**Assignment No. :**

**Title:** Sliding Window Protocol

**Aim:** Implementation of sliding window protocol using C++

**Objective:** To study the Sliding Window Protocol

**Theory:**

A sliding window protocol is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the Data link layer (OSI model) as well as in the Transmission Control Protocol (TCP).

Conceptually, each portion of the transmission (packets in most data link layers, but bytes in TCP) is assigned a unique consecutive sequence number, and the receiver uses the numbers to place received packets in the correct order, discarding duplicate packets and identifying missing ones. The problem with this is that there is no limit on the size of the sequence number that can be required.

By placing limits on the number of packets that can be transmitted or received at any given time, a sliding window protocol allows an unlimited number of packets to be communicated using fixed-size sequence numbers. The term "window" on the transmitter side represents the logical boundary of the total number of packets yet to be acknowledged by the receiver.

**Types of sliding window protocols:-**

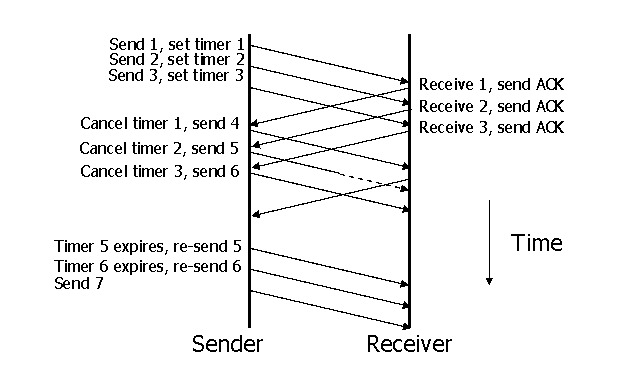
1. **Go-Back-N**

Go-Back-N is a specific instance of the automatic repeat request (ARQ) protocol, in which the sending process continues to send a number of frames specified by a window size even without receiving an acknowledgement (ACK) packet from the receiver. It is a special case of the general sliding window protocol with the transmit window size of N and receive window size of 1.

The receiver process keeps track of the sequence number of the next frame it expects to receive, and sends that number with every ACK it sends. The receiver will discard any frame that does not have the exact sequence number it expects (either a duplicate frame it already acknowledged or an out-of-order frame it expects to receive later) and will resend an ACK for the last correct in-order frame.

Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to the sequence number of the last ACK it received from the receiver process and fill its window starting with that frame and continue the process over again.

The receiver refuses to accept any packet but the next one in sequence. If a packet is lost in transit, following packets are ignored until the missing packet is retransmitted, a minimum loss of one round trip time. For this reason, it is inefficient on links that suffer frequent packet loss.



1. **Selective repeat**

The most general case of the sliding window protocol is Selective Repeat ARQ. This requires a much more capable receiver, which can accept packets with sequence numbers higher than the current sequence no. and store them until the gap is filled in.

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.The receiver’s window size need only be larger than the number of consecutive lost packets that can be tolerated. Thus, small values are popular; 2 is common.

We have used Go-Back-N protocol in our program

**Algorithm:**

**Sender**

* + Create a socket and establish a connection with the receiver.
  + Loop(till user does not enter exit)
  + Accept data to be transmitted(fr) and window size from the user
  + Initialize the buffer(fr1) used to store data of size=window size
  + Store data of size=window size in buffer (fr1).This data is packed along with frame no. to send
  + Start the timer and send data to receiver
  + Receive the cumulative acknowledgment frame.
  + Check if acknowledgment is negative. If it is nack resend the frames in window from the frame whose nack has been received else continue.
  + Check if timer expires if true send frames from the frame for which timer expired else continue
  + Set the counter to frame no. received in the acknowledgement and end loop
  + Once all frames have been sent successfully send “msg\_receive” to indicate end of message
  + Close socket and exit

**Receiver:**

* + Create a socket and establish a connection with the sender.
  + Initialize buffer(receive) to store received data
  + Receive frames from the sender
  + Unless “msg\_receive” is received continue
  + Check for duplicity and display each frame if it’s not duplicate else ignore
  + Ask user if all data displayed is right
  + If answer is ‘n’ send nack along with frame no. of damaged frame
  + Else if answer is ’y’ pack ack frame with frame no. the receiver is expecting next and send to sender (this step is applied even if duplicate data is received to indicate the frame no. receiver is expecting)
  + If data received was msg\_receive the complete message is displayed which was stored in another buffer (string).
  + Close socket and exit

**Assumptions:**

1. 1 Frame=1 character entered by user.
2. Use of cumulative acknowledgement for correct reception (however, NACK is sent for only 1 frame i.e. the frame which is wrong).
3. Time out is in 10 sec

**Note: write the mathematical model and conclusion**

**FAQs:**

* 1. What is the difference between selective repeat and go-back-n?
  2. Station A uses 32 byte packets to transmit messages to Station B using a sliding window protocol. The round trip time delay between A and B is 80ms and the bottleneck bandwidth on the path A and B is 128kbps. What is the optimal window size that A should use?
  3. What is the maximum window size for data transmission using the selective repeat protocol with n-bit frame sequence number?

Q. Station A uses 32 byte packets to transmit messages to Station B using a sliding window protocol. The round trip time delay between A and B is 80ms and the bottleneck bandwidth on the path A and B is 128kbps. What is the optimal window size that A should use?  
a) 20          b) 40            c)160        d)320  
  
Ans: Option b  
Explanation:

Since we have to find the optimal window size, we are indirectly said that maximum throughput should be achieved. Hence the maximum throughput that we can achieve is 128kbps.

RTT = 80ms  
Packet size = 32 x 8 bits  
  
We know, Throughput =1 window/ RTT  
Therefore optimal window size = Throughput x RTT   
                                              = 128x103 x 80x10-3  
                                              = 128 x 80 bits  
Therefore, optimal window size in terms of packets = (128 x 80) / (32 x 8) = 40

Q.The maximum window size for data transmission using the selective reject protocol with n-bit frame sequence numbers is  
a) 2n            b) 2n-1                    c) 2n-1                   d)2n-2  
  
Ans: Option b  
Explanation:

Selective Reject (or Selective Repeat) protocol is one of the automatic repeat-request (ARQ) techniques used for communications.

In SR protocol the window size of the receiver and sender must be (N+1)/2, where N is the maximum sequence number.

If N is the maximum available sequence numbers then, the window size of both sender and receiver must be N/2.

If n is the number of bits in the frame sequence field then, the window size of both sender and receiver must be  2n-1.