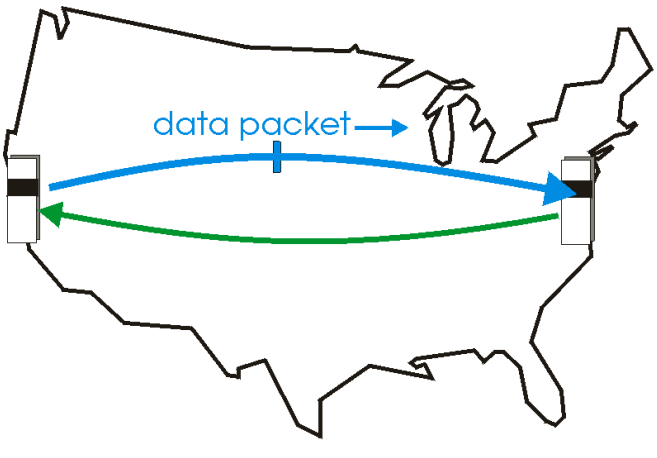
I have read and understood the course academic integrity policy.

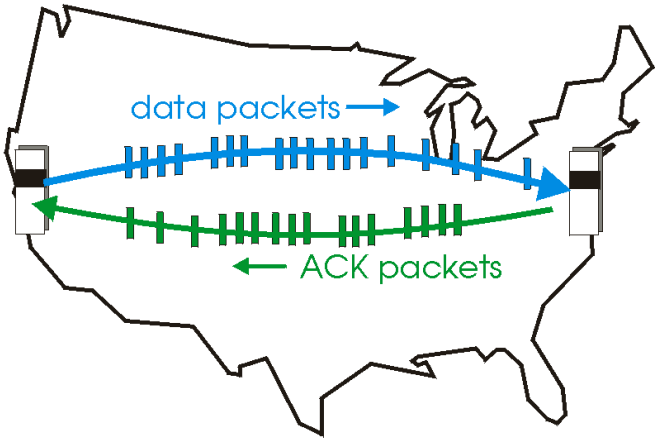
CSE 489/589 PA2 Report

Reliable Transport Protocols

Yincheng Jin, Jiyang Li 11/10/2018



(a) a stop-and-wait protocol in operation



(b) a pipelined protocol in operation

Timeout Scheme

We choose *TRAVEL_TIME* to be 20 time units for all three protocols.

A packet sent into the network takes an average of 5 time unites to arrive at the other side when there are no other messages in the medium. So it needs 10 time unites as the RTT in ideal situation. For real situation, the delay of the packet delivery can caused by many reasons, like transmission, propagation, processing and queuing. So extra time unites are needed for a successful rdt protocol.

We tried {15, 20, 25} time units for ABT, GBN and SR and we found out that 15-time-units will trigger more retransmission and 25-time-units is not sensitive enough to detect the loss. Both cases cause the throughput less than 20 time units. So 20 time unites is decided as our *TRAVEL_TIME* as the result of the simple pilot study.

Method Description

- Global Variables

int *msgs* - number of messages come from layer 5 of A side.
int *count_msg* - number of messages sent to layer 3 on A side.
int *seq* - sequence number for making packet on A side.
int *exp_seq* - expected sequence number of received packet on B side.
struct pkt *msg_pkt* - pkt structure made on A side in ABT protocol and array of pkt sturcture made on A side for GBN and SR.
struct *pkt ack_pkt* - pkt structure made on B side.
struct msg *buffer* - array of msg structure.

There are variables used for different protocols. *base* and *next_seq* are used in GNB and SR for pipelined transmission. For ABT, we use *stop_wait* to indicate a state that whether there is a message in transit, there is no state indication of for GBN because it neither should stop_and_wait nor manipulate the timer for each packet. For SR, *timevalue* is used to record the absolute time, *ackval* and *b_ack* are the states of acknowledgement on A side and B side separately, and *rcv_pkt* for buffering the received packet on B side.

- How Does Timer work for SR

The timer works based on the function *starttimer* by setting the correct increment.

i. Record the absolute time *timevalue[next_seq]* for each packet output from A side and set the *ackval[next_seq]* to ZERO;

ii. Start timer when *next_seq == base*;

iii. Set the *ackval[next_seq]* to ONE and *stoptimer* when input the expected packet on A side, and move the window to the next packet until the *ackval[base]* of this packet is ZERO;

iv. *starttimer* for this packet with the increment (*TRAVEL_TIME* - *(get_sim_time()* - *timevalue[base])*

v. When time interrupt:

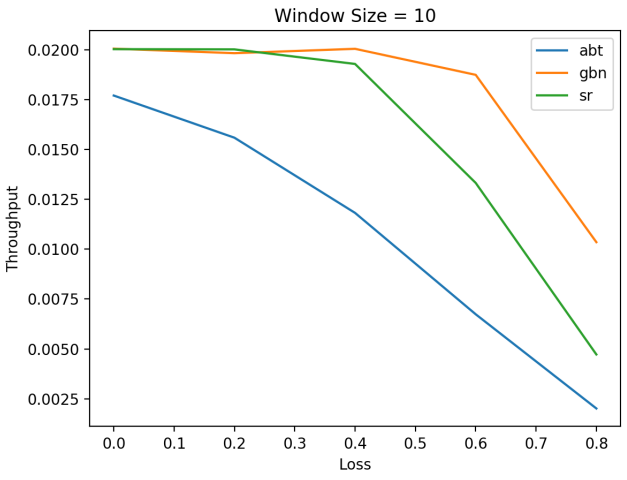
a. Locate the packet *index* with the smallest *timevalue* and with *ackval* equals ZERO (the earliest unacknowledged packet), and update its *timevalue* with *get_sim_time();*

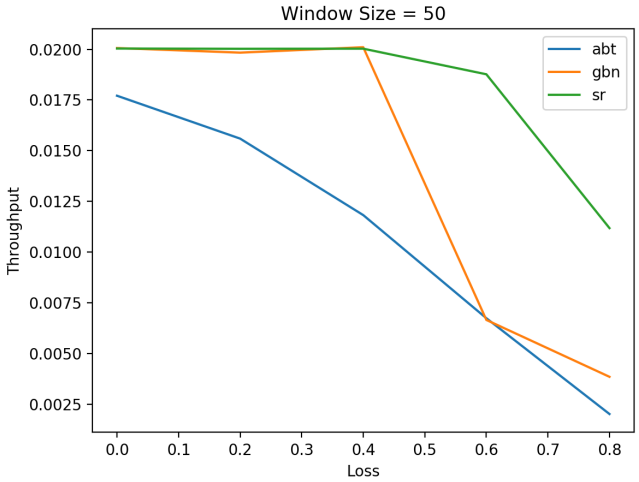
b. Locate the *index* with the smallest *timevalue* in the updated timevalue array and with *ackval* equals ZERO, then *starttimer* for it with the increment *(TRAVEL_TIME - (get_sim_time() - timevalue[index])).*

Experiments

I.

Window size 10 and 50			
Protocol	**Message**	**Corruption**	**Loss (X-axis)**
Alternative-Bit-Protocol	1000	0.2	{0.1, 0.2, 0.4, 0.6, 0.8}
Go-Back-N	1000	0.2	{0.1, 0.2, 0.4, 0.6, 0.8}
Selective-Repeat	1000	0.2	{0.1, 0.2, 0.4, 0.6, 0.8}



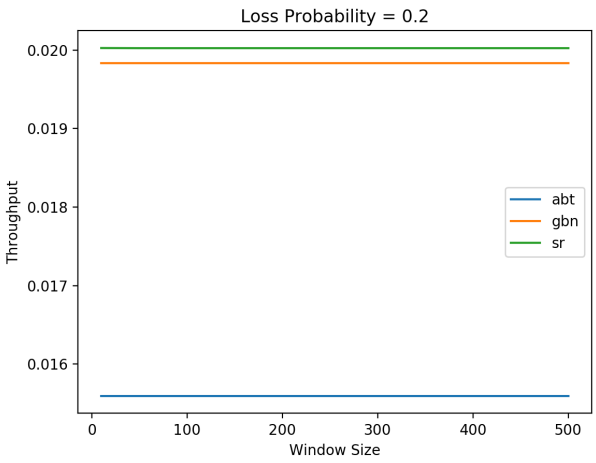


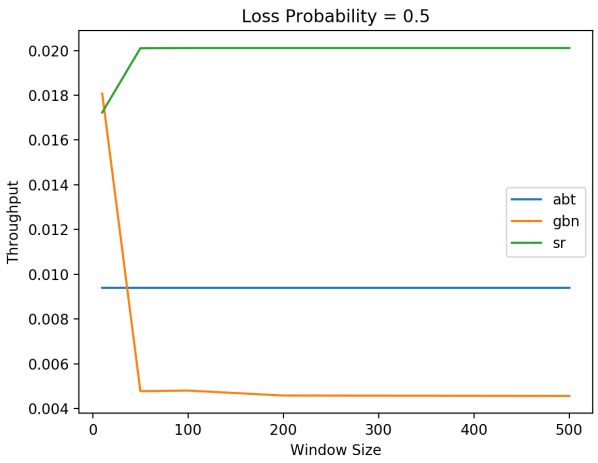
The results of the experiments show as expected a decreasing trend in throughput as the loss level increases. The results of ABT in above figures have no relation with window size, so they are the same in those two figures. GBN works better with window size 10 than 50 because increasing the

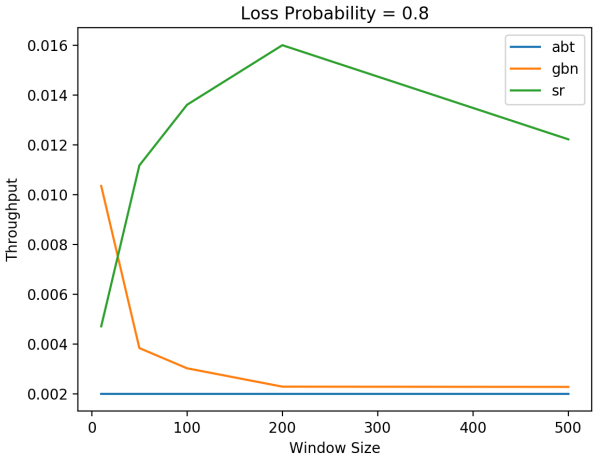
window size aggravates the load of retransmission process so that decreases the throughput. On the contrary, SR results in a larger throughput when the window size is larger because the efficiency is improved under the same *TRAVEL_TIME*. In conclusion, with window size 10, the performance GBN > SR > ABT, with window size 50, the performance SR > GBN > ABT except when the loss probability equals 0.2 for both cases. The performance of GBN drops sharply with loss probability from 0.6 to 0.8 with window size 10, and this process comes later with window size 50. So GBN is sensitive to the loss probability when the window size is relatively large, and could be more robust with smaller window size. And on the contrary, SR is more robust with relatively large window size.

II.

Loss probability 0.2, 0.5, and 0.8			
Protocol	**Message**	**Corruption**	**Window Size (X-axis)**
Alternative-Bit-Protocol	1000	0.2	{10, 50, 100, 200, 500}
Go-Back-N	1000	0.2	{10, 50, 100, 200, 500}
Selective-Repeat	1000	0.2	{10, 50, 100, 200, 500}







From the above three figures, we can see that SR does a better job under most of conditions compared with other two, and it reaches the highest throughput in all three figures. The performance of ABT has no relation with window size. And when the loss probability is small enough, like 0.2, the other two protocols also do not influenced by the window size. But the window size should be considered when the loss probability increases. The same as the result from Experiment I, GBN and SR have the opposite performance toward different window sizes. The performance of GBN drops sharply when increasing the window size. When the window size is larger than 50, the throughput of GBN is too small to be influenced. On the contrary, the performance of SR comes out better when increasing the window size. However, the performance of SR could be damaged by increasing the window size for a larger probability, as shown in the third figure after window size is larger 200.

In conclusion, ABT protocol is simple to implement and has no relation with window size, but the performance is poor. GBN protocol works as good as SR with small window size and can even be better. SR is the best in general but should be careful with the window size when working with a network with loss probability larger than 0.2, because the throughput could be damaged largely.