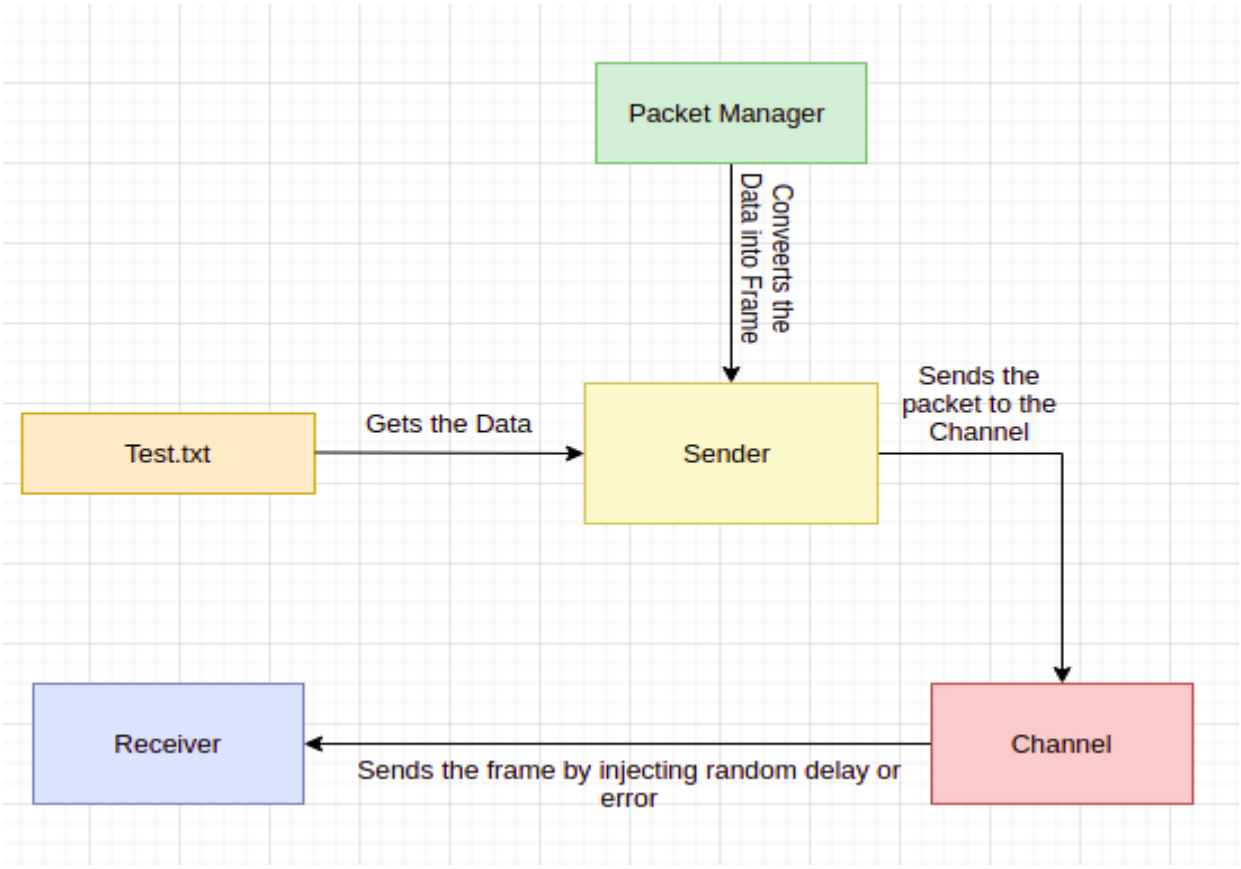
Network Lab Report **Assignment 2**

Name :- Debargha Mukherjee BCSE UG-3 Group A3 Roll: 001910501067

Problem Statement: Implement three data link layer protocols, Stop and Wait, Go Back N Sliding Window and Selective Repeat Sliding Window for flow control.

Network Architecture Design



Network Architecture (fig 1)

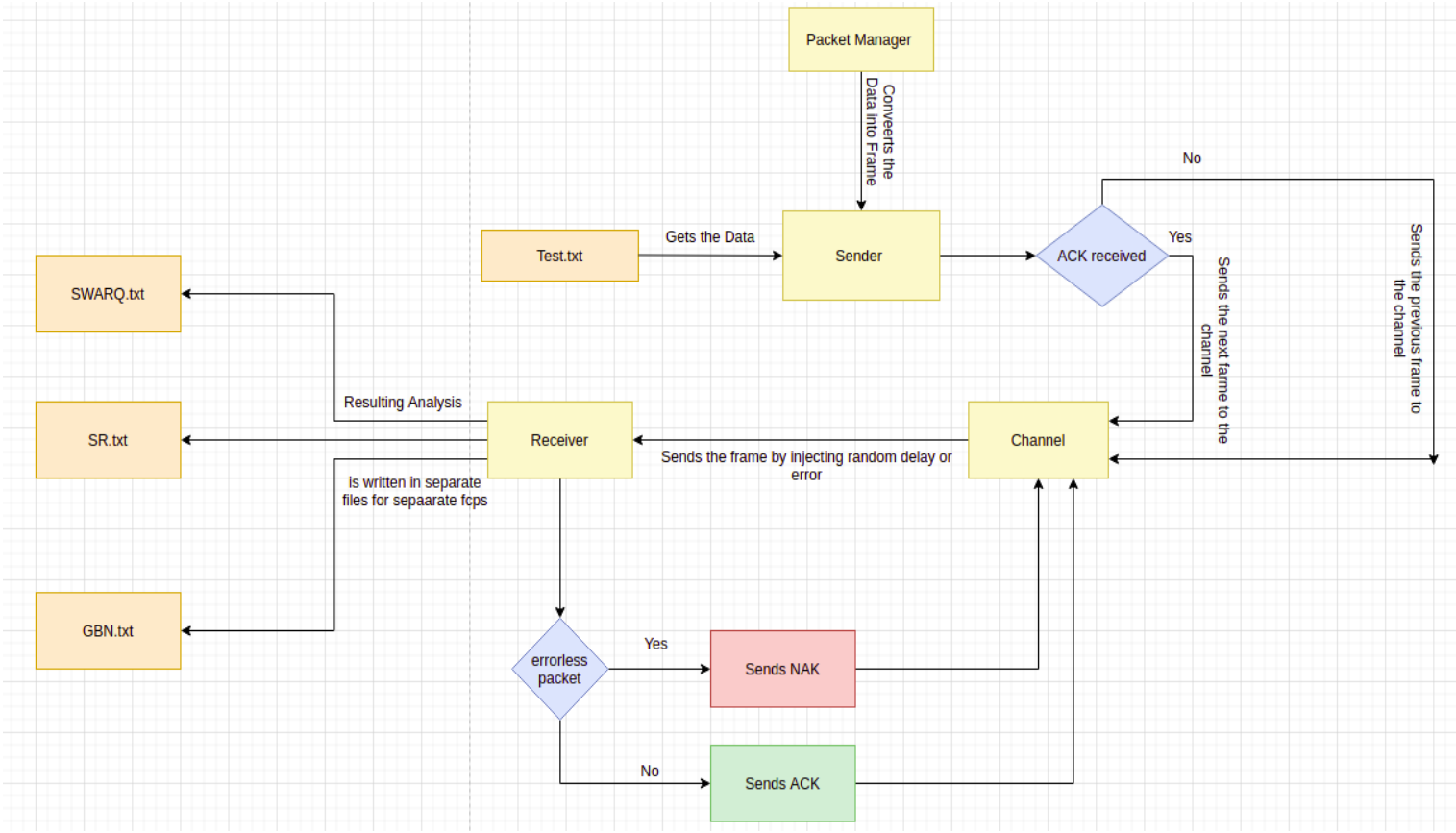
Different Roles in the architecture

1. **Sender:** Reads the data from file test.txt and Converts the data into packet following the IEEE 802.3 Frame Format (using the module **packet manager**). It starts three threads listening to the same port 1232. First one is responsible for sending the packet to the channel. Second to receive any acknowledgement (ACK or NAK). Third, send the previous packets, if ACK (positive

acknowledgement) is not received. The three processes are implemented using a lock class to prevent any type racing condition while listening to the same port.

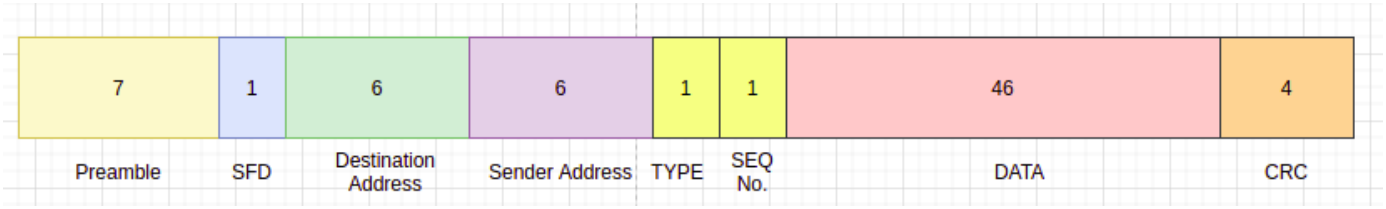
2. **Channel:** Receives the data packets from the sender (s). Processes the packet and introduces random delay or error into the packets. It is designed in a way that it will send the packet with 65 % chance, introduce delay with 10 % chance, injects error into the packet with 15 % chance and drops the packet with a 10 % chance. Finally it will receive acknowledgement from the receiver and pass them to the sender.

3. **Receiver:** Receives data packets from the sender via the channel. Checks if the data is erroneous or not. If the data is error free it will send a positive acknowledgement to the channel. In sliding window protocols, if the packet ranges aren't valid at any moment, send a NAK. If the sequence number is still stuck at the last packet, resend the acknowledgement for the previous packet simultaneously.



Flow diagram representing the complete end to end service (Fig 2)

Data Frame Format (Packet Format)



(Fig 3)

The Data Packet used is the IEEE 802.3 Ethernet Frame Format for designing the packets. Total Size of the frame is 72 bytes.

Details Regarding the IEEE 802.3 Ethernet Frame Format:

PREAMBLE – Ethernet frame starts with 7-Bytes Preamble. This is a pattern of alternative 0's and 1's which indicates starting of the frame and allows sender and receiver to establish bit synchronization. Initially, PRE (Preamble) was introduced to allow for the loss of a few bits due to signal delays. But today's high-speed Ethernet doesn't need Preamble to protect the frame bits. PRE (Preamble) indicates to the receiver that the frame is coming and allows the receiver to lock onto the data stream before the actual frame begins.

Start of frame delimiter (SFD) – This is a 1-Byte field that is always set to 10101011. SFD indicates that upcoming bits are starting off the frame, which is the destination address. Sometimes SFD is considered the part of PRE, this is the reason Preamble is described as 8 Bytes in many places. The SFD warns stations or stations that this is the last chance for synchronization.

Destination Address – This is a 6-Byte field that contains the port address of the destination socket.

Source Address – This is a 6-Byte field that contains the port address of the sender socket.

Type - This is a 1-Byte field that contains the type of the packet, whether it is a data packet or an acknowledgement.

Sequence No. - This is a 1-Byte field that contains the sequence no. of the current frame.

Data – This is the place where actual data is inserted, also known as Payload. Both IP header and data will be inserted here if Internet Protocol is used over Ethernet. The maximum data present may be as long as 1500 Bytes. In case data length is less than minimum length i.e. 46 bytes, then padding 0's is added to meet the minimum possible length.

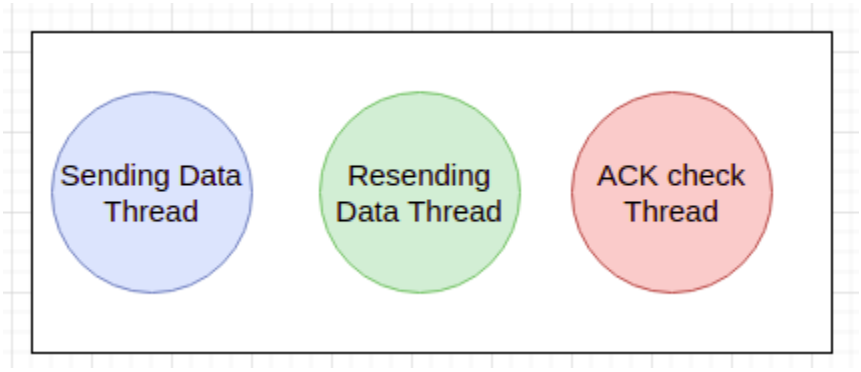
Cyclic Redundancy Check (CRC) – CRC is a 4 Byte field. This field contains a 32-bits hash code of data, which is generated over the Destination Address, Source Address, Length, and Data field. If

the checksum computed by destination is not the same as the sent checksum value, the data received is corrupted.

Implementation and Flow Control Protocols:

1. Stop and Wait ARQ (For noisy channel)

Sender side Stop and Wait: Sender sends one data packet at a time.Sender sends the next packet only when it receives the acknowledgment of the previous packet.Therefore, the idea of stop and wait protocol in the sender's side is very simple, i.e., send one packet at a time, and do not send another packet before receiving the acknowledgment.



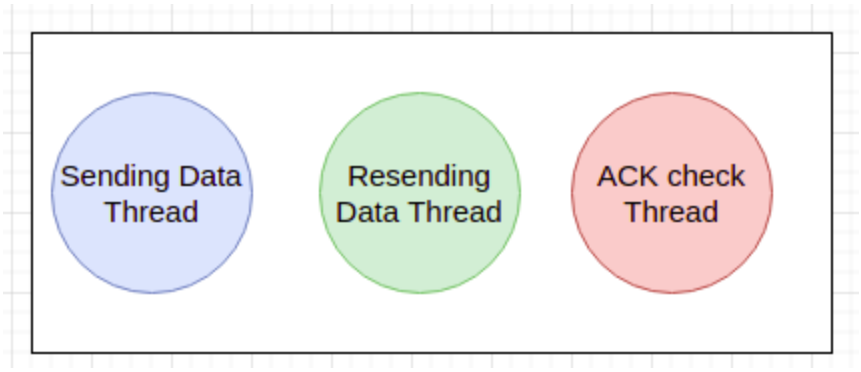
(Fig 4)

Stop and Wait implemented using multithreading and lock class to prevent race conditions.

Receiver Side Stop and Wait: Receive and then consume the data packet.When the data packet is consumed, receiver sends the acknowledgment to the sender.Therefore, the idea of stop and wait protocol in the receiver's side is also very simple, i.e., consume the packet, and once the packet is consumed, the acknowledgment is sent. This is known as a flow control mechanism.

2. Go Back N ARQ

Sender side GBN: It is a sliding window protocol responsible for sending packets in windows of some fixed size and waiting for ACK for the entire window. If some frame is lost in the window before reaching the receiver, it will send the entire frame again.



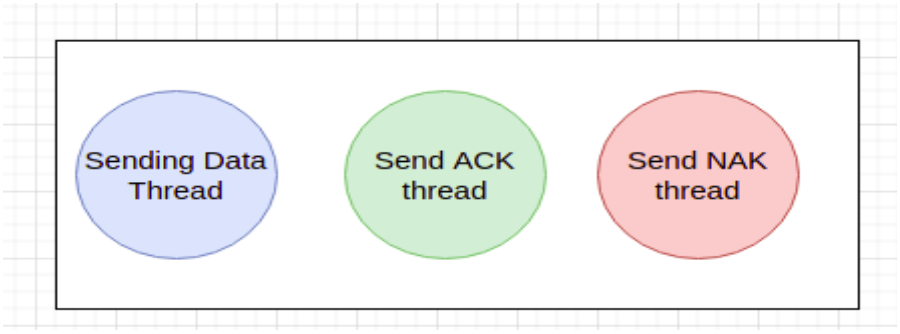
(Fig 5)
Go Back N implemented using multithreading and lock class to prevent race conditions.

Receiver side GBN: The receiver side is responsible for sending an ACK when it receives the entire window of frames and sending a NAK when a packet is lost due to time out or an erroneous packet is sent.

3. Selective Repeat ARQ

(Window size should be less than or equal to the half of the sequence number in SR protocol. This is to avoid packets being recognized incorrectly. If the windows size is greater than half of the sequence number space, then the ACK is lost, the sender may send new packets that the receiver believes are retransmissions.)

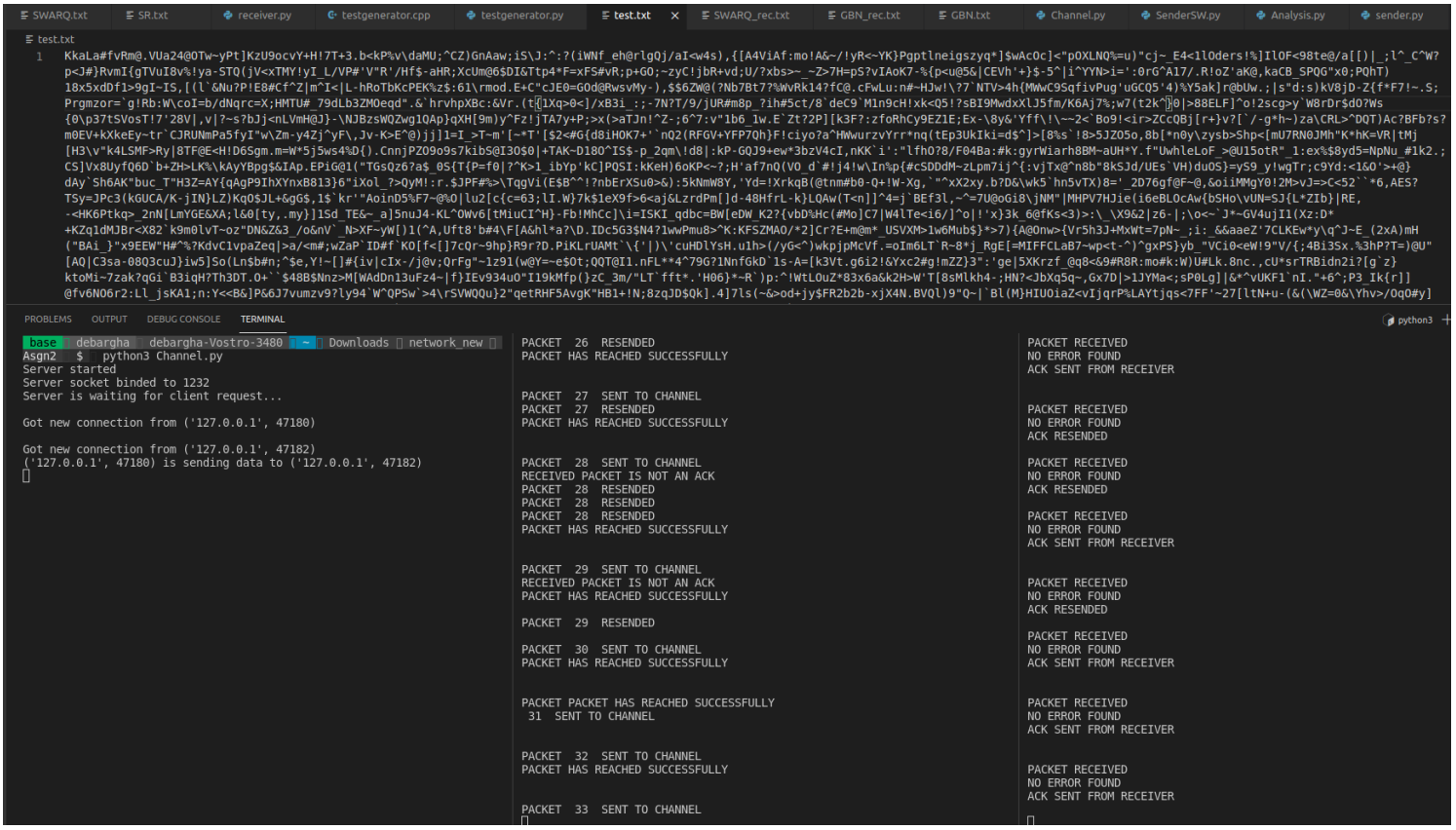
Sender Side SR: Sender can transmit new packets as long as their number is with W of all unACKed packets. It retransmits un-ACKed packets after a timeout – Or upon a NAK if NAK is employed.



(fig 6)
Sender side Selective repeat implemented using multithreading and lock class to prevent race condition

Receiver Side SR: Receiver ACKs all correct packets. Receiver stores correct packets until they can be delivered in order to the higher layer.

Output Snapshot



Snapshot of a test case running on terminal (Test case generated using testgenerator.py, random ASCII string of length 32768).

Results and Analysis

We have used throughput, efficiency, and average successful transmission time of a packet as the parameters to compare the three flow control protocols.

Essential Relations required to compare the different flow control protocols.

$$\text{Throughput} = \frac{\text{Effective number of bits sent (no. of packets * packet size in bits)}}{\text{Total Time taken}}$$

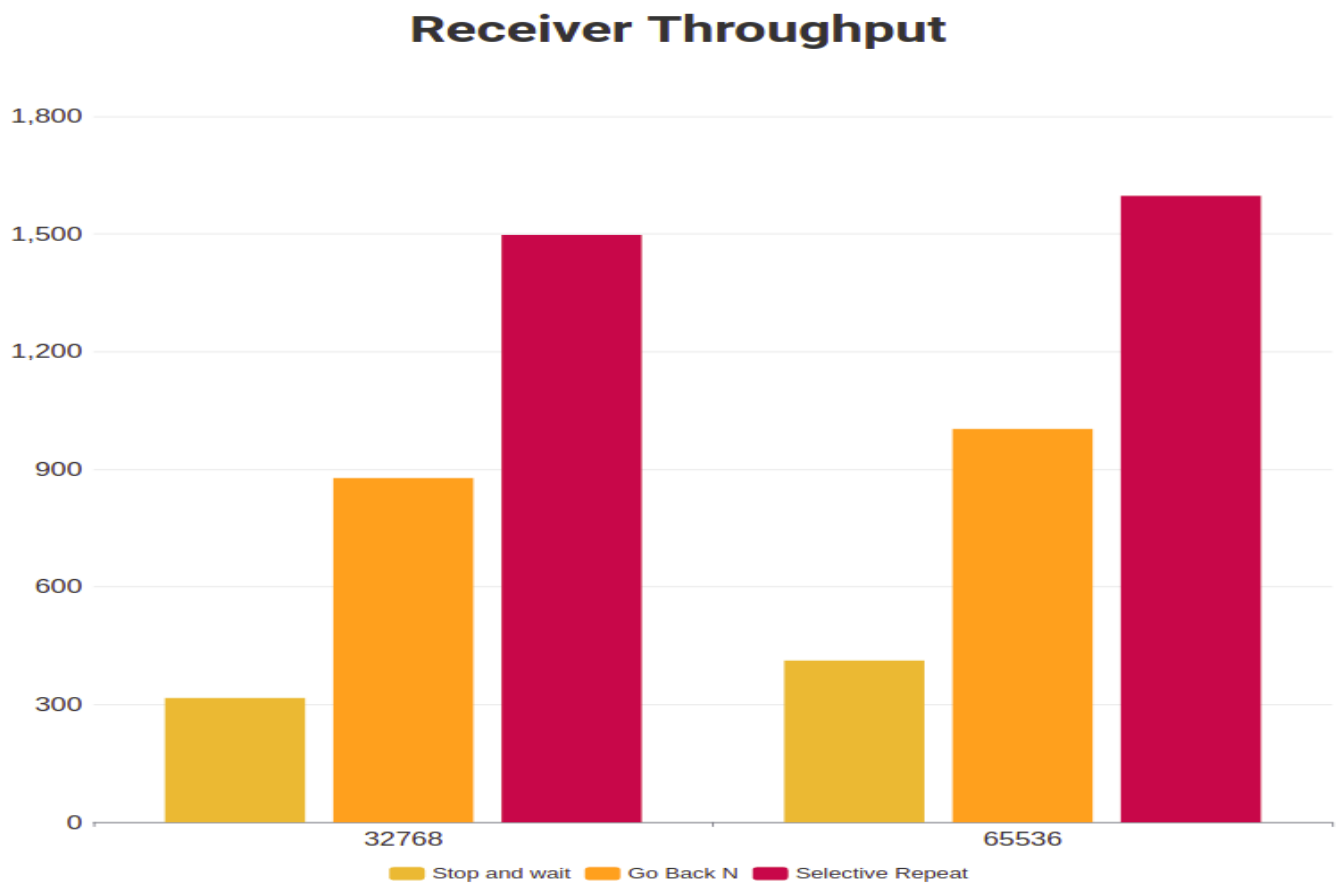
$$\text{Efficiency} = \frac{\text{Throughput}}{\text{Bandwidth}}$$

$$\text{Avg successful Transmission time of a packet} = \frac{\text{Total Time Taken}}{\text{Effective number of packets}}$$

Receiver Throughput

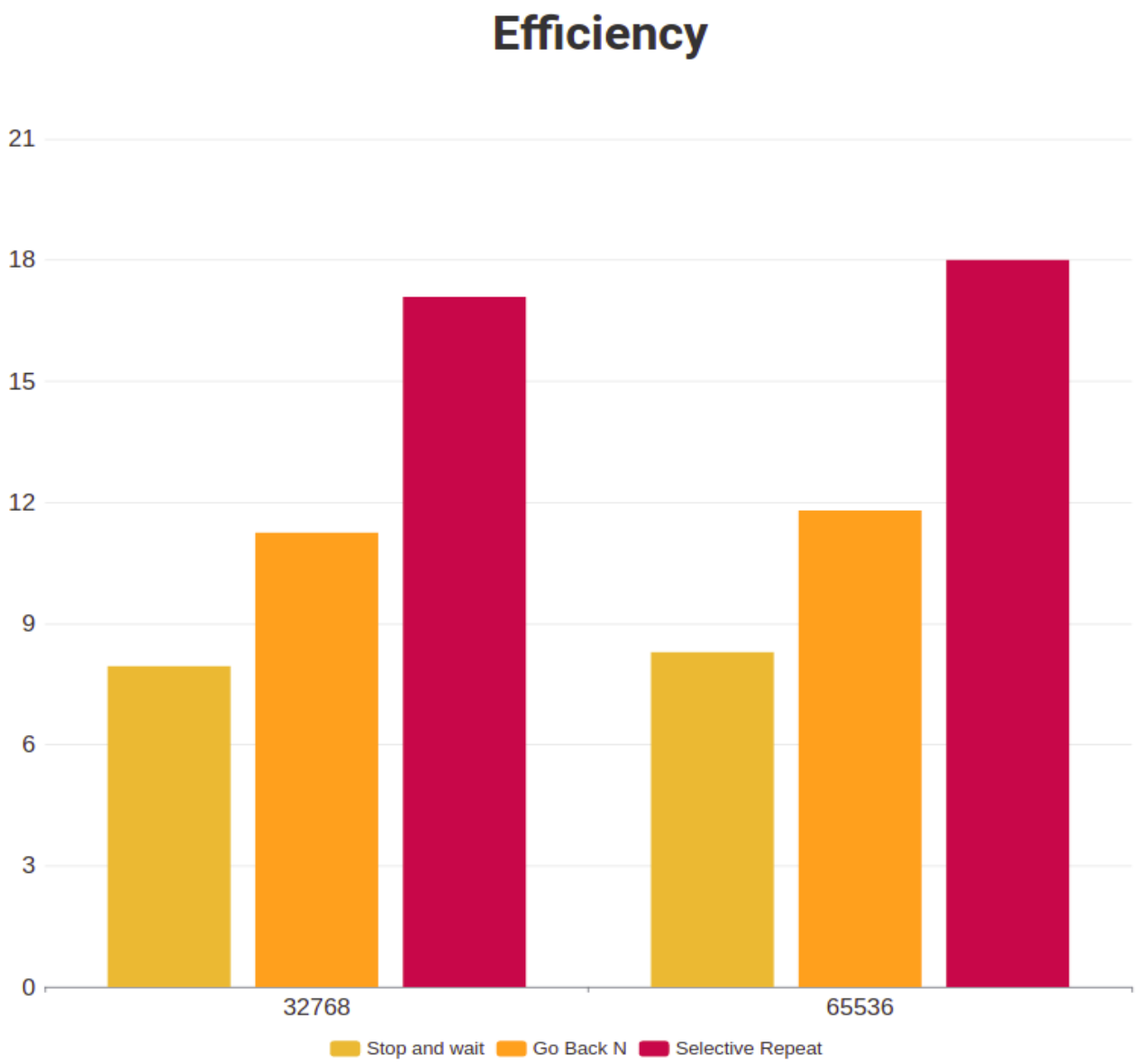
protocols / Data size	Stop and Wait	Go Back N	Selective Repeat
32768 bytes	317 bps	878 bps	1498 bps
65536 bytes	412 bps	1003 bps	1598 bps
average	364.5	940.5	1548 bps

Column Chart Representing the same :



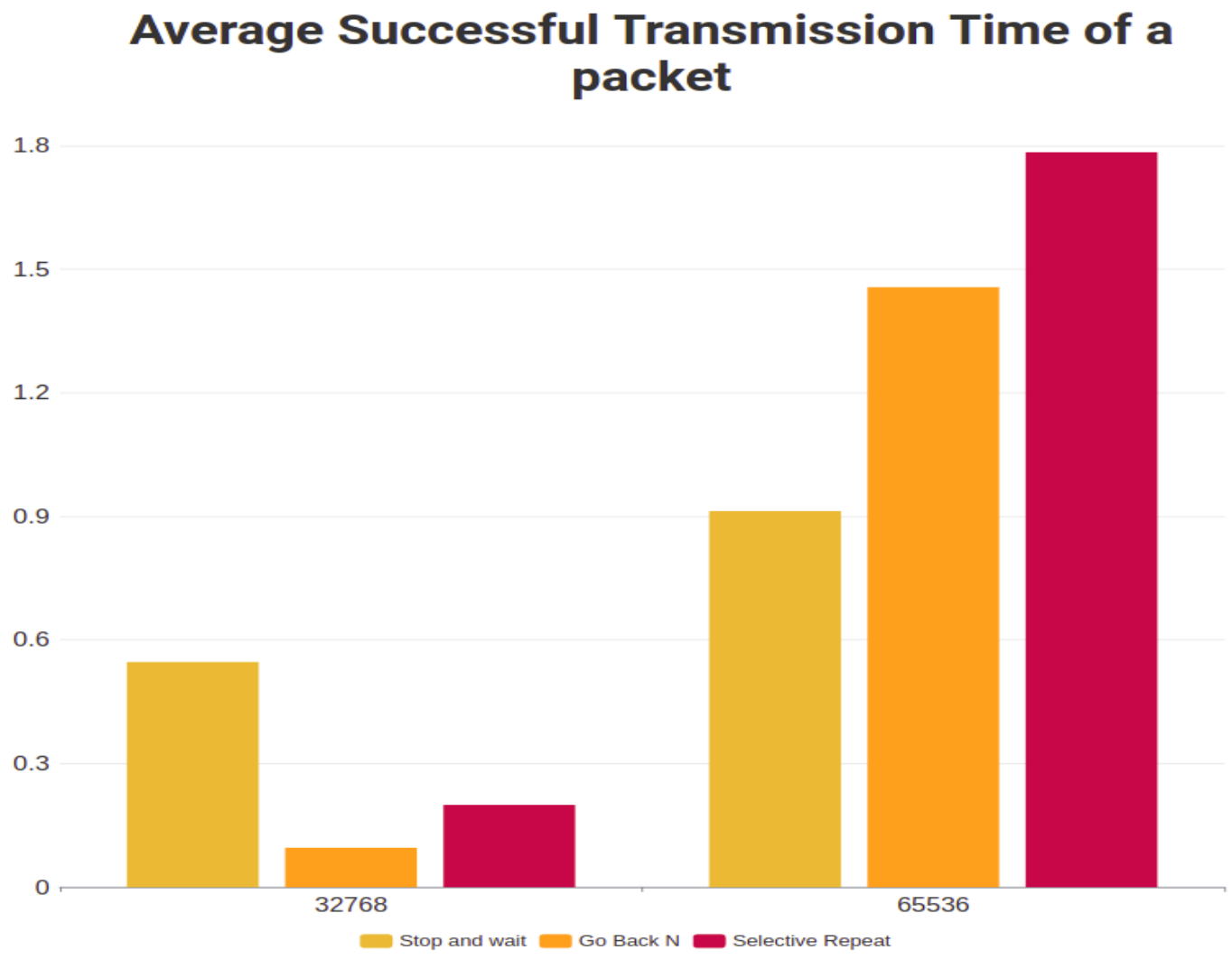
Efficiency

protocols / Data size	Stop and Wait	Go Back N	Selective Repeat
32768 bytes	7.94 %	11.25 %	17.09%
65536 bytes	8.64 %	12.34 %	18.92%
average	8.29 %	11.795 %	18.005 %



Average Successful Transmission Time of a packet

protocols / Data size	Stop and Wait (seconds)	Go Back N (seconds)	Selective Repeat (seconds)
32768 bytes	0.54667	0.09505	0.199779
65536 bytes	1.27865	1.45634	1.784322
average	0.91266	0.82322	0.9920505



Analysis

From the tables above we clearly rule out the differences between the performance measured in terms of throughput efficiency and average transmission time of the different flow control protocols. Selective Repeat is better than Go Back N and stop and wait. The sliding window protocols are more efficient than the primitive protocol Stop and Wait.

Round Trip Time, which is the total time taken by a packet to move to and for should not be a measuring parameter because it's a property of the network and not the protocol. Although It can be checked by evaluating the RTT for a noiseless channel and then comparing with the other protocols. Also while measuring the RTT the processing and transmission time has been ignored which can play a significant role.

Similarly the **Bandwidth Delay Product** because it is too a property of the channel and not the protocol.

Instead of multiprocessing ,multithreading has been incorporated due to the difficulties faced while establishing a communication between different ports. All the threads running are made to communicate through a single port 1232 here.

Limitations

1. Exceptions related to multiple receivers may arise which are not handled in this code.

Comments

Overall assignment revolves around analysing the protocols associated with the data link layers and comparing their performance. It will help us to choose the correct protocol while data transfer. Apart from this if the time related to system calls and other system induced delays can be eradicated the measurement would have been more accurate.