Computer Science And Engineering
University of Dhaka
CSE-3112

Experiment name: Comparative Performance Analysis of Go-Back-N and Selective Repeat ARQ Mechanisms

Submitted by: Md. Faruk Hossain
Roll: 84
Md. Mizanur Rahman
Roll: 60

Submitted to: Dr. Rezaul Karim
Ms.Nusrat Mehajabin
Mrs.Iffat Anjum

Index

1. Introduction:

In data communications, flow control is the process of managing the rate of data transmission between two nodes to prevent a fast sender from overwhelming a slow receiver. It provides a mechanism for the receiver to control the transmission speed, so that the receiving node is not overwhelmed with data from transmitting node.
Flow control is important because it is possible for a sending computer to transmit information at a faster rate than the destination computer can receive and process it. This can happen if the receiving computers have a heavy traffic load in comparison to the sending computer, or if the receiving computer has less processing power than the sending computer.
A method of flow control in which a receiver gives a transmitter permission to transmit data until a window is full. When the window is full, the transmitter must stop transmitting until the receiver advertises a larger window.
Sliding-window flow control is best utilized when the buffer size is limited and pre-established.

There are two common Sliding-window flow control mechanism :

Go-Back-N ARQ
Selective Repeat ARQ

2. Problem definition:

a. Define the problem as what was expected to accomplish:

This experiment we will have to implement both Go-Back-N and Selective Repeat ARQ mechanisms. Each of our implementations should have the

following features.

1. Window size n. [take from user]

2. Sender should read data from a file (size >5MB) into 8 byte segments.

3. Data link layer Frames has a defined structure with header, trailer and other flag information. Data (8 byte) within such a structure need to be enclosed. We need to separate this data . [we should compute CRC-16 on data]

4. Acknowledgements, negative acknowledgements should also follow similar frame structure. In that case data field will contain acknowledged sequence number.

5. Separate thread for each data frame sent.

6. Timeout should be implemented for each data frame. Therefore each frame should have its own thread which will die when the frames acknowledged.

7. The receiver can send:

a) Both single and cumulative acknowledgements.
b) Negative acknowledgement is sent when an erroneous packet is received.
c) There should be timer for both positive and negative acknowledgements.

8. For Go-Back-N ARQ and Selective Repeat ARQ: Window will slide and adjust on each in-order successful transmission.

9. Our program should consider situations like packet loss, error in packet, acknowledgement loss, error in acknowledgement, negative acknowledgement loss and error in negative acknowledgement [all of these could be implemented probabilistically]

10. Upon receiving a packet you should perform CRC check and reconstruct the source file.

b.The inputs and outputs:

In this experiment we have to send data from a text file ,these data are the input for the sender and the ACK are the output.In receiver side the data with sequence number and CRC code are the output and the ACK are the input.

c. Equipment required:

1.IDE netbeans(java compiler)
2.Sender class(A thread class to send requested data)
3.Receiver class.
4.Theoretical background.

2.Theoretical background

a.Definition of Go-Back-N ARQ:

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In Go-Back-N Automatic Repeat Request, we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

Frames from a sending station are numbered sequentially. If the header of the frame allows m bits for the sequence number, the sequence numbers range from 0 to 2^m- 1. For example, if m is 4, the only sequence numbers are 0 through 15 inclusive. However, we can repeat the sequence. So the sequence numbers are

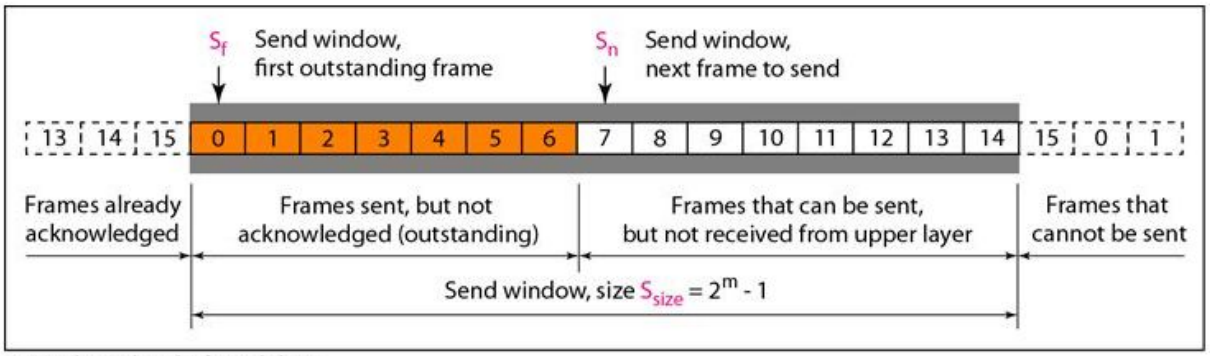
0, 1,2,3,4,5,6, 7,8,9, 10, 11, 12, 13, 14, 15,0, 1,2,3,4,5,6,7,8,9,10, 11, ...

Operational principles:

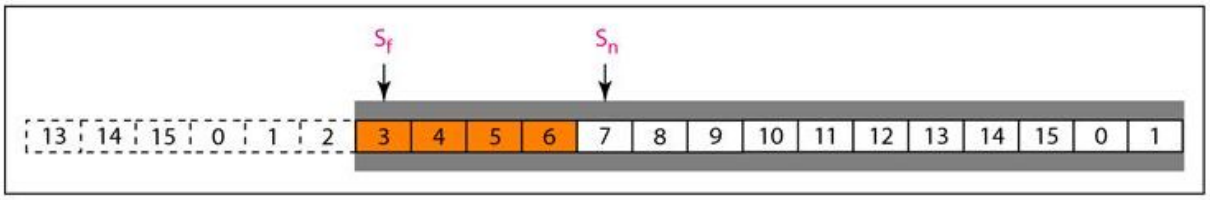
Sliding Window:

In this protocol (and the next), the sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers. The range which is the concern of the sender is called the sending sliding window; the range that is the concern of the receiver is called the receiver sliding window. The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit. In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent. The maximum size of the window is 2^m – 1. The size can be fixed and set to the maximum value. The following figure

shows a sliding window of size 15 (m =4).



a. Send window before sliding



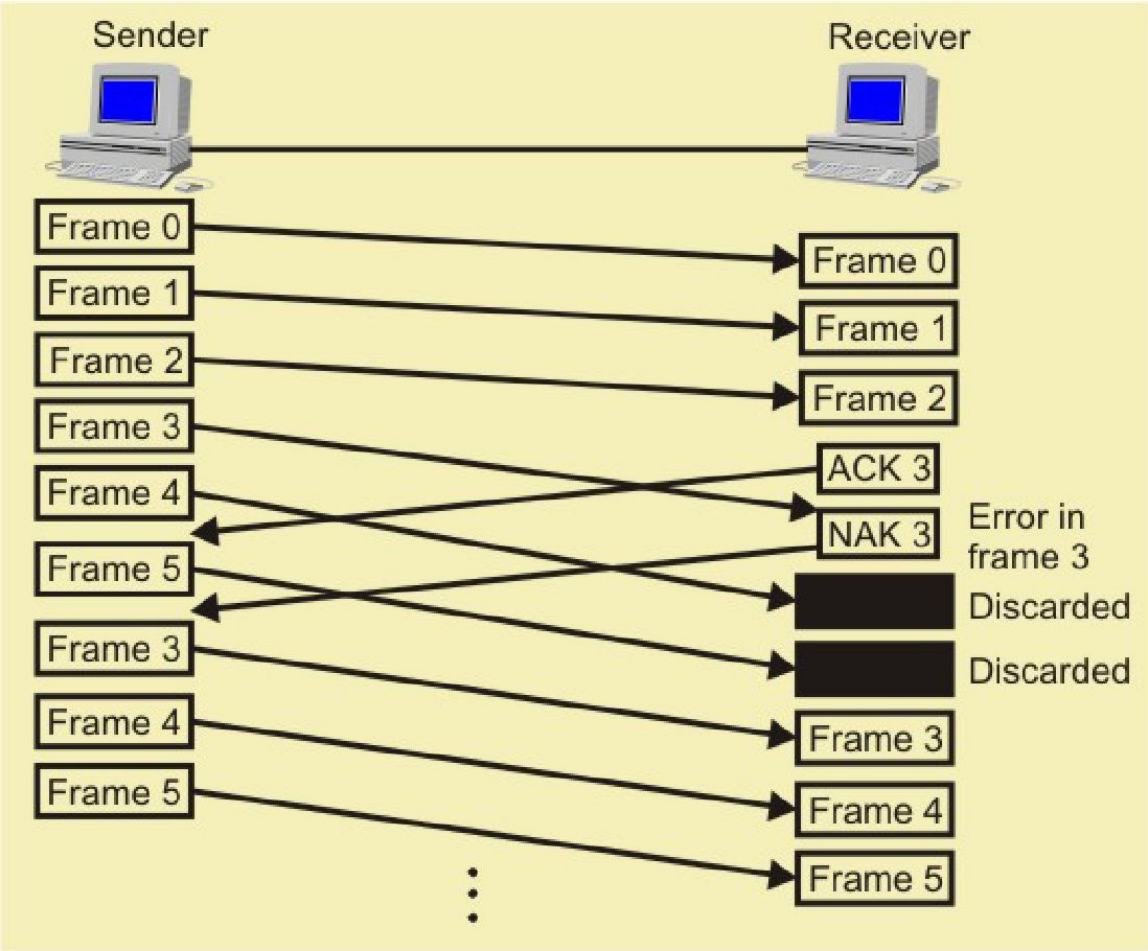
b. Send window after sliding

The window at any time divides the possible sequence numbers into four regions. The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them. The second region, colored in the above figure- a, defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames. The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer. Finally, the fourth region defines sequence numbers that cannot be used until the window slides, as we see next.

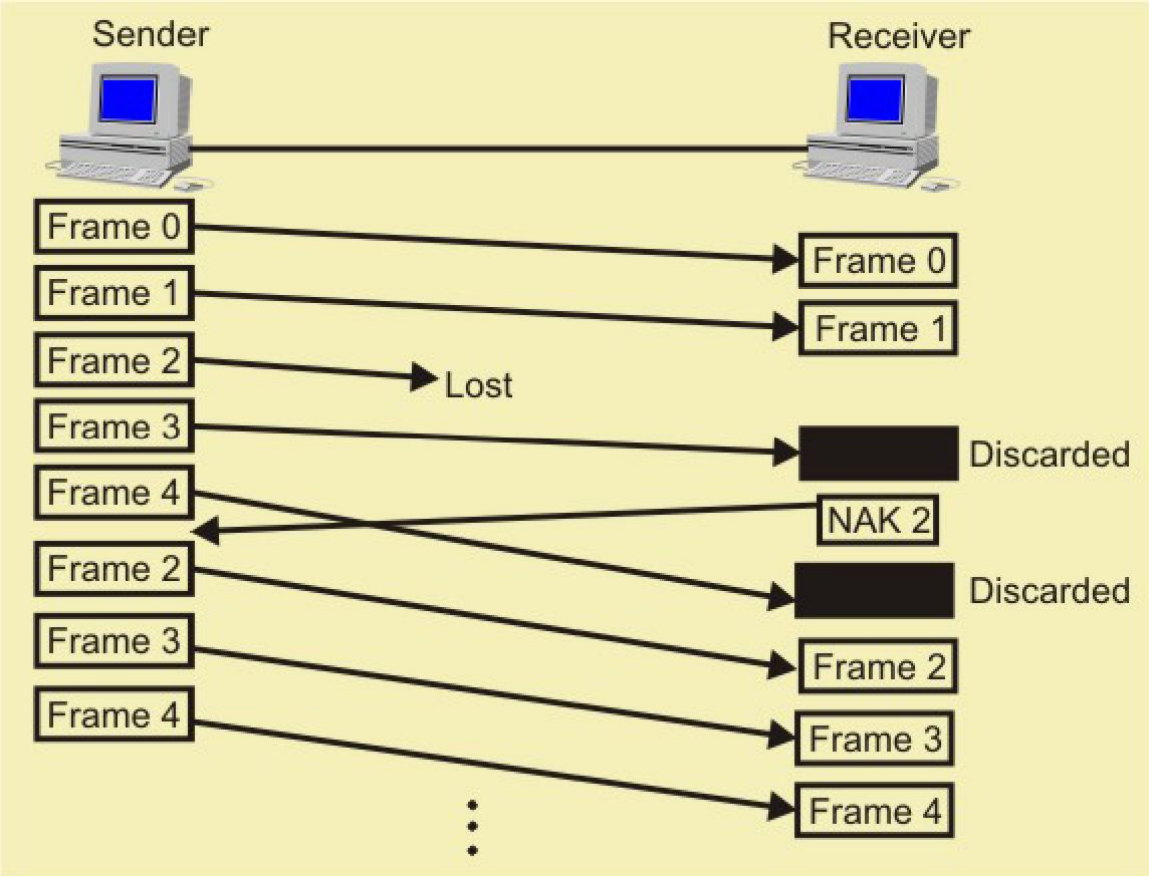
The window itself is an abstraction; three variables define its size and location at any time. We call these variables Sf(send window, the first outstanding frame), Sn (send window, the next frame to be sent), and Ssize (send window, size). The variable Sf defines the sequence number of the first (oldest) outstanding frame. The variable Sn holds the sequence

number that will be assigned to the next frame to be sent. Finally, the variable Ssize defines the size of the window, which is fixed in our protocol. The Figure-b shows how a send window can slide one or more slots to the right when an acknowledgment arrives from the other end.

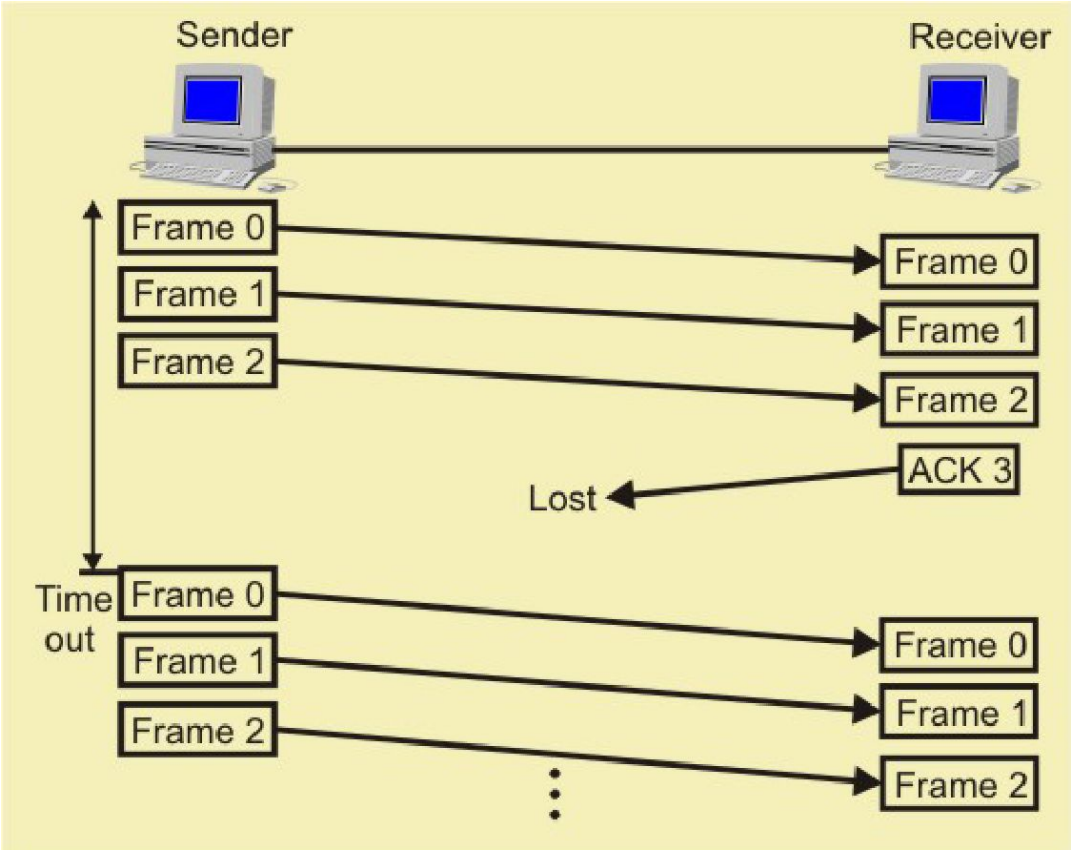
The acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame. In in the figure-b, frames 0, 1, and 2 are acknowledged, so the window has slid to the right three slots. Note that the value of Sf is 3 because frame 3 is now the first outstanding frame.



Frames in error in go-Back-N ARQ



Lost Frames in Go-Back-N ARQ



Lost ACK in Go-Back-N ARQ

Advantages:

Sender can send many frame at a time.
Timer can be set for a group of frame.
One ACK can acknowledge more than one frame.

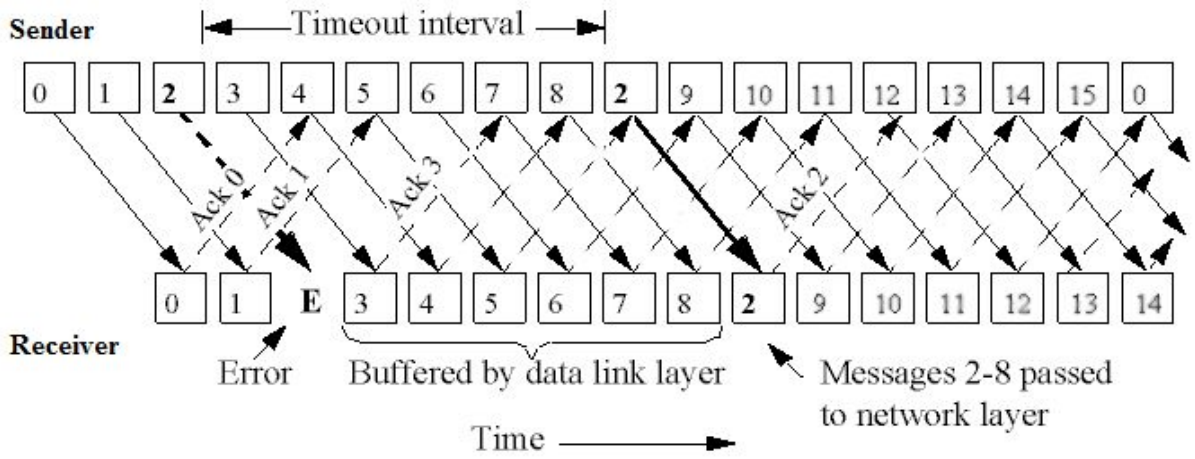
b.Definition of Selective Repeat ARQ:

Selective Repeat is part of the automatic repeat-request (ARQ). With selective repeat, the sender sends a number of frames specified by a window size even without the need to wait for individual ACK from the receiver as in Go-Back-N ARQ. The receiver may selectively reject a single frame, which may be retransmitted alone; this contrasts with other forms of ARQ, which must send

every frame from that point again. The receiver accepts out-of-order frames and buffers them. The sender individually retransmits frames that have timed out.

Operational principles:

- Selective Repeat ARQ overcomes the limitations of Go-Back-N by adding two new features:
    - Receiver window > 1 frame: Out-of-order but error-free frames can be accepted
    - Retransmission mechanism is modified: Only individual frames are retransmitted
- In this method, only specific damaged or lost frame is retransmitted
- Sender only retransmits frames for which a NAK is received.
- NAK number refer to the frame lost.
- If a frame is corrupted in transmit, a NAK is returned and the frame is resent out of sequence.
- The sender needs to maintain all data that hasn't been acknowledged yet.
- The receiving device must be able to sort the frames it has and insert the retransmitted frame into its proper place in the sequence.
- It has advantage that few re-transmisions than go-back-n. But complexity at sender and receiver is involved.
- Example: Frame 2 has an error, so receiver maintains buffer to store the next frames.



- **Damaged frames** :
    - In Selective reject, If a receiver receives a damaged frame, it sends the NAK for the frame in which error or damage is detected.
    - The NAK number, like in go-back-n also indicate the acknowledgement of the previously received frames and error in the current frame.
    - The receiver keeps receiving the new frames while waiting for the damaged frame to be replaced.
    - The frames that are received after the damaged frame are not be acknowledged until the damaged frame has been replaced.
- **Lost Frame** :
    - As in a selective repeat protocol, a frame can be received out of order and further they are sorted to maintain a proper sequence of the frames.
    - While sorting, if a frame number is skipped, the receiver recognise that a frame is lost and it sends NAK for that frame to the sender.
    - After receiving NAK for the lost frame the sender searches that frame in its window and retransmits that frame.
    - If the last transmitted frame is lost then receiver does not respond and this silence is a negative acknowledgement for the sender.

- **Lost Acknowledgement** :
    - In Selective reject, If the sender does not receive any ACK or the ACK is lost or damaged in between the transmission.
    - The sender waits for the time to run out and as the time run outs, the sender retransmit all the frames for which it has not received the ACK.
    - The sender identifies the loss of ACK with the help of a timer.

Advantages:

The advantage, however, is that it is not necessary to discard following correct data for one round-trip time before the transmitter can be informed that a retransmission is required. This is therefore preferred for links with low reliability and/or a high bandwidth-delay product.

d.Key Differences between Go-Back-N and Selective Repeat:

"Go-Back-N Protocol and "Selective Repeat Protocol" are the sliding window protocols. The sliding window protocol is primarily an error control protocol, i.e. it is a method of error detection and error correction. The basic difference between go-back-n protocol and selective repeat protocol

is that the “go-back-n protocol” retransmits all the frames that lie after the frame which is damaged or lost. The “selective repeat protocol” retransmits only that frame which is damaged or lost.

e.Comparison Chart:

BASIS FOR COMPARISON	GO-BACK-N	SELECTIVE REPEAT
Basic	Retransmits all the frames that sent after the frame which suspects to be damaged or lost.	Retransmits only those frames that are suspected to lost or damaged.
Bandwidth Utilization	If error rate is high, it wastes a lot of bandwidth.	Comparatively less bandwidth is wasted in retransmitting.
Complexity	Less complicated.	More complex as it require to apply extra logic and sorting and storage, at sender and receiver.
Window size	N-1	<= (N+1)/2
Sorting	Sorting is neither required at sender side nor at receiver side.	Receiver must be able to sort as it has to maintain the sequence of the frames.
Storing	Receiver do not store the	Receiver stores the frames

3. Working principle:

a.pseudo code for Go-Back-N:

GO-Back-N:

Sender:

```
Server{
        Int  window_size;//user input
        Char[][]  data = new Char[window_size][8];//buffer for data storage
        Ackno=window_size;
        While(DataExist){
                String codeword=" ";
                If(ackno ==n){
                        For(0-7=i times){
                                Data[i]  //8 byte array of ith row of data[][];
                                codeword+= toString(data[i])+CRC(data[i])+toString(i)+"  ";
                                //CRC() Method to generate CRC reminder
                        }
                }
                Else{
                        String codeword =slide_window(ackno,data[][]);
                        //slide_window() method to create new baffer and  codeword
//By collect some new data from file and the data[i] from the error frame to last frame

                }
                Codeword =ramdomERROR(codeword);
                // ramdomERROR() method to random error creating
                Send(codeword) //

        while(i=0-3times)//delay
        {
                If(get any ack/Nack){
                        Ackno=get_ackno();
                        Break;
                }
```

```
            Else{
                Ackno=n;
                If(last time of loop){
                    Resend(codeword);
                    I=0; //to create re_loop
                }
            }

        }

}
```

Client:

```
Client{
    While(true){
        String  data = receved_data();//get data from the sender
        Ackno = error_detection(data);//error_detection() method find error
in data or a frame missing and return frame number of the error frame
Otherwise return 16;
    Send(ackno);
    Print the sequally receved data in a file;
    }
}
```

Selective repeat ARQ:

Sender:

```
Server{
    Int  window_size;//user input
    Char[][]  data = new Char[window_size][8];//buffer for data storage
    Ackno=16;
    While(DataExist){
        String codeword=" ";
        If(ackno ==16){
            For(0-7=i times){
                Data[i]  //8 byte array of ith row of data[][];
                codeword+= toString(data[i])+CRC(data[i])+toString(i)+"  ";
                //CRC() Method to generate CRC reminder
```

```
                }
        }
        Else{
                String codeword =slide_window(ackno,data[][]);
                //slide_window() method to create new baffer and  codeword
//By adding only the error frames
        }
        Codeword =ramdomERROR(codeword);
        // ramdomERROR() method to random error creating
        Thread.Send(codeword) //

    while(i=0-3times)//delay
    {
        If(get any ack/Nack){
                Ackno=Thread.get_ackno();
                Break;
        }
        Else{
                Ackno=16;
                If(last time of loop){
                        Resend(codeword);
                        I=0;
                }
        }
        Delay(100)
    }

}
```

Client:

```
Client{
    While(true){
        String  data = receved_data();//get data from the sender
        Ackno = error_detection(data);//error_detection() method find error
in data or a frame missing and return frame number of the error frame
    Otherwise return 16;
    Send(ackno);
```

```
        //if a frame is missing take the next data in buffer until the frame
retransmitted.
        Print the secqually received data in a file;
        }
}
```

b.Main challenges and their solutions:

In this experiment there were some challenge we had to meet them.one of them is frame handling when a NACK is get.In go-back-n we send the NACK frame again and the others whose sequel number is greater than NACK and we scan (N-NACK) frame from the input file.In selective repeat ARQ we send the NACK frames again and the others whose sequel number is greater than NACK and we scan (N-NACKs) frame from the input file

4. Applications:

The Transmission Control Protocol uses a variant of Go-Back-N ARQ to ensure reliable transmission of data over the Internet Protocol, which does not provide guaranteed delivery of packets; with Selective Acknowledgement (SACK), it uses Selective Repeat ARQ.

The ITU-T G.hn standard, which provides a way to create a high-speed (up to 1 Gbit/s) local area network using existing residential wiring (power lines, telephone lines, and coaxial cables), uses Selective Repeat ARQ to ensure reliable transmission over noisy media.

5. Comparative analysis

a.Why Selective Repeat performs better than Go-Back-N?

- GBN weakness is the fact that when the window size is too large, the number of packets in the pipeline grows and one packet error causes the retransmission of many packets unnecessarily.
- Selective Repeat solves this by acknowledging just the suspicious packets, which slightly makes performance better, but if a wrong window size is chosen, then the receiver doesn't know if a packet is being retransmitted or another packet is being sent by first time.

b.Is there a situation where the reverse is true?

In high bit rate data transmission systems with ARQ error control, the throughput efficiency is a function of bit error rate, block or packet size, and the effect of significant round trip delays such as may be experienced in satellite communication systems. The selective-repeat ARQ scheme is capable of providing superior throughput performance independent of round trip delay, but requires excessively large receiver buffers; as a result the inferior Go Back N procedure is commonly adopted. This paper analyzes a class of mixed-mode ARQ protocol models which incorporate a selective repeat mode with finite receiver buffer.

c.The throughput for Go-Back-N and Selective Repeat ARQ:

For the highest possible throughput, it is important that the transmitter is not forced to stop sending by the sliding window protocol earlier than one round-trip delay time (RTT). The limit on the amount of data that it can send before stopping to wait for an acknowledgment should be larger than the bandwidth-delay product of the communications link.
p - loss probability
t_trans - pkt transmission time
rtt - round trip time

tput_GB = (1-p)/(p rtt + t_trans)
tput_SR = (1-p)/t_trans

d. Create graphs for the following:

(y-axis: throughput of the protocols) vs (x-axis: window size) (Line chart)

o Vary the window size from 4 to 24 (increase 4 on each step)

o For a specific window size calculate throughput for both the protocols and plot them in the graph

o Discuss, why the graph has taken such shapes?

(y-axis: throughput of the protocols) vs (x-axis: packet size) (Column Chart)

o Vary the packet size from 8 to 64 bytes (increase 8 on each step)

o For a specific packet size calculate throughput for both the protocols and plot them in the graph

o Discuss, why the graph has taken such shapes?

(y-axis: throughput of the protocols) vs (x-axis: packet loss rate) (Line Chart)

o Vary the loss rate from 0 to 80% (increase 10 on each step)

o For a specific loss rate calculate throughput for both the protocols and plot them in the graph

e. Discuss, why the graph has taken such shapes?

6. Observation:

In this report, performance analysis of two ARQ protocols are analyzed . It was shown that SR-ARQ has the best performance in both cases. It was also shown that if we increase the packet size for SW and GBN ARQ schemes, the throughput would also slightly increase. It was shown that SR-ARQ performs better with variable length packet size. It was also cleared that the performance of GBN-ARQ improves with multi-copy transmission when the channel error rate is high, while degrades when the channel bit error rate is high.

7. Conclusion:

It was a great experience to implement Go-Back-N and Selective repeat ARQ.It was our first time ,so we have to face a lot of challenges.but our teachers were very helpful.We were facing problem at the Selective repeat ARQ in Threading and controlling buffer size. But at the end of the day,we became successful in implement Go-Back-N and Selective repeat ARQ .