.

But when we transfer packet from A to B, there are more processing such as make packet in sender and receiver, queued delay, and checksum. These will cost additional time in real transportation.

Firstly, the timeout setting must be larger than RTT, otherwise each packet need resend since the timeout always happen before the ack packet go back to sender.

For ABT protocol, since this is a stop-and-wait protocol, we do not except large timeout. The reason is that if many packets is lost or corrupted, the packet should wait the whole timeout and then resend. During this time, if the timeout is high, it will cost more time. In summary, too short time which is shorter than RTT will cause too much retransmission and long time will cause wasted time because of stop-and-wait mechanism.

Therefore, we set the timeout a little larger than RTT which means we expect the timeout as short as possible under normal condition. After some test, we find timeout = 20 can get good performance.

For go back N protocol, the timeout setting is like ABT protocol. However, there is a problem, if the window size is large, if timeout, we need resend all the packets in the window. So, for this situation, we can take high timeout for big window size. In the experiment part, we take the same timeout with ABT protocol.

For selective repeat protocol, it solves the window size problem in GBN protocol, so the timeout, we set the same timeout with ABT protocol.

3. Experiments

To compare the performance for these three different protocols, we follow the instruction file and run two experiments.

3.1 Experiment 1

For experiment 1, we set the experiment environment below.

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.

3.1.1 window size = 10

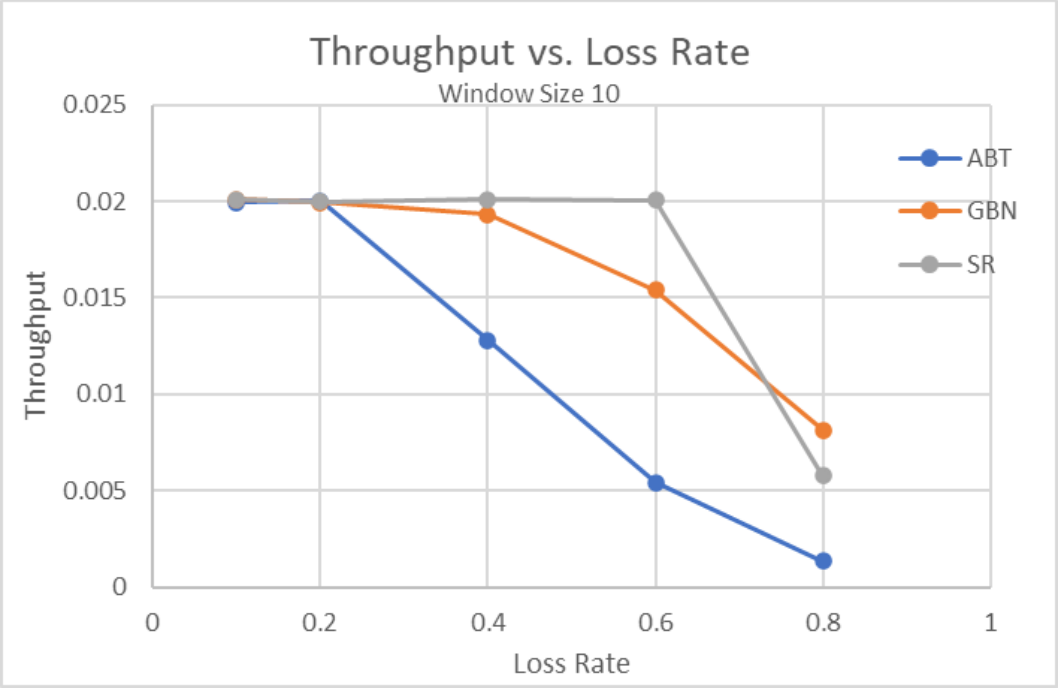

Figure 1: throughput -loss rate for window size = 10

Observation:

As shown in figure 1, we can find all of three protocol performance decreases with the loss rate increase, and for the throughput performance, SR > GBN > ABT.

Explanation:

Since the loss probability is higher, there are more packets which are needed to resend. For each protocol, it will decrease the performance when the loss rate increases. SR and GBN performance are better than ABT, since these two protocols use the pipeline mechanism to increase the bandwidth utilization. Also, SR has the best performance, since it has two buffer windows in both of sender and receiver side, and it just resend the wrong packet instead of being like GBN to resend all the packets in the window which contains some unnecessary packets.

3.1.2 window size = 50

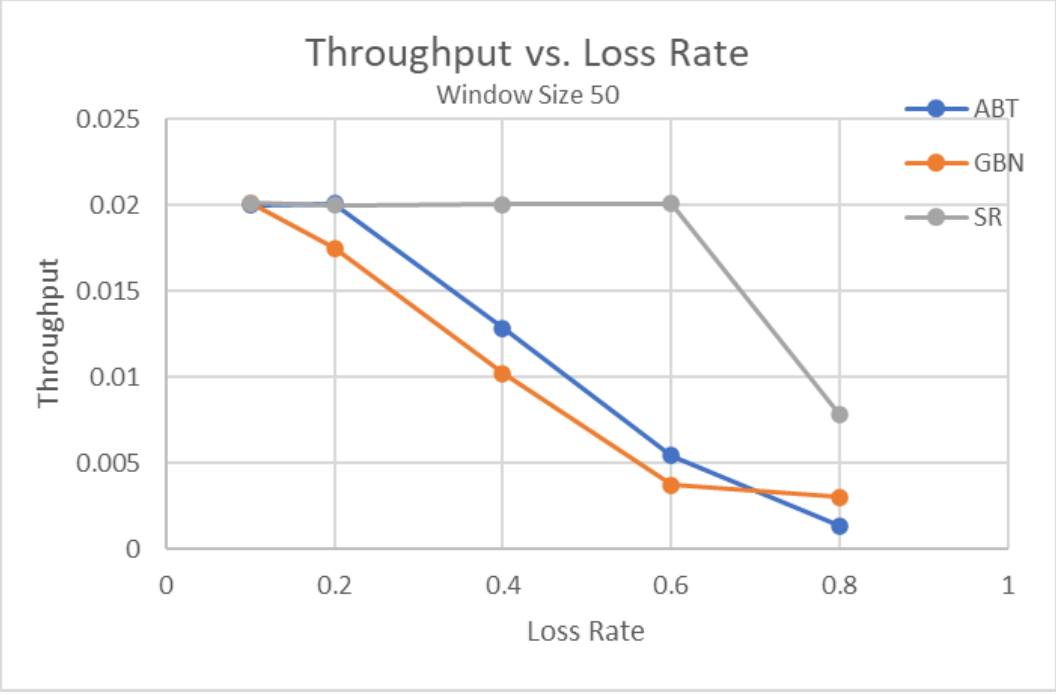

Figure 2: throughput – loss rate for window size = 50

Observation:

This result in figure 2 for ABT and SR is the same as figure 1, but the GBN protocol performance has changed a lot.

Explanation:

As we have discussed in the implementation part, the GBN protocol will resend all the packets in the window if a packet is lost. For this situation, we have a larger window size which will cause to resend more unnecessary packets. Therefore, the figure 2 GBN performance is worse than the figure 1.

3.2 Experiment 2

For experiment 1, we set the experiment environment below.

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.

3.2.1 loss possibility = 0.2

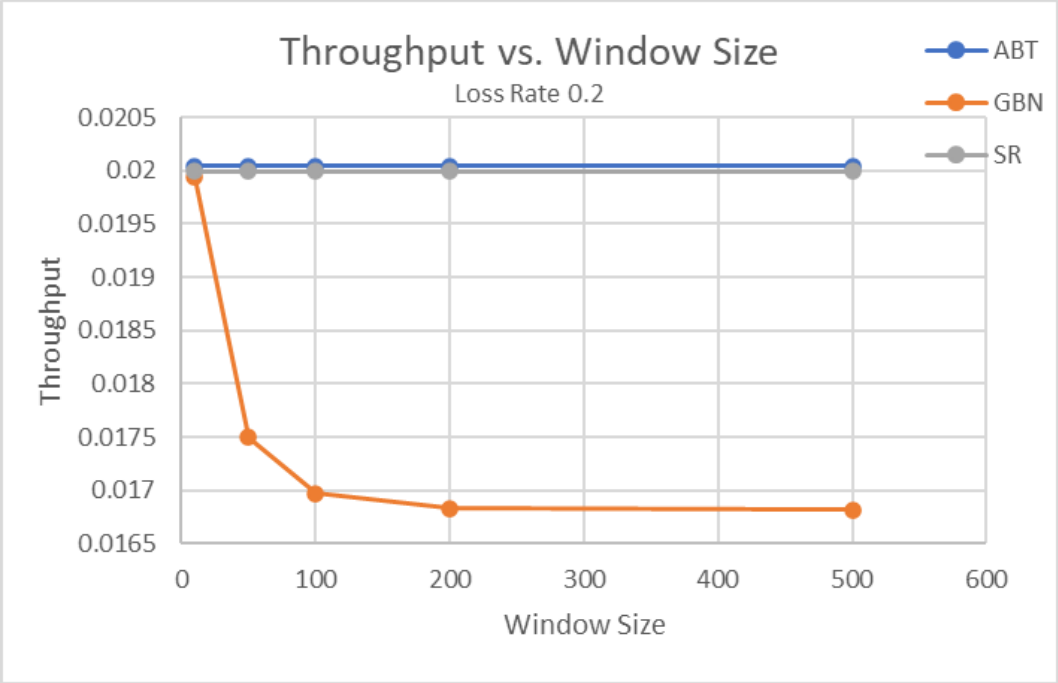

Figure 3: throughput – window size for loss probability = 0.2

Observation:

As shown in figure 3, SR and ABT performance keep steady when the window size increases, and the GBN performance decrease quickly and then keep steady. Also, the SR and ABT performance is much better than GBN

Explanation:

Since ABT protocol use stop-and-wait mechanism which means it has fix window size = 1, change the window size will not affect the ABT performance. But for GBN protocol, the window size is large, if a packet is lost, it will resend many unnecessary packets which decrease the performance quickly. For SR protocol, it uses the buffer window both on sender and receiver, so it can just resend loss or corruption packet which save time efficiently in big window size.

3.2.2 loss possibility = 0.5

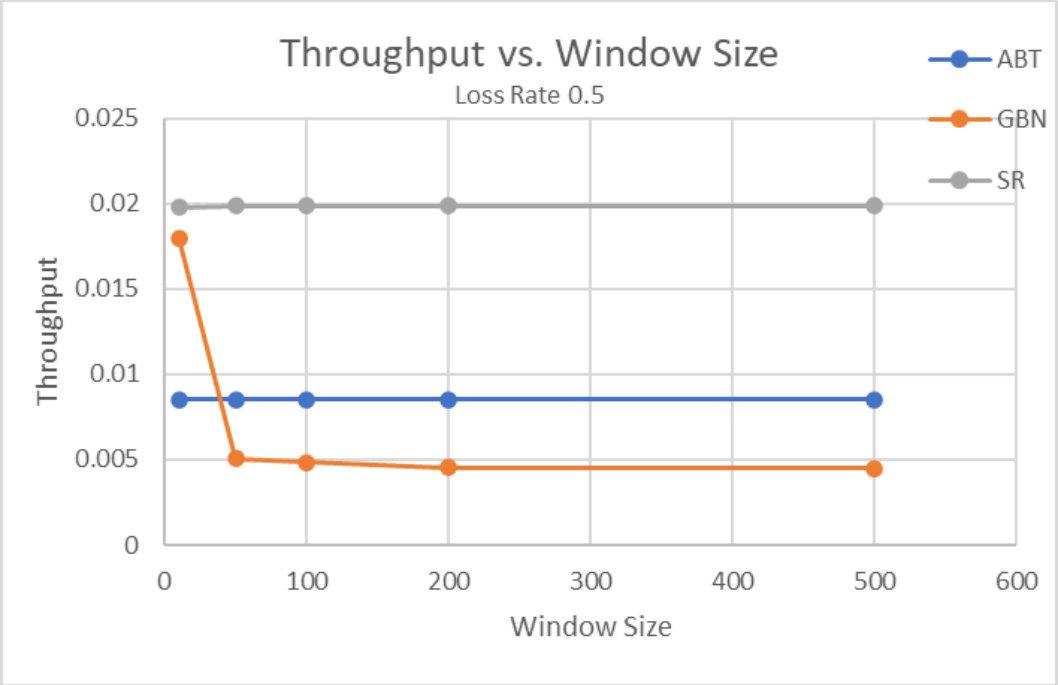

Figure 4: throughput – window size for loss probability = 0.5

Observation:

The trend in figure 4 is same as figure 3. When the loss rate is high, the ABT and GBN performance significantly decreases. However, the SR performance is also good even if the loss rate is high.

Explanation:

The reason is same as 3.2.1, and additionally, we find the GBN performance is very bad, since the loss rate is higher, the resend frequency will be higher, when the window size is large which is 50, 100, 200, 500, the GBN performance is bad.

3.2.3 loss possibility = 0.8

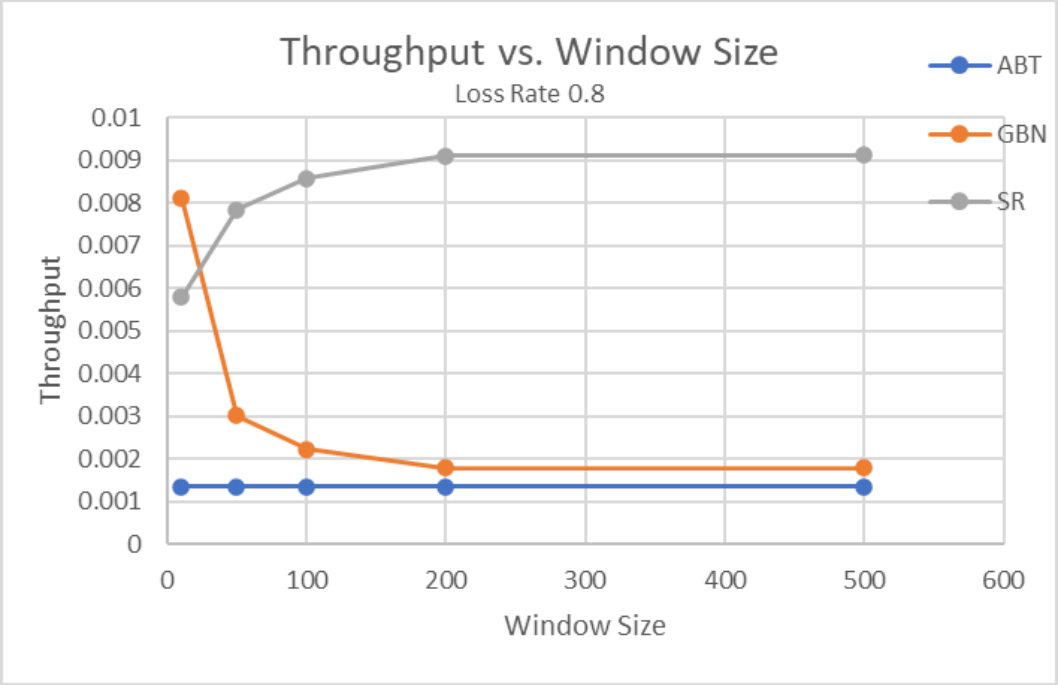

Figure 5: throughput – window size for loss probability = 0.8

Observation:

The ABT and GBN performance trend is same as figure 3 and 4. Since the loss rate = 0.8, that is very serious loss, all the three protocols performance significantly decreases. However, SR performance will be better when the window size is larger.

Explanation:

Even if the SR performance is low when the window size is small, when increase the window size, both the sender and receiver can buffer more packets which will conflict the high loss probability situation.