EECE-7374

FUNDAMENTALS OF COMPUTER NETWORKS

Programming Assignment-2

Report

Date of Submission: 6th December 2024

Submitted to:

Prof. Dimitrios Koutsonikolas

Professor, Department of ECE,

Northeastern University, Boston MA 02115.

Submitted by:

Shreyas Kapanaiah Mahesh,

Sakthivel Ponnampalayam Sivakumar

NUID: 002332297; 002312294

Northeastern University, Boston MA 02115

ABT IMPLEMENTATION

The **alternating bit protocol** is the reliable protocol, wherein **packet sequence numbers** alternate between **0 and 1**. If a pkt 0 is send from A to B, then A waits a reasonable amount of time to receive an ACK for pkt 0, and if no ACK packet is received or the ACK is delayed or if it receives a wrong ACK, then **it retransmits the pkt 0**. If no loss, then the sender A sends the next packet with **seq_no 1** and this process repeats untill all the data packets are transmitted from A to B.

Variables initialized:

Message_buffer_size: Defines the maximum size of the message buffer, and allows only a limited number of messages that can be stored (1000) for transmission.

Timeout: This specifies the duration after which a timeout event is called, if an ACK is not received.

Sequence_number: This function keeps track of the sequence number of the packet being sent by A. In ABT it alternates between 0 and 1

Expected_seqnum: Expected sequence number expected by Host B

Expecting_ACK: A boolean value to check whether host A is waiting for an ACK from Host B after sending a packet.

Last_pkt: It stores the last sent packet by Host A, needed if there is a possibility of retransmission.

Message_buffer: It is a data structure that follows First in First Out principle, that stores data.

TIMEOUT: we chose 10, because it gave a better throughput on all the experiments with a good runtime.

GBN IMPLEMENTATION:

The Go-Back-N (GBN) protocol is a reliable data transfer protocol allowing the sender to transmit multiple packets without waiting for individual acknowledgements, up to a finite limit N. The sender tracks the sequence number of the oldest unacknowledged packet (base) and the next unused sequence number. Packets in the range are sent but unacknowledged, while those in are ready to be sent. Packets beyond must wait until an earlier packet is acknowledged. The protocol uses a sliding window of size NN to manage the permissible range of unacknowledged packets dynamically.

Variables initialized: Along with some similar variables initialized in the previous ABT protocol, the GBN protocol also has ...

Base: Represents the sequence number of the first unACKed packet in the window.

Next_seqnum: The sequence number of the next packet to be sent. (base + getwinsize())

Window: It is a vector – a dynamic array thta that is used for storing packets in the sliding window of the GBN protocol.

TIMEOUT: we chose 20, because it gave a better throughput for experiments with loss rate greater than 0.4 and had a better runtime for window sizes greater than 50, when compared with other timeout values in the range of 10 – 30. (increments of 5)

SELECTIVE REPEAT IMPLEMENTATION

The Selective Repeat protocol avoids the unnecessary retransmissions caused on the previous Go-Back N protocol by having the sender retransmit only the packets, that are lost. And doen't retransmit the whole window. This individual retransmission of packets will require that the receiver individually acknowledge correctly received packets. Similarly like GBN, a window size of N will again be used to limit the number of unACKed packets. But the sender will have received the ACKs already for some packets in the window.

The receiver side of the SR protocol, will ACK a correctly received packet, the out-of-order packets are buffered until any missing packet (with lower sequence numbers) are received.

Variables initialized: Along with some similar variables initialized in the previous ABT & GBN protocol, the SR protocol also has ...

B_window: This represents the Host B side's sliding window, to store the unordered packets arrived from Host A

Window (map): It basically tracks the packets in the Host A's sender's window. A map stores key-value pairs in a program. Here in general, the map is used to store timestamps, ACKs, packets, etc.

Timestamp: A map of float values, it records the time of each packet in Host A's window, and calculates remaining time for any potential retransmission.

Ack_received: A map of booleans, it indicates whether an ACK has been received for a specific sequence number.

B_buffer: It stores or buffers out-of-order packets received by Host B.

B_received: It tracks which sequence numbers have been received and buffered.

TIMEOUT: Since GBN gave a good result for a timeout value of 20, we implemented the same for SR.

Timer Implementation: A single timer and the simulation time is integrated together to implement multiple timers virtually. During the transmission of all the packets in a window, the time of transmission is noted in a buffer. When the first packet of the window is transmitted, the timer would run for TIMEOUT duration to wait for the acknowledgement from the time of transmitting the packet. When the ack signal arrives, the timer is discarded and the next one should start.

The timer for the next packet's ack signal would not be the entire TIMEOUT duration. Instead, it will only be for the remaining time from the time of transmitting the packet. It is calculated using this formula

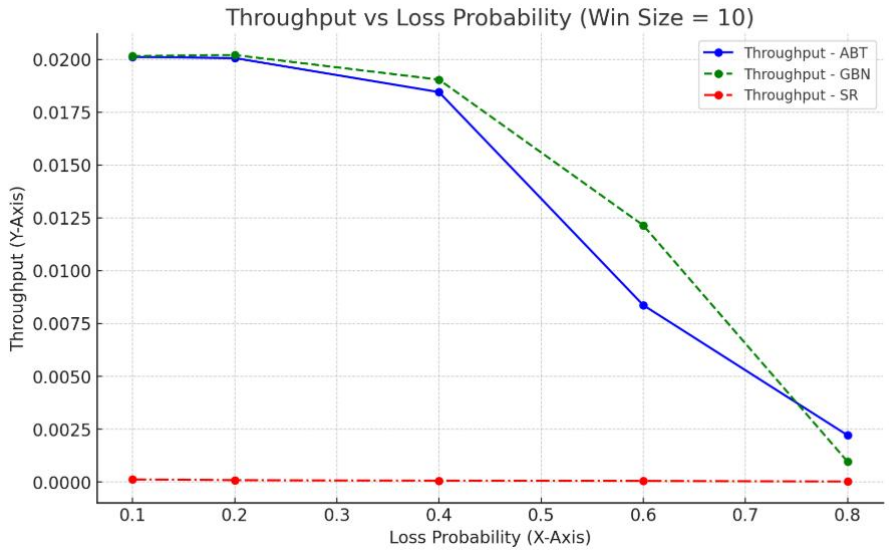
Remaining Duration for ACK = TIMEOUT - (Current Simulation Time - Packet's Transmitted Time)

PERFORMANCE COMPARISONS:

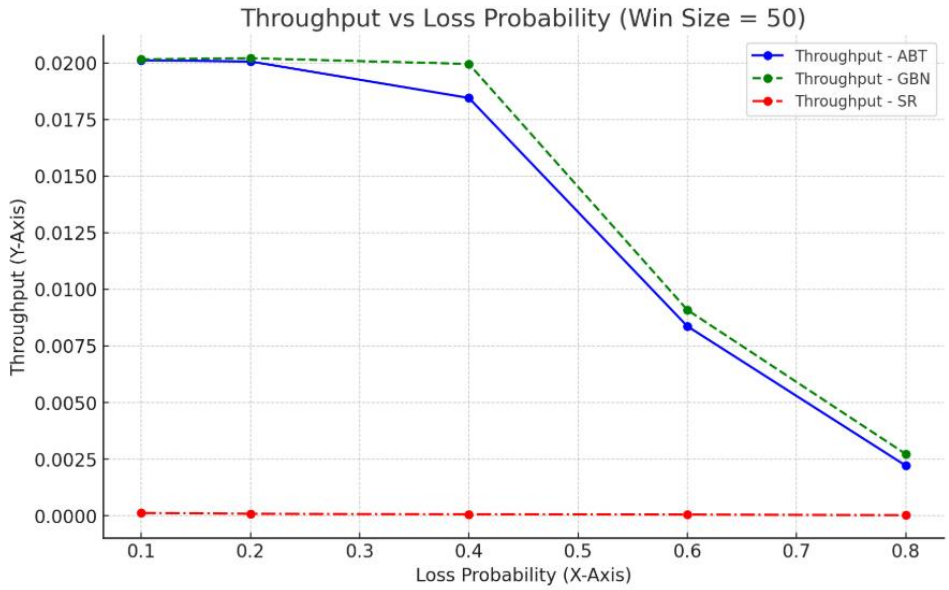
Here the performance of ABT, GBN & SR. In the following two experiments, both the implemented protocols are run with a total of 1000 messages, mean time - 50 and a corruption probability of 0.2. All are common throughout.

EXPERIMENT-1: Testing loss probabilities {0.1,0.2,0.4,0.6,0.8} vs obtained throughput, with respect to the given window sizes {10,50}

Results of EXPERIMENT-1 with window size 10:



Results of EXPERIMENT-1 with window size 50:

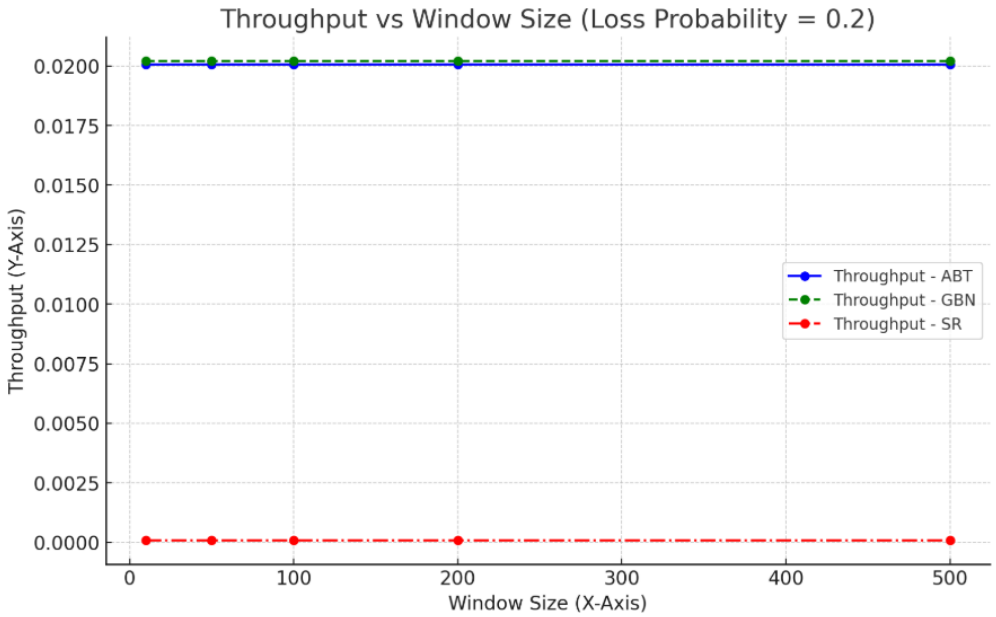


From the **loss probability vs throughput** graphs with each of the respective window sizes fixed, We can observe that GBN has a better throughput compared to ABT and SR for both window sizes. ABT has the second best throughput and SR has a very low throughput. As the Loss probability increases, the throughput values also decrease in both the cases.

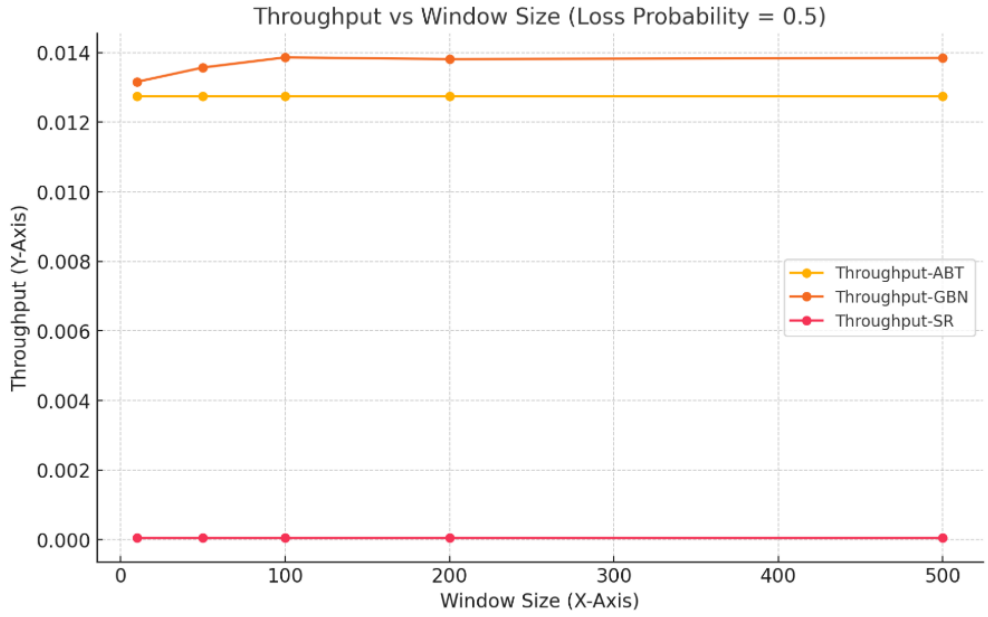
In ABT, as there is no window size required, therefore the throughput plots of ABT with any window size remains the same. In GBN & SR protocol, since it has window sizes and 2 different window sizes are used, we observe a slight difference in the plots of GBN & SR.

EXPERIMENT-2: Testing window sizes {10,50,100,200,500} vs obtained throughput, with respect to the given loss probabilities {0.2, 0.5, 0.8}

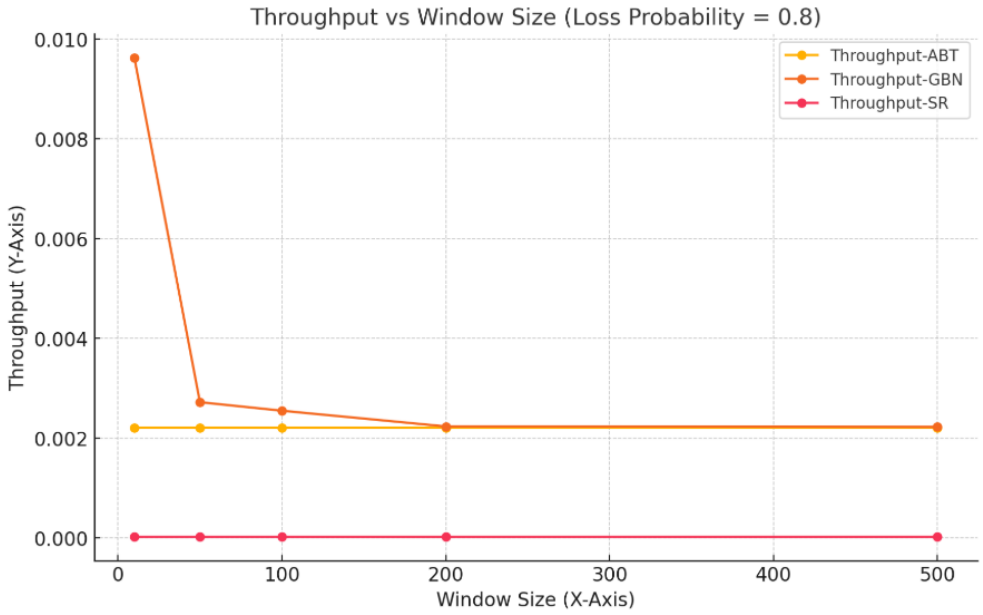
Results of EXPERIMENT-2 with loss probability 0.2



Results of EXPERIMENT-2 with loss probability 0.5



Results of EXPERIMENT-2 with loss probability 0.8



For each loss probabilities {0.2,0.5,0.8} and with varying window sizes {10,50,100,200,500}, the throughput of the ABT, GBN and SR protocols remain very low and remain constant throughout for loss probability of 0.2. With a loss probability of 0.5, there is a slight increase in throughput of GBN, and becomes constant throughout, as window size increases, while ABT & SR remain constant with lower values.

With a loss probability of 0.8, a sudden decrease of throughput is observed when window size is increased from 10 to 50. Then a gradual decrease from 50 to 200 and afterwards, it remains constant at a throughout of 0.002. While that of ABT and SR remain constant from the beginning with such lower values.

FINAL OBSERVATIONS:

Observations made in Experiment-1 of the 3 protocols implementation is that the value of throughput remains very low. But amongst the 3, the GBN protocol shows better behaviour. As the loss-probabilities are increased, the throughput of ABT and GBN decrease down. While that of SR it will be the lowest, constantly running.

Observations made in Experiment-2 of the implementation of the 3 protocols, All the 3 protocols exhibit a very low throughput. Among the 3, the GBN protocol exhibits better throughput than ABT and SR with exhibit very low near-constant levels of throughput as the window size increases.