CNL Assignment

- **TITLE:**

  Write a program for the Simulation of Sliding Window Protocols

- **OBJECTIVES:**
- **PROBLEM DEFINITION:**

  Part A: Write a program to simulate Go back N Mode of Sliding Window Protocol in peer to peer mode.

  Part B: Write a program to simulate Selective Repeat Mode of Sliding Window Protocol in peer to peer mode.

- **OUTCOME:**
- **SOFTWARE & HARDWARE REQUIREMENTS**

  Hardware: - Intel(R) Core (TM) (Preferred any latest or compatible processor), 2 GB RAM, 320GB/1TB HDD (Available System Hardware can also be considered)
  Operating System recommended: - 64-bit Open source Linux or its derivative
  Programming tool: C-C++, Wireshark/Ethereal and Packet Tracer
- **THEORY:**
- The most important responsibilities of the data link layer are flow control and error control. Collectively, these functions are known as data link control.
- The flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgement.
- DLL- combine framing, flow control, and error control to achieve the delivery of data from one node to another.
- The protocols are normally implemented in software by using one of the common programming languages.
- To make our discussions language-free, we have written in pseudo code a version of each protocol that concentrates mostly on the procedure instead of delving into the details of language rules.

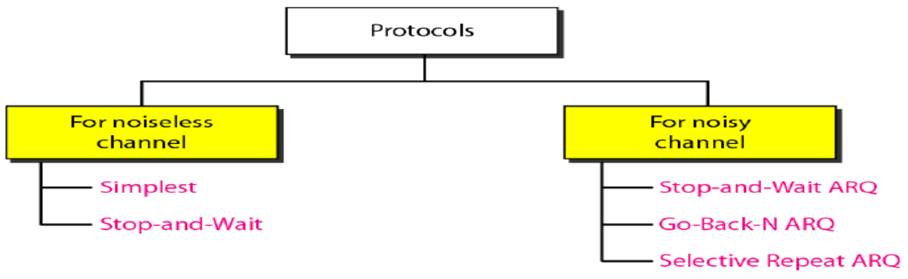


Figure 1: Flow Control Protocols

- **Noiseless channel:**

An ideal channel in which no frames are lost, duplicated, or corrupted. There are two protocols for this type of channel.

- Simplest Protocol
- Stop-and-Wait Protocol

- **Noisy Channels**

Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent (imaginary/ unreal).There are three protocols in this section that use error control.

- Stop-and-Wait Automatic Repeat Request
- Go-Back-N Automatic Repeat Request
- Selective Repeat Automatic Repeat Request

In this assignment we will focus Go-Back-N and Selective Repeat

- **Go-Back-N**

- Go-Back-N ARQ 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.
- It can transmit N frames to the peer before requiring an ACK.
- 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.

- Go-Back-N ARQ is a more efficient use of a connection than Stop-and-wait ARQ, since unlike waiting for an acknowledgement for each packet; the connection is still being utilized as packets are being sent. In other words, during the time that would otherwise be spent waiting, more packets are being sent.

- However, this method also results in sending frames multiple times – if any frame was lost or damaged, or the ACK acknowledging them was lost or damaged, then that frame and all following frames in the window (even if they were received without error) will be re-sent. To avoid this, Selective Repeat ARQ can be used.

- Sender is allowed to transmit multiple packets without waiting for an acknowledgement, but is constrained to have no more than some maximum allowable number (N)
- Use cumulative acknowledgement
- Discard out-of-order packets, no receiver buffering

- **Choosing a Window size (N)**

There are a few things to keep in mind when choosing a value for N:

- The sender must not transmit too fast. N should be bounded by the receiver's ability to process packets.
- N must be smaller than the number of sequence numbers (if they are numbered from zero to N) to verify transmission in cases of any packet (any data or ACK packet) being dropped.
- Given the bounds presented in (1) and (2), choose N to be the largest number possible
  - In Go-Back-N Protocol, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

- **Design of Go-Back-N ARQ**

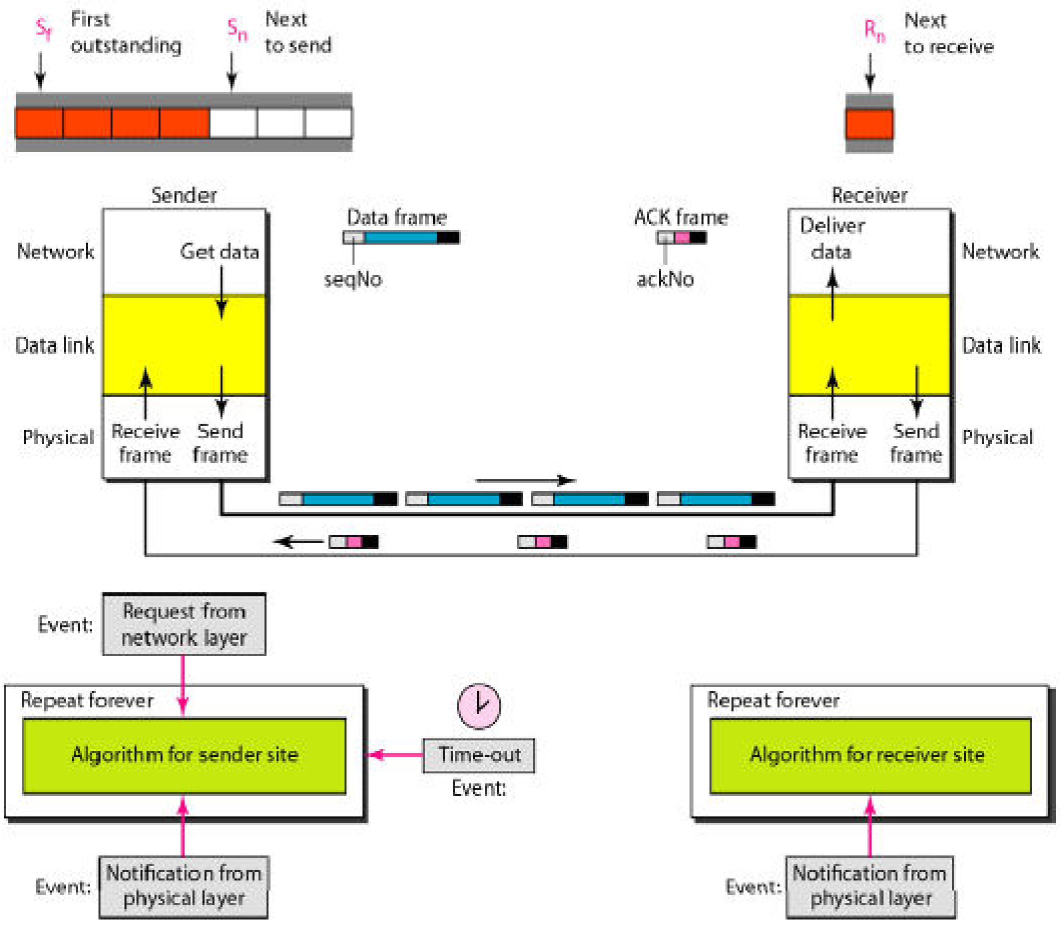


Figure 2: Design of Go-Back-N ARQ

- **Window size for Go-Back-N ARQ**

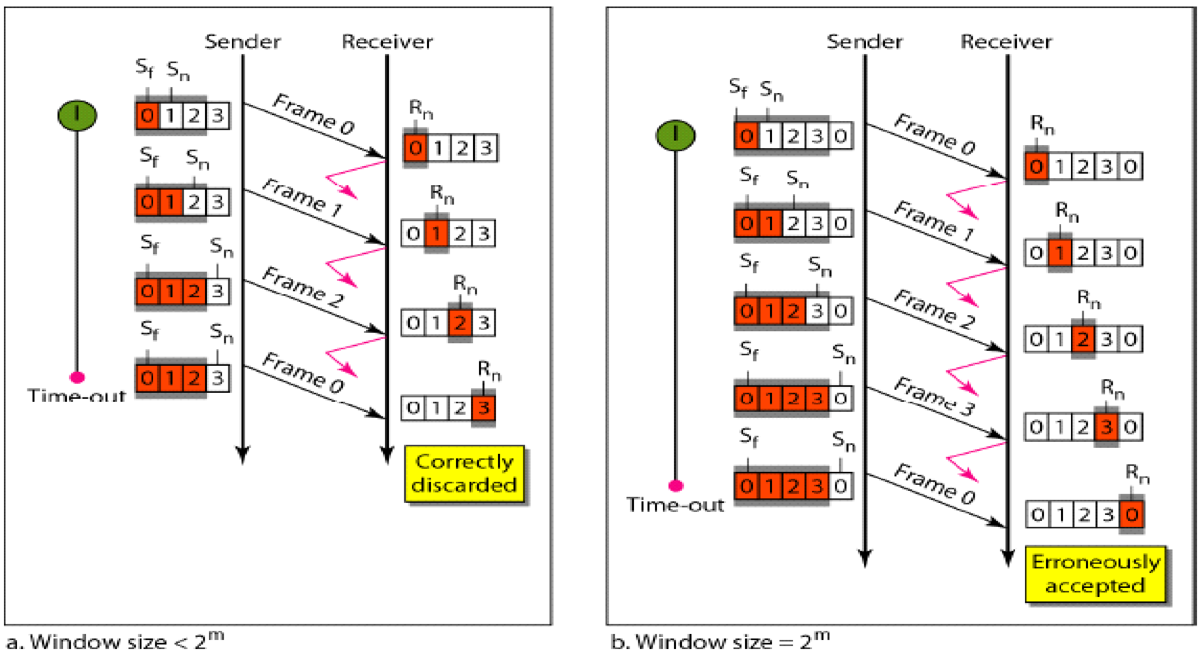


Figure 3: Window size for Go-Back-N ARQ

- In Go-Back-N ARQ, the size of the send window must be less than 2^m; the size of the receiver window is always 1.
- **Go-Back-N sender algorithm**

```
$S_w$ = $2^m$ - 1;
$S_f$ = 0;
$S_n$ = 0;

while (true)                              //Repeat forever
{
 WaitForEvent();
  if(Event(RequestToSend))                //A packet to send
  {
     if($S_n$-$S_f$ >= $S_w$)             //If window is full
            Sleep();
     GetData();
     MakeFrame($S_n$);
     StoreFrame($S_n$);
     SendFrame($S_n$);
     $S_n$ = $S_n$ + 1;
     if(timer not running)
          StartTimer();
  }

```

(continued)

```
  if(Event(ArrivalNotification))  //ACK arrives
  {
     Receive(ACK);
     if(corrupted(ACK))
          Sleep();
     if((ackNo>$S_f$)&&(ackNo<=$S_n$))  //If a valid ACK
     While($S_f$ <= ackNo)
      {
       PurgeFrame($S_f$);
       $S_f$ = $S_f$ + 1;
      }
      StopTimer();
  }

  if(Event(TimeOut))                   //The timer expires
  {
   StartTimer();
   Temp = $S_f$;
   while(Temp < $S_n$);
    {
     SendFrame($S_f$);
     $S_f$ = $S_f$ + 1;
    }
  }
}
```

- **Go-Back-N receiver algorithm**

```
$R_n$ = 0;

while (true)                         //Repeat forever
{
  WaitForEvent();

  if(Event(ArrivalNotification)) /Data frame arrives
  {
     Receive(Frame);
     if(corrupted(Frame))
          Sleep();
     if(seqNo == $R_n$)              //If expected frame
     {
       DeliverData();                //Deliver data
       $R_n$ = $R_n$ + 1;            //Slide window
       SendACK($R_n$);
     }
  }
}
```

- **Example of Go-Back-N ARQ:**

Figure shows an example of Go-Back-N. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost. After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK 3.

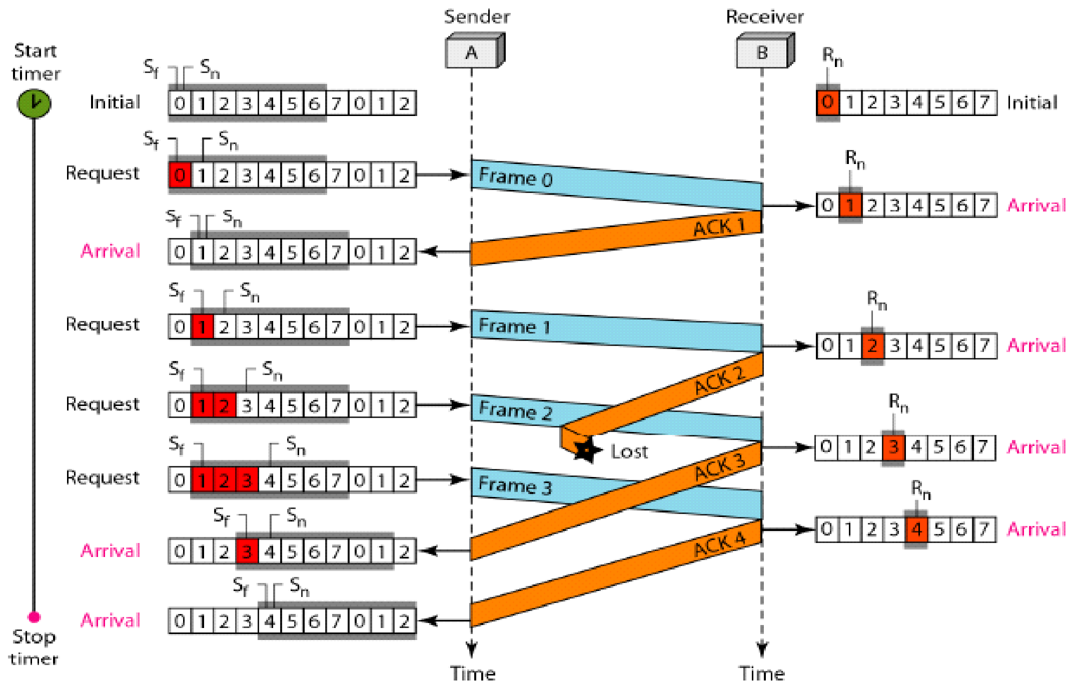


Figure 4: Example of Go-Back-N ARQ

- **Selective Repeat**

- 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 contrast 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.

- It may be used as a protocol for the delivery and acknowledgement of message units, or it may be used as a protocol for the delivery of subdivided message sub-units.

- When used as the protocol for the delivery of messages, the sending process continues to send a number of frames specified by a window size even after a frame loss.

- Unlike Go-Back-N ARQ, the receiving process will continue to accept and acknowledge frames sent after an initial error; this is the general case of the sliding window protocol with both transmit and receive window sizes greater than 1.

- The receiver process keeps track of the sequence number of the earliest frame it has not received, and sends that number with every acknowledgement (ACK) it sends.

- If a frame from the sender does not reach the receiver, the sender continues to send subsequent frames until it has emptied its window.

- The receiver continues to fill its receiving window with the subsequent frames, replying each time with an ACK containing the sequence number of the earliest missing frame.

- Once the sender has sent all the frames in its window, it re-sends the frame number given by the ACKs, and then continues where it left off.

- The size of the sending and receiving windows must be equal, and half the maximum sequence number (assuming that sequence numbers are numbered from 0 to n−1) to avoid miscommunication in all cases of packets being dropped.

- To understand this, consider the case when all ACKs are destroyed. If the receiving window is larger than half the maximum sequence number, some, possibly even all, of the packets that are resent after timeouts are duplicates that are not recognized as such.

- Out-of-order packets are buffered until any missing packets are received (receiver also has a window)

- Re-acknowledgement is required, otherwise the sender's window will never move.

- The sender moves its window for every packet that is acknowledged.

- **Design of Selective Repeat ARQ**

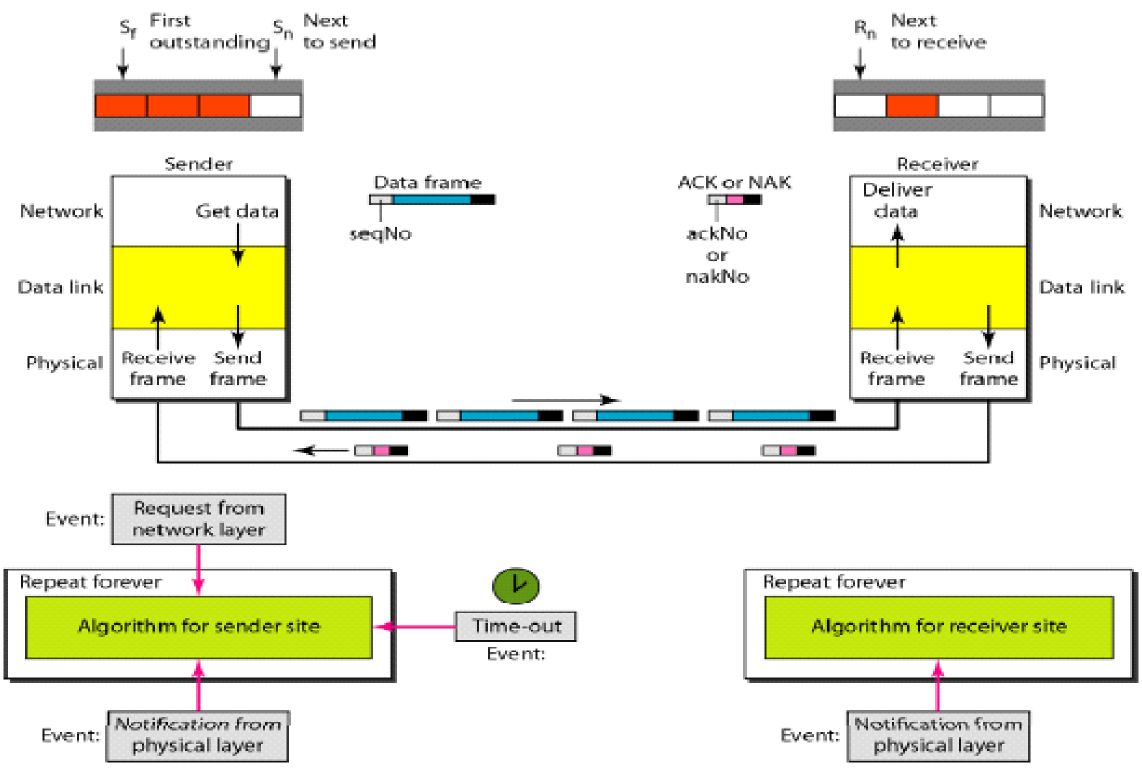


Figure 5: Design of Selective Repeat ARQ

- **Selective Repeat ARQ, window size**

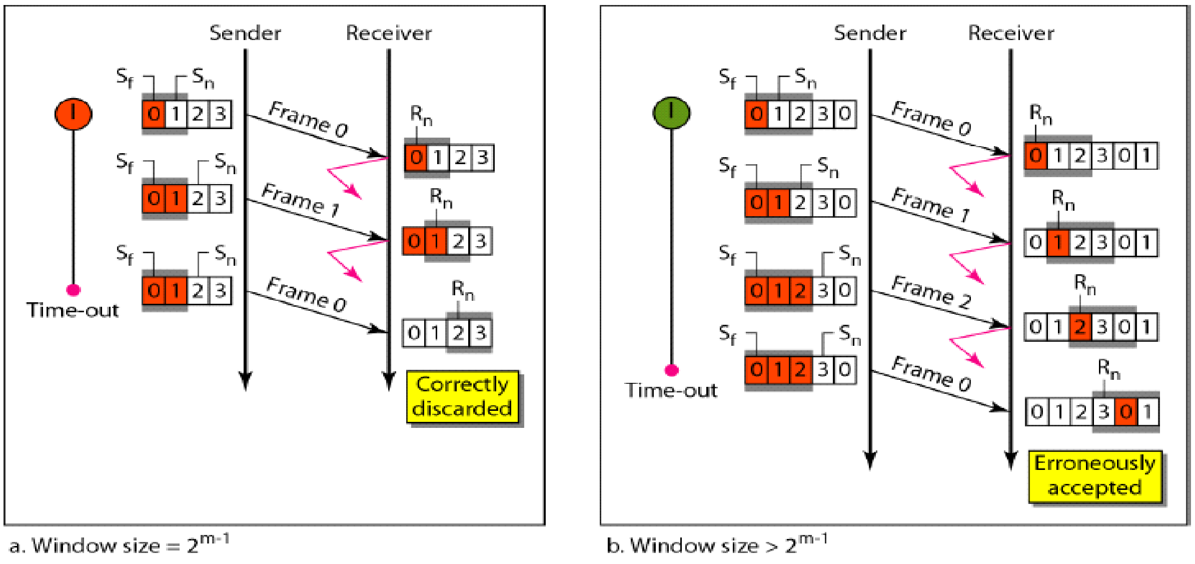
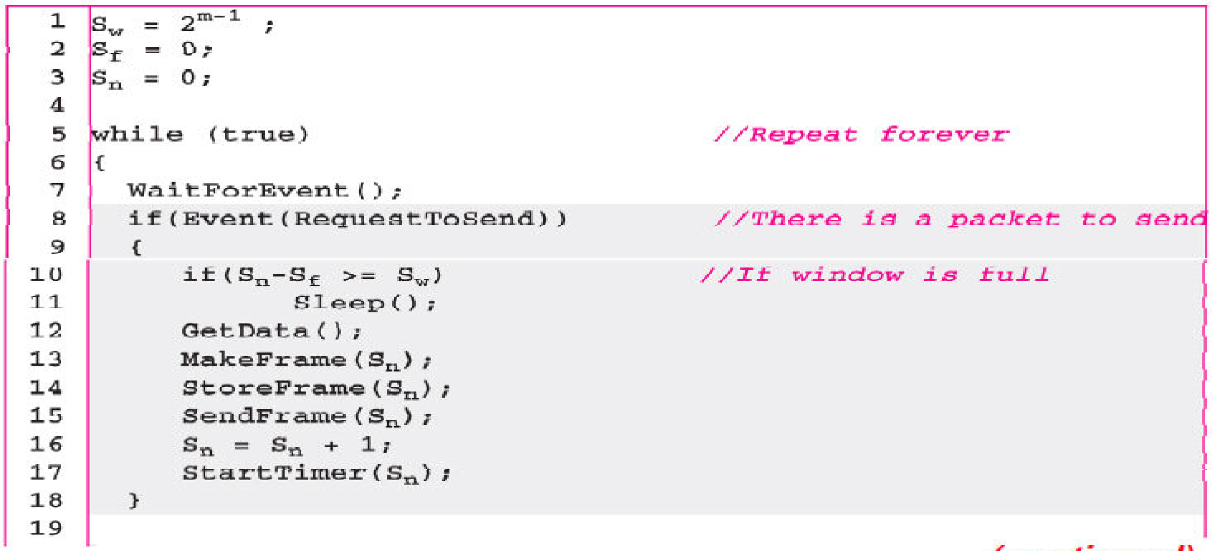


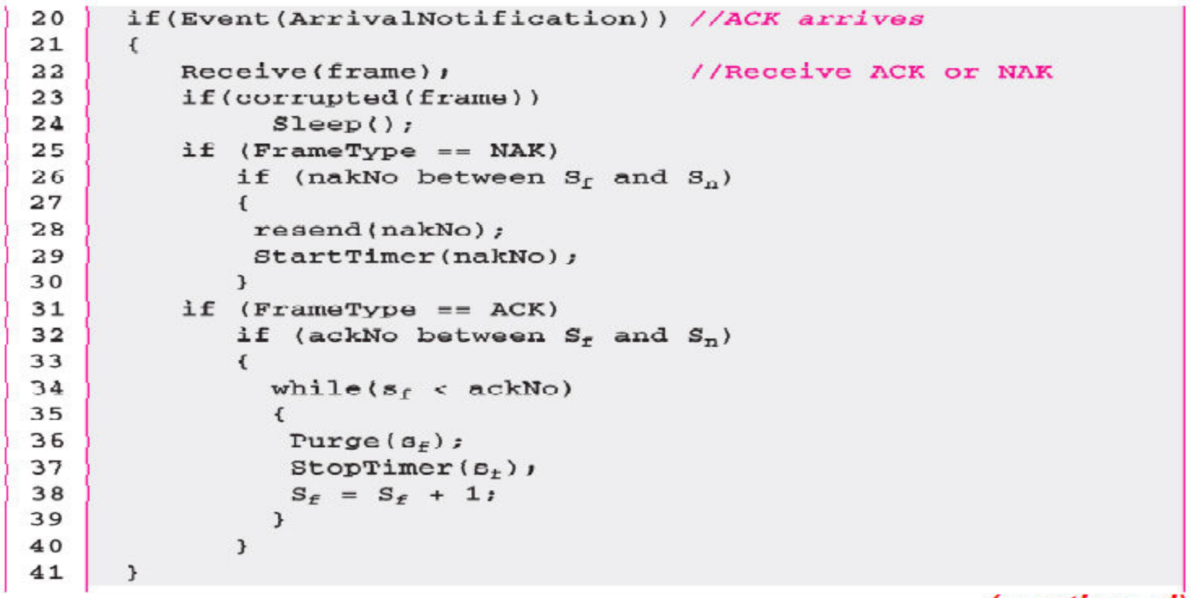
Figure 6: Window size of Selective Repeat ARQ

- In Selective repeat ARQ, the size of the sender and receiver window must be most one –half 2^m.
- **Sender-site Selective Repeat algorithm**



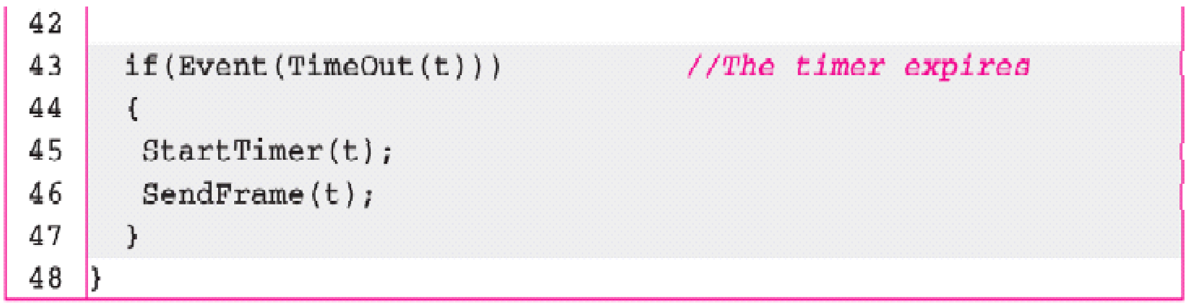
```
$S_w$ = $2^{m-1}$ ;
$S_f$ = 0;
$S_n$ = 0;

while (true)                              //Repeat forever
{
  WaitForEvent();
  if(Event(RequestToSend))                //There is a packet to send
  {
     if($S_n$-$S_f$ >= $S_w$)                  //If window is full
           Sleep();
     GetData();
     MakeFrame($S_n$);
     StoreFrame($S_n$);
     SendFrame($S_n$);
     $S_n$ = $S_n$ + 1;
     StartTimer($S_n$);
  }

```

(continued)



```
  if(Event(ArrivalNotification)) //ACK arrives
  {
     Receive(frame);              //Receive ACK or NAK
     if(corrupted(frame))
          Sleep();
     if (FrameType == NAK)
        if (nakNo between $S_f$ and $S_n$)
        {
         resend(nakNo);
         StartTimer(nakNo);
        }
     if (FrameType == ACK)
        if (ackNo between $S_f$ and $S_n$)
        {
          while($s_f$ < ackNo)
          {
           Purge($s_f$);
           StopTimer($s_f$);
           $S_f$ = $S_f$ + 1;
          }
        }
  }
```

(continued)



```

  if(Event(TimeOut(t)))          //The timer expires
  {
   StartTimer(t);
   SendFrame(t);
  }
}
```

- **Receiver-site Selective Repeat algorithm**

```
R_n = 0;
NakSent = false;
AckNeeded = false;
Repeat(for all slots)
    Marked(slot) = false;

while (true)                                          //Repeat forever
{
  WaitForEvent();

  if(Event(ArrivalNotification))                      /Data frame arrives
  {
     Receive(Frame);
     if(corrupted(Frame))&& (NOT NakSent)
     {
      SendNAK(R_n);
      NakSent = true;
      Sleep();
     }
     if(seqNo <> R_n)&& (NOT NakSent)
     {
      SendNAK(R_n);
      NakSent = true;
      if ((seqNo in window)&&(!Marked(seqNo))
      {
       StoreFrame(seqNo)
       Marked(seqNo)= true;
       while(Marked(R_n))
       {
        DeliverData(R_n);
        Purge(R_n);
        R_n = R_n + 1;
        AckNeeded = true;
       }
        if(AckNeeded);
        {
        SendAck(R_n);
        AckNeeded = false;
        NakSent = false;
        }
      }
     }
  }
}
```

- **Example of Selective Repeat ARQ:**

This example is similar to Example in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure shows the situation. One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives. The timer for frame 1 starts at the second request restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event. At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked, but it cannot be delivered because frame 1 is missing. At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered. Only at the last arrival, when finally a copy of frame 1 arrives, can frames 1, 2, and 3 be delivered to the network layer. There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window. Another important point is that a NAK is sent after the second arrival, but not after the third, although both situations look the same. The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames. The second NAK would still be NAK1 to inform the sender to resend frame 1 again; this has already been done. The first NAK sent is remembered (using the nakSent variable) and is not sent again until the frame slides. A NAK is sent once for each window position and defines the first slot in the window. The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to n frames are delivered in one shot, only one ACK is sent for all of them.

- **Flow diagram of Example:**

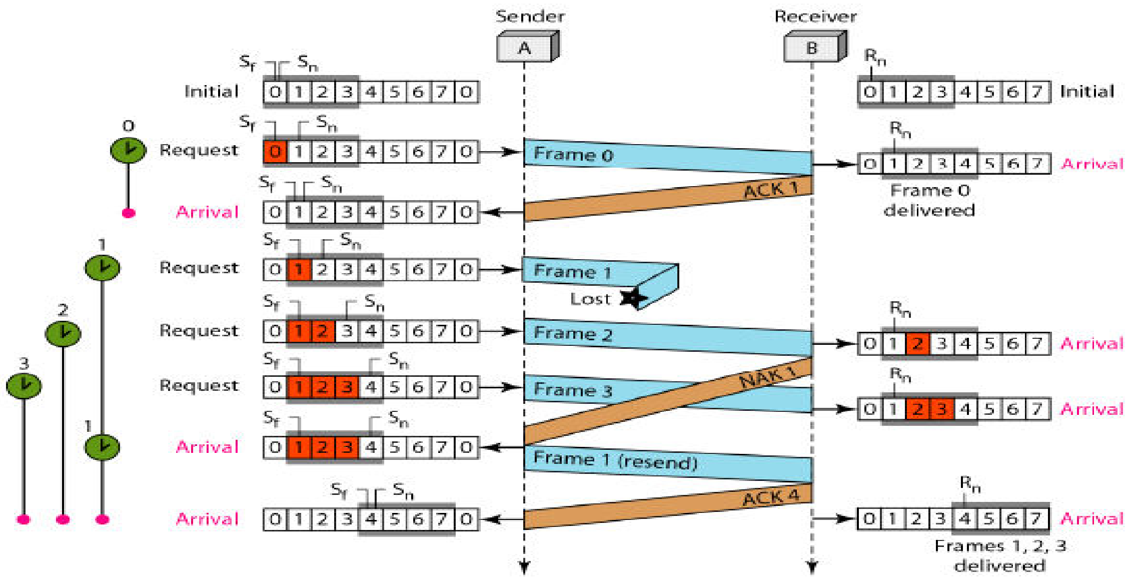


Figure 7: Example of Selective Repeat ARQ

- **CONCLUSION:**

Hence we have learned flow control through simulation of sliding window protocols- Go-Back-N and Selective Repeat.

- **REFERECNES:**

- Fourauzan B., "Data Communications and Networking", 5th Edition, Tata McGraw- Hill, Publications, 2006