Reliable Transport Protocols

Vaibhav Narayan Chincholkar
vchincho@buafflo.edu
UBIT person number: 50290169

Contents

1 **Academic Integrity** **3**

2 Timeout scheme **4**
- 2.1 ABT . . . 4
- 2.2 GBN . . . 4
- 2.3 SR . . . 4

3 Experiment: 01 **5**
- 3.1 Window size=10 . . . 5
- 3.2 Window size=50 . . . 6

4 Experiment: 02 **7**
- 4.1 Loss probability: 0.2 . . . 7
- 4.2 Loss probability: 0.5 . . . 8
- 4.3 Loss probability: 0.8 . . . 9

5 Buffer Implementation **10**

6 Checksum Implementation **10**

7 Window Implementation **10**

8 References **11**

1 Academic Integrity

I have read and understood the course academic integrity policy.

2 Timeout scheme

2.1 ABT

I have used timeout as 20, it came from trial and error, first I started with value 1 and gradually increased the value till 25 and found out that the 5 is the single side propagation time so after considering processing time on both side, it came down to 20.

2.2 GBN

I started with value 20 but due early timeout retransmission rate was getting high so I increased the values gradually and optimized value selected as 30.

2.3 SR

In Selective repeat we have to maintain timer for each packet sent, therefore we need N packet for window of size N. But we only have one hardware timer. To implement multiple logical Timer I am using timer for interval of 1 unit. And counting how many intervals have occurred in the system. Basically that give us a time which can used for packet. In my code I have saved in current_timer variable. And used timeover variable associated with every packet. Timeover is basically current time + Timeout. In timer_interrupt I am checking if packets timeout is occurred if it is then I am sending that packet to B.

3 Experiment: 01

3.1 Window size=10

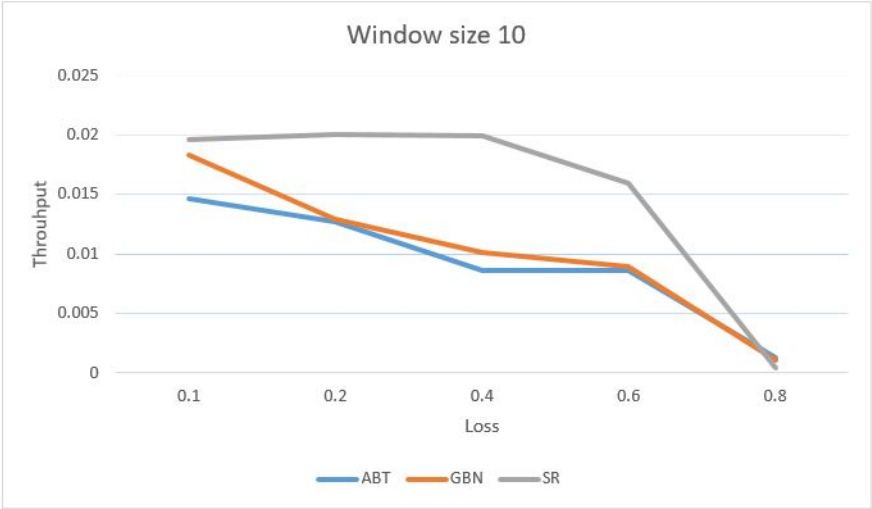

Figure 1: Throughput/loss for window size 10

Observation and Analysis:

In above graph we can see that the SR through put is better than GBN and ABT, as loss increases the number of transmission also increases in GBN that is why we see close graph line between ABT and GBN, whereas SR transmission only takes for unacked packets of window so we get better performance is SR, but when we increase the loss it basically degrades to stop and wait so we merging of all the graph line at 0.8 loss.

3.2 Window size=50

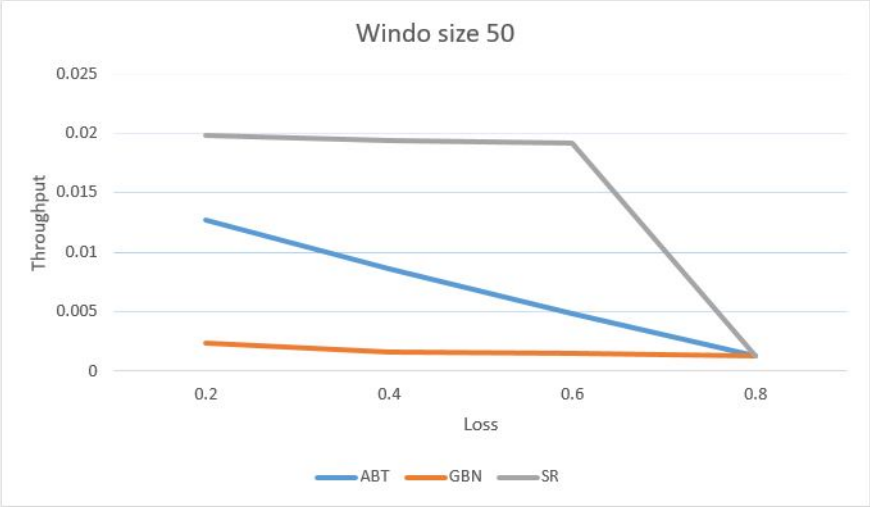

Figure 2: Throughput/loss for window size 50

Observation and analysis:

Now here we can see that GBN throughput decreases a lot as we increase the loss probability as window size is 50 we have to send lot of packet to B which decreases the throughput. In case of SR it performance remains the same.

4 Experiment: 02

4.1 Loss probability: 0.2

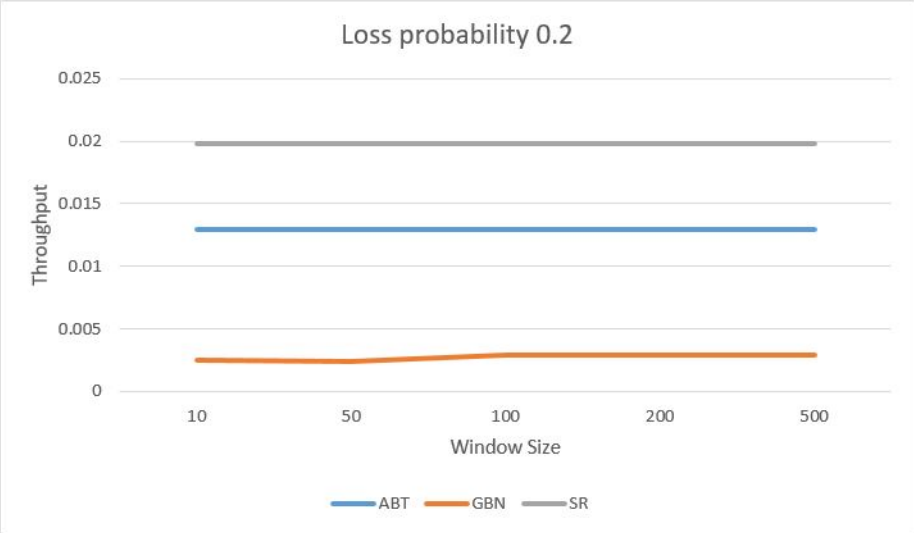

Figure 3: Throughtput/window size loss probability 0.2

Observation and analysis:

Here we can see at loss probability 0.2 we have same through put for variation of window sizes, we can conclude that the window sizes at low rate of loss probability window size does not affect the throughput.

4.2 Loss probability: 0.5

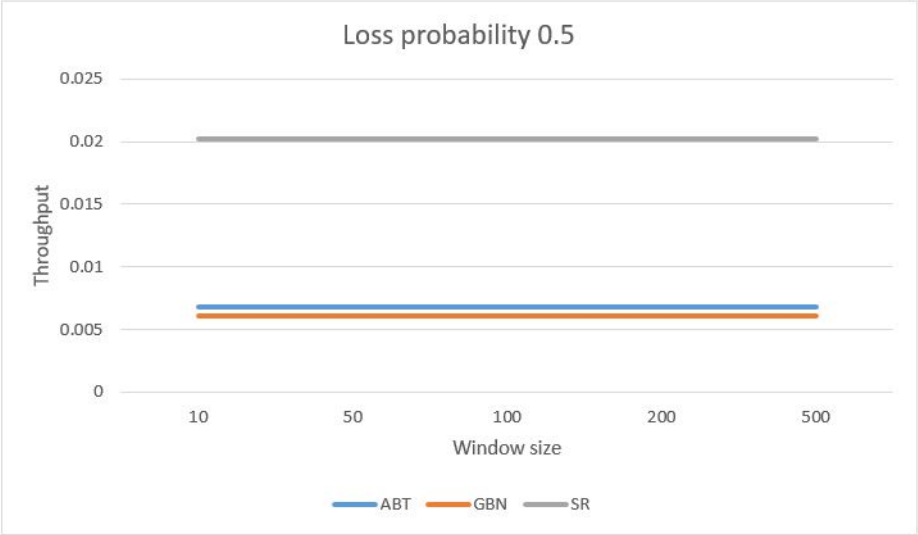

Figure 4: Throughtput/window size loss probability 0.5

Observation and analysis:

Here we can see at loss probability 0.5 we have same through put for variation of window sizes, we can conclude that the window sizes. Though we can see as due to large number of retransmission throughput of the GBN is decreased.

4.3 Loss probability: 0.8

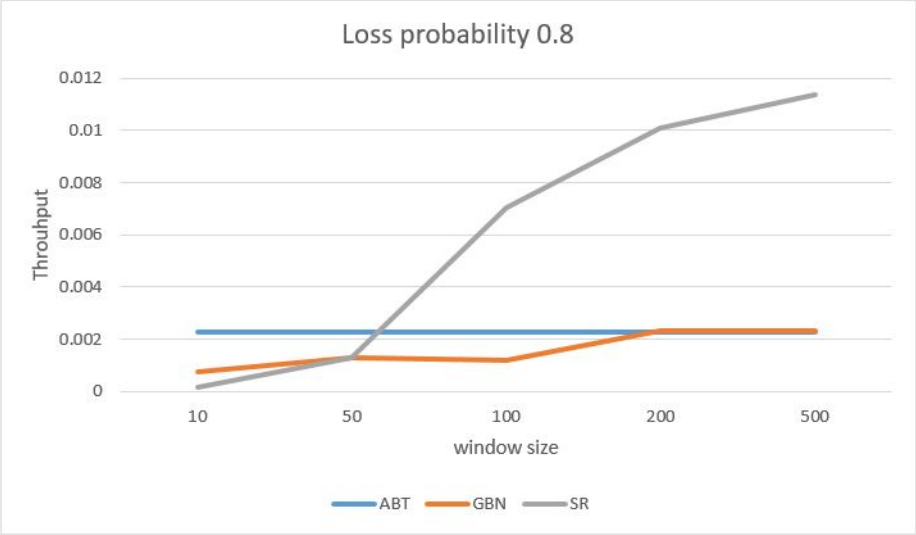

Figure 5: Throughtput/window size loss probability 0.8

Observation and analysis:

Here we can see at loss probability 0.8 GBN performance decreased dramatically as we are retransmitting same packet multiple times. Though SR initially due to high probability gives low throughput but catches on as we increase the window size, here ADT is not affected due to stop and wait.

5 Buffer Implementation

When a new message comes from layer5, we may not be able to serve that message at that time because sender might be waiting for last message ack or there is no space in window (in GBN and SR). At this time, we need to store this message to a buffer. Then when the window size decrease, we can send the message again. We will use a single linked list to implement the extra buffer. The structure for each element is:

struct node
{
struct msg message; //store the message
struct node *next; //point to the next element
};
We defined two method to store and retrieve the message from the linked list. We defined append_msg(message) to store the message and pop_msg() to retrieve the message from the linked list.

6 Checksum Implementation

A checksum is a small-sized datum derived from a block of digital data for the purpose of detecting errors which may have been introduced during its transmission. We defined a method calc_checksum() which calculate check sum for the packet. We are considering packet's payload, sequence number, acknowledgment number, for calculating checksum. We are basically performing addition of all this component and passing it as a checksum in packet.

7 Window Implementation

We are using array of packets, initially we are creating a pointer of pkt. We allocate the memory in init method. We use get getwinsize() to get the window size specified and allocate that much memory and use pointer to point to that location. We use concept of ring window so that the base and last pointer keep on moving in circular fashion. We increment pointer as follows Pointer=(pointer+1)%window_size This keeps pointer in window size and we store the packet to be transmitted in that ring window

8 References

https://github.com/Santosh01/Reliable-Transport-Protocols
https://en.wikipedia.org/wiki/Checksum