Reliable Transport Protocols

Ziang Li, Lei Chen
ziangli@buffalo.edu, lchen76@buffalo.edu

Academic Integrity

I have read and understood the course academic integrity policy.

Multiple logical timer on single hardware clock

In computer, timer is set by specific clock chip. In this project, we need more than one timer to record packages' time of interruption, so we need to use one timer which built in computer to simulate multiple timers.

In this project, we inherit the package class, and base on that, we overwrite it to add up call_time variable. Call_time is set to record the sum of current time and time out interval, which means it stands for the time which this package should be retransmitted is no ack is received. In our design, we put all of package which in sending queue into a vector, and make call_time variable record the time of interruption. This keeps a timeline. After received ack, or exceeded time out interval, time line would be updated, corresponding package would be handled, removed from sending queue or retransmitted, which depend on it get its correspondingly ack or not. Update function is designed to grab the minimum call_time of timeline, subtract current time to be second parameter of starter of timer. This make the project have multiple logical timer on single hardware clock.

In this project, selective repeat algorithm is implemented as above. Go-back-n protocol, however, is optimized and simplified by the features of go-back-n algorithm. Due to the first-in-first-out mechanism in sending list, the first package's call time should be always closest to current time. So when updating the timeline, we only need to grab the very first call time of waiting list after updated the sending list.

Timeout scheme

As average roundtrip time is given, assumption can be made that a message can get its ack after being sent in roundtrip time. If roundtrip time passed after the message is sent, and get no ack, then a conclusion can be made that maybe the message is lost or ack is lost. Then we need to resend the message. So that time out interval is set intuitively larger a little than round-trip time as there maybe some delay happens during the transmission and processing of package of each host end.

About setting of specific time of timeout, in alternative bit protocol, value of time out interval affects transmission efficiency directly. Too short time out interval would attributes unnecessary retransmission, and too long time out interval would decrease transmission efficiency either due to its mechanism of stop-and-wait. Therefore, time out interval should be as short as possible under the condition of slightly longer than round trip time to ensure the highest efficiency.

In selective repeat protocol, theoretically, if buffer of receiver is big enough, longer the round trip time, better the performance. Because longer round trip time can let package without mistake transmitted first and set all of packages which need to be retransmitted to the last of sending list. This can make the highest utilization of window size, but drawbacks of it is obvious too. Buffer of receiver side should be big enough, and due to receiver only send packages to layer 5 when receiving package without loss, if retransmission not occurred, layer 5 would have a delay on receiving messages. Therefore, although it looks like longer time out interval increases

transmission efficiency, considering the actual application, time out interval need to be set as a reasonable value.

In go-back-n protocol, situation is similar to alternative bit protocol. Due to the feature of go-back-n algorithm, too short time out interval would trigger unnecessary retransmission, too long time out interval would attribute to decreasing of transmission efficiency because of extra waiting time.

Windows Size

Window is the largest number of packages which be sent simultaneously in go-back-n protocol and selective repeat protocol. For alternative bit protocol, window size can be viewed as 1, which means only one message can be sent as a time.

For selective repeat, larger the window size, higher the efficiency. Because each of sending of package is relatively independent, each of package has its own timer, mistake and retransmission of one package barely affects its front and back package.

Go-back-n protocol is more complex. Intuitively, transmission efficiency should be higher under the condition of larger window size, because larger window size would let more packages being sent simultaneously. Bigger the bandwidth, higher the efficiency. But to go-back-n protocol, after ascended some specific threshold, go-back-n protocol need to retransmit all of packages in its sending list when time out and mistake happens. Therefore, larger the window size would let it retransmit more packages. In this case, large window size is not something good, but the reason of decreasing of transmission efficiency.

Experiment

An overview of the experiment parameters is described below:

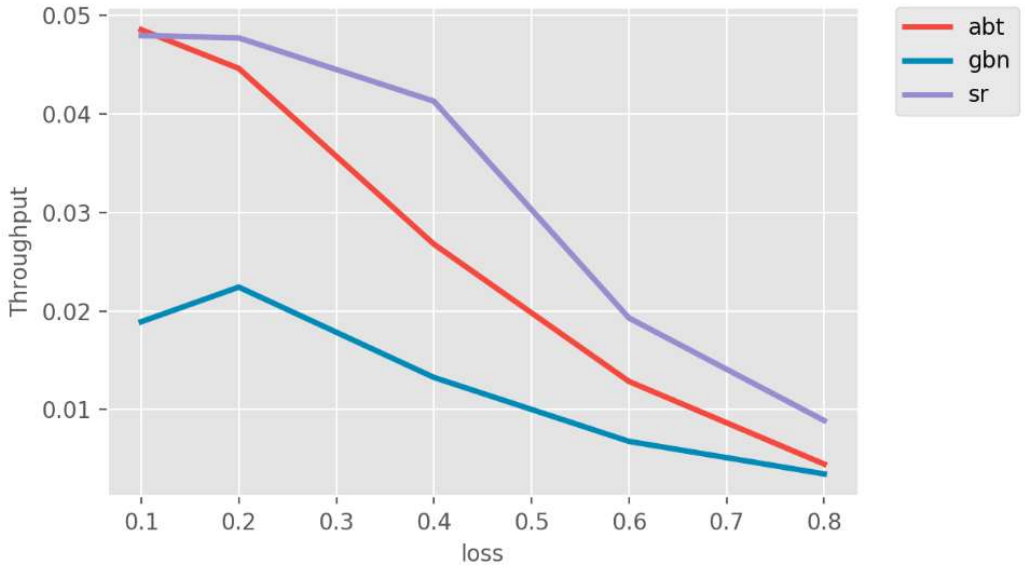
Experiment	Message	Corruption	Loss	Window	Total
Experiment1	100	0.1	0.1-0.8	10, 50	30
Experiment2	100	0.1	0.2-0.8	10-500	45

Incremental loss probability on window sizes (Experiment 1)

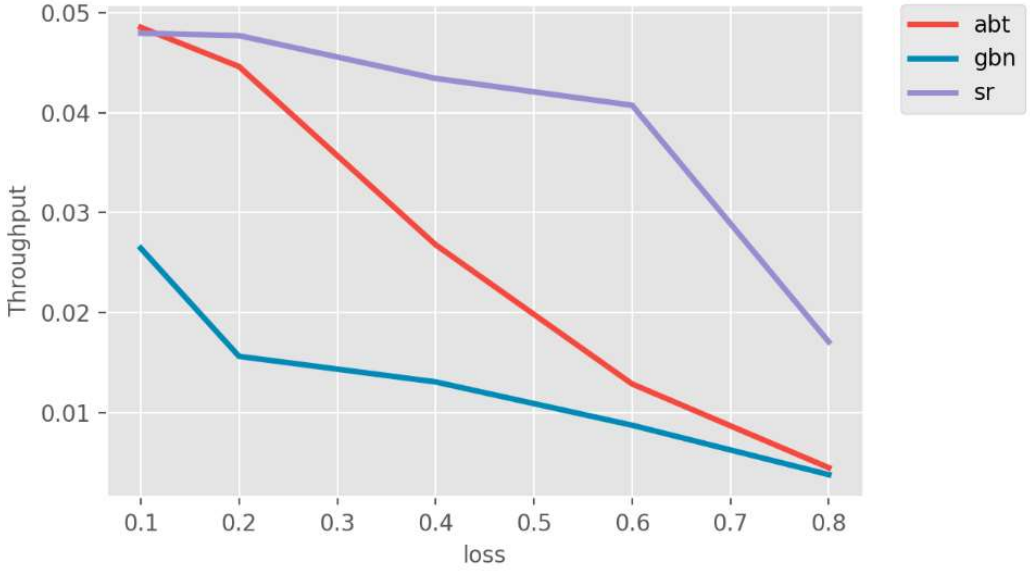
As suggested the test conducted with the following parameters

Protocol	Message	Corruption	Loss	Window
Alternative Bit protocol	100	0.1	0.1-0.8	10, 50
Go Back N	100	0.1	0.1-0.8	10, 50
Selective repeat	100	0.1	0.1-0.8	10, 50

In experiment 1, the loss level of transmission keep increasing, so intuitively the throughput of message should be decreasing. In this project, three protocol are used to transmit the message, which is alternative bit protocol, go-back-n and selective repeat. Each of them supports transmission differently. As loss probability increasing, fix window size as 10, the throughput of three protocols as follows.



As loss probability increasing, fix window size as 50, the throughput of three protocols as follows.



As graphs show above, selective repeat protocol does the best job of all, for it keep the highest throughput as loss probability increases. Selective protocol has buffer mechanism on the sender and receiver, so both of them can retransmit lost package without occupying much link resource. On the other hand, go-back-to-n protocol need to retransmit all of window size message after time out, it takes much of communication link to retransmit a lost package. Alternative bit protocol keeps stop and wait mechanism, whenever loss happens, it needs to stop and wait for retransmission. It is also a waste of communication link resource.

As graphs show above, alternative bit protocol shows better performance than go-back-n protocol. This is because go-back-n protocol need to retransmit window-size-number packages at a time when lost happens. This mechanism decreases the utilization of communication link as it retransmit lot of packages which have already been received by receiver side successfully, and

also increases the risk of losing package and ack corruption during the retransmission of packages. Because go-back-n protocol need to retransmit more packages than alternative bit protocol, the drawbacks of its mechanism becomes more obviously, so the performance of go-back-n protocol is not very well.

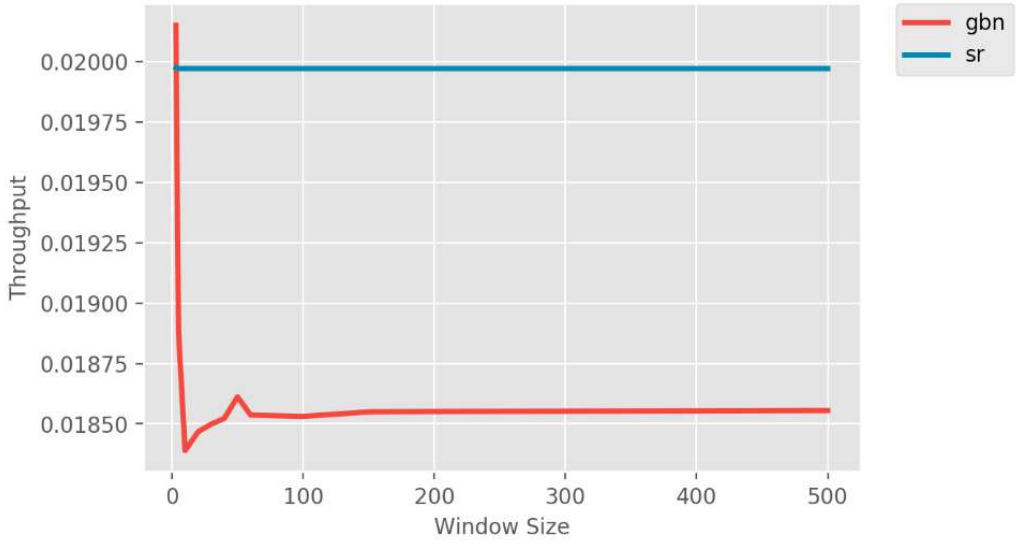
Generally, as loss probability increasing, throughput of all three protocols keep decreasing. Also, window size 50 shows better performance than window 10. This is result of sender can send message more frequently, which increases communication link utilization.

Incremental loss probability on window sizes (Experiment 2)

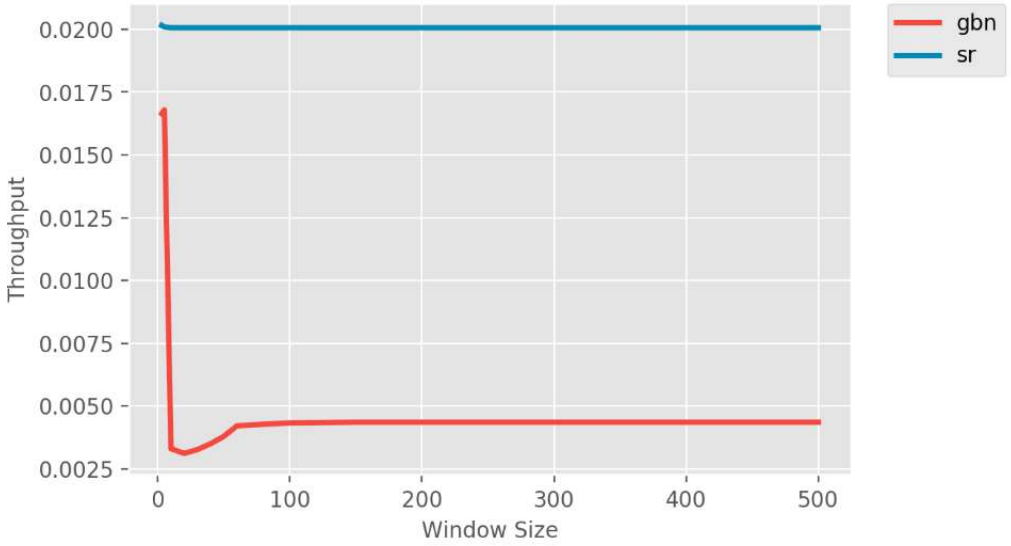
As suggested the test conducted with the following parameters

Protocol	Message	Corruption	Loss	Window
Alternative Bit protocol	100	0.1	0.2, 0.5, 0.8	10-500
Go Back N	100	0.1	0.2, 0.5, 0.8	10-500
Selective repeat	100	0.1	0.2, 0.5, 0.8	10-500

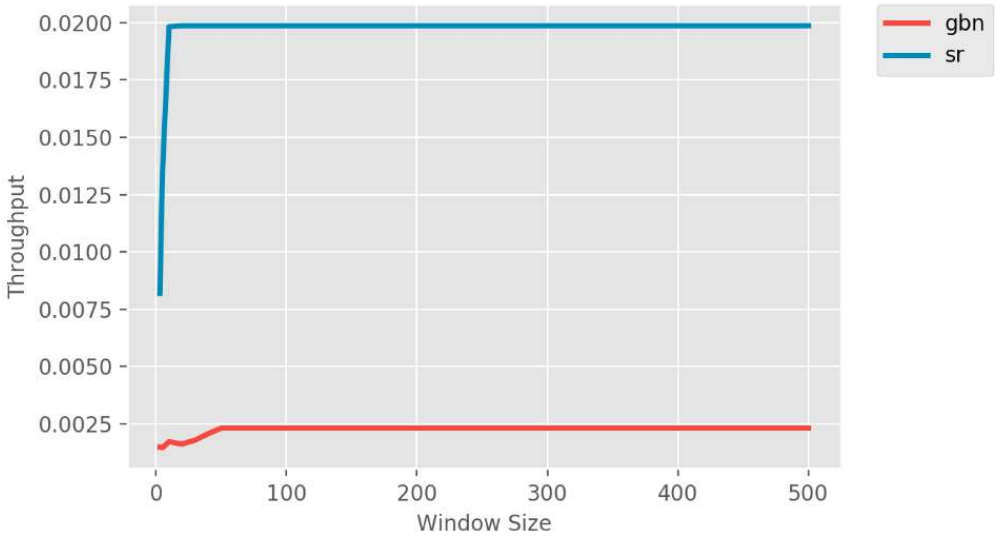
In experiment 2, loss probability is fixed as 0.2, 0.5 and 0.8. Window size is set from 10 to 500. As the increasing of window size, the frequently of transmission increases. Intuitively, alternative bit protocol should have no difference when window size changes, as it keeps stop and wait mechanism and have no window in it. The throughput of three protocols in different window size under loss probability of 0.2 is as follows.



The throughput of three protocols in different window size under loss probability of 0.5 is as follows.



The throughput of three protocols in different window size under loss probability of 0.8 is as follows.



As graphs show, selective repeat protocol and go-back-to-n protocol show generally steady performance as window size increasing. When loss probability is set as 0.8. When window size is set a small number, throughput of selective repeat protocol is relatively low, but as window size

increasing, throughput of selective repeat protocol become better and steady, this is because of the increasing of frequently of sending package, decreases the time out corruption frequency. As the increasing of window size, go-back-n protocol performance becomes better, then worse, this is explained above, because when package loss, go-back-n protocol need to window size number of packages, which has more risk to causing package loss again.

Analysis and discussion

In this project, go-back-n protocol and selective repeat protocol show their advantages and disadvantages. In go-back-n protocol, when window size is set to be a large number and the first package was sending with lost, it need to retransmit all of it window size number of packages, it increases the loss of time and delay of time. In selective protocol, sender side and receiver side may not always have the same result, which means windows of sender and receiver is not always the same. So when sequence number is limited, Unsynchronized windows between the sender and the receiver can have serious consequences. Such a consequence would result in the receiver not being able to determine whether the packet was a retransmission or a new packet, and thus the SR window could not be large.

But in other words, despite the weaknesses of GBN and SR mentioned above, if they exist, they must have the advantages of their two protocols. In GBN protocol, although all sequential groupings have to be discarded, this seems to be a weakness, in fact, it is also an advantage of GBN. The purpose of this is that the receiver must deliver the data to the upper layer in the correct order. If the packet is wrong, the receiver may cache the packet and deliver the data to the upper layer in the correct order. An obvious advantage of GBN is that it is easy to accept caching, that is, the receiver does not need to cache any disordered packets. Aimed at the weakness of GBN, it leads to the SR protocol. The advantage of SR protocol is that it solves the unnecessary retransmission problem in GBN and improves the utilization of channel in a sense. If a packet is wrong, it only needs to retransmit the packet, and after the receiver's cache has collected the correct order, it can be delivered to the upper layer. The requirement of realizing this function is that the receiver needs a certain cache capability.