Data Link control

Flow and Error control

The most important responsibility of Data Link Layer (DLL) is flow control and error control. Collectively, it is known as Data Link control.

Flow control controls the data that can be sent before receiving an acknowledgment. ~~It is one of the~~ In most protocols, Flow control is set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the receiver. The flow control should be able not to overwhelm the receiver.

The rate of ~~xmission~~ processing is usually slower than rate of xmission. For this reason, each receiving device has a buffer ~~for~~ for storing incoming data until they are processed. If the buffer begins to fill up, the receiver must be able to tell to sender to halt xmission until it is once again able to receive.

Error control is both error detection & error correction. It allows the receiver to inform sender of any frames lost or damaged in transmission and co-ordinates retransmissions of those frames by sender. In DLL, error control refers primarily to error detection & retransmission. Error control in DLL is implemented simply: Any time an error is detected in an xchange, specified frames are retransmitted. The process is called Automatic Repeat Request (ARQ).

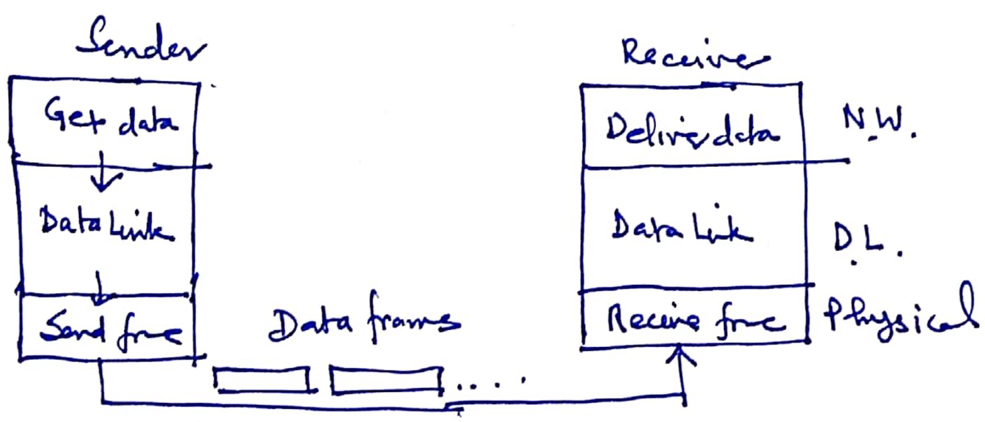
Protocols:

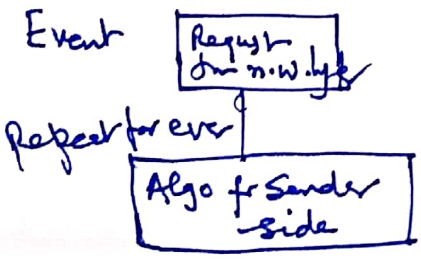
Data Link layer combines: framing, Flow control & error control to acheive delivery of data from one node to another. Protocols normally implemented in S.W.

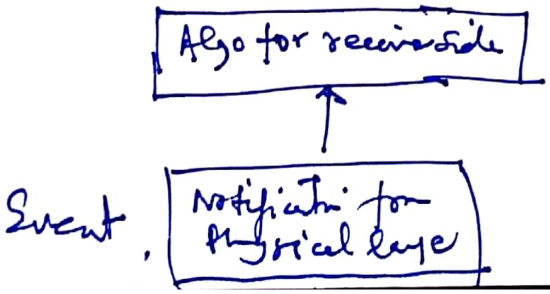
- Protocols
  - For noiseless channels
    - Simplest
    - STOP & WAIT
  - For noisy channels
    - Stop and Wait ARQ
    - Go back-N ARQ
    - Selective Repeat ARQ.

Simplest: No flow or error control. Unidirectional and data frames travel only in one direction; from Sender to receiver. We assume that receiver can handle any frame it receives with processing time that small enough to be neglegible.

Receiver is never overwhelmed with incoming frames.



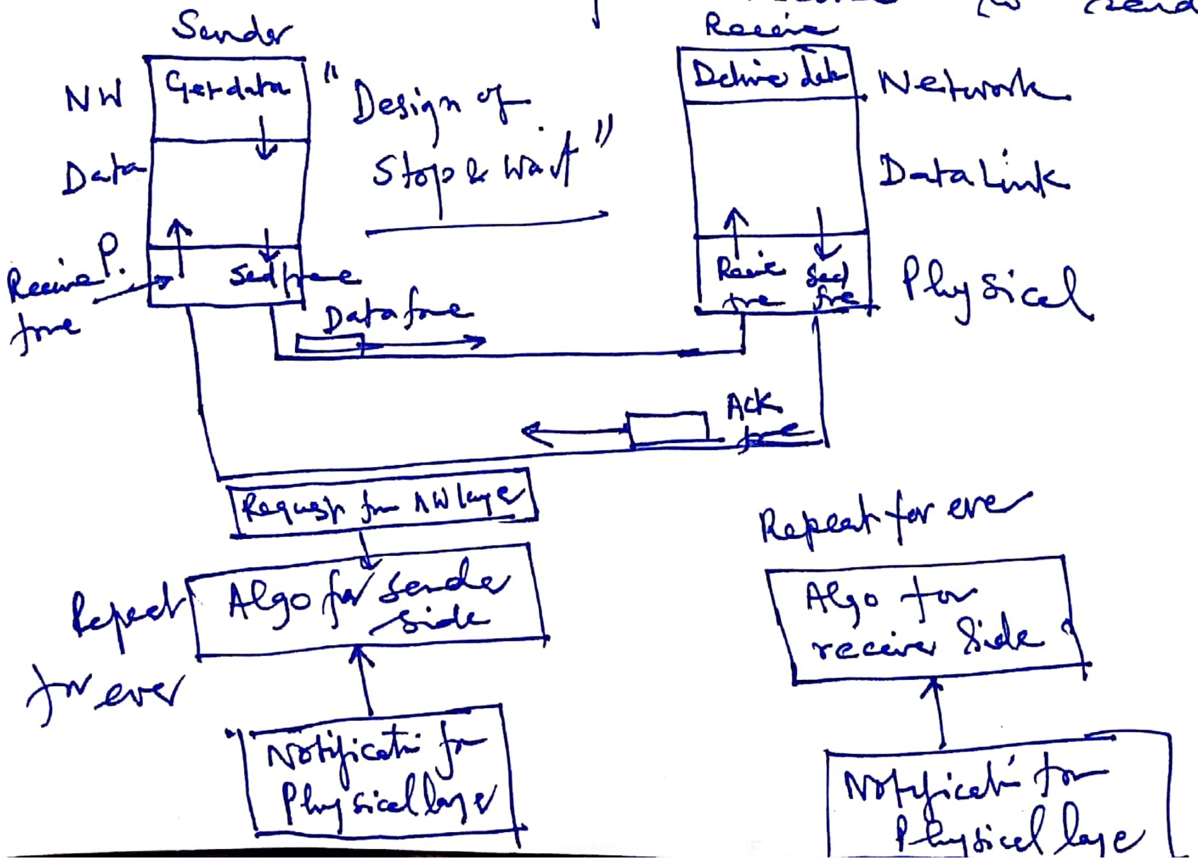




STOP and Wait Protocol

If the data frames arrive at receiver site faster than they can be processed, frames must be stored for until their use. Usually receiver does not have enough storage especially if it is receiving data from many sources. To prevent receiver from becoming overwhelmed with frames, we some how need to tell sender to slow down. This necessiate requirement of feedback from receiver to sender.

The required protocol is STOP and Wait Protocol. Here sender sends a frame, stops until it receive confirmation from receiver and then sends another frame. Therefore, we have unidirectional communication for data frames, but auxillary ~~acknowl~~ ACK frame travel from receiver to sender.



Noisy channels.

In actual practice ~~noisy~~ noise less channels are in existant. channels are usually noisy so transmitted data is ~~always~~ almost always received with a probability of error.

Therefore, it is important That flow control to be ~~is~~ always associated with error control mechanism.

First flow control ~~or~~ that is associated with Error control mechanism is:

STOP and Wait Automatic Repeat Request
(Stop and Wait ARQ).

This adds error control mechanism to STOP and Wait protocol.

We know that errors are genrally of two types

a) ~~D~~ Corrupted frames: We need to add redundancy bits to data frames. When the frame arrives at receiver site, it is checked and if found corrupted is silently discarded. Accordingly new frame (copy) is sent by Tx.

b) Lost frames: These are more difficult to handle. The received frame could be correct one, or a frame out of order. Therefore, the frame are numbered. When the receiver receives the frame that is out of order, this means that the frame was lost.

Therefore, in this protocol, corrupted or lost frames need to be resent. If the receiver does not respond when there is an error, how can sender know which frame to be resend.

To overcome this problem, sender keeps a copy of the sent frame. At the same time, it starts a timer. If timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held and timer is restarted.

Although, there are more than one copies of specific frame, only one specific frame needs ACK.

Since ACK frame can also be corrupted and lost, it too needs redundancy & sequence No. Therefore, ACK frame for this protocol has a sequence no. field. In this protocol, sender simply discards a corrupted ACK frame or ignores an out of order one.

In short: eg. Frame is sent / ACK lost (not received)
frame will be unnecessarily resent and it will be invalid frame for receiver

Therefore, receiver should have a way to know that frame is valid one for Acceptance.

Therefore, we give a sequence no. to frames.

Sequence no: A field is added to data frame to hold sequence no. of the frame.

Sequence no. should have minimum range so that less bits are used — otherwise this will contribute an overhead.

∴ We look for smallest range that provides unambiguous communication.

To reason out range of Sequence Nos reqd.

Assume x as sequence no, we need x+1 after that. There is no need of x+2. To show this

Assume sender has sent frame with sequence no. x

Three things can happen:

1. Frame (x) arrives safe to receiver site; receiver sends acknowledgment. Acknowledgment is received by sender side – sender will send next frame with sequence no. x+1.
2. Sender sends frame, it arrives safe to receiver side. Receiver sends acknowledgment. Acknowledgment is corrupted or lost. Sender resends frame x after time out. Note: here frame is duplicate one. Receiver expects (x+1) but it receives (x); receiver can recognize this fact
3. Frame is corrupted and never ~~recoo~~ reaches receiver site; sender resends frame (x) after time out.

Thus there is a need for seq. no. x and x+1; because the receiver needs to distinguish between 1. & 2. above. There is no need of x+2.

Let x = 0; x+1 = 1 ∴ Sequence Nos. 0, 1, 0, 1 . . . . .

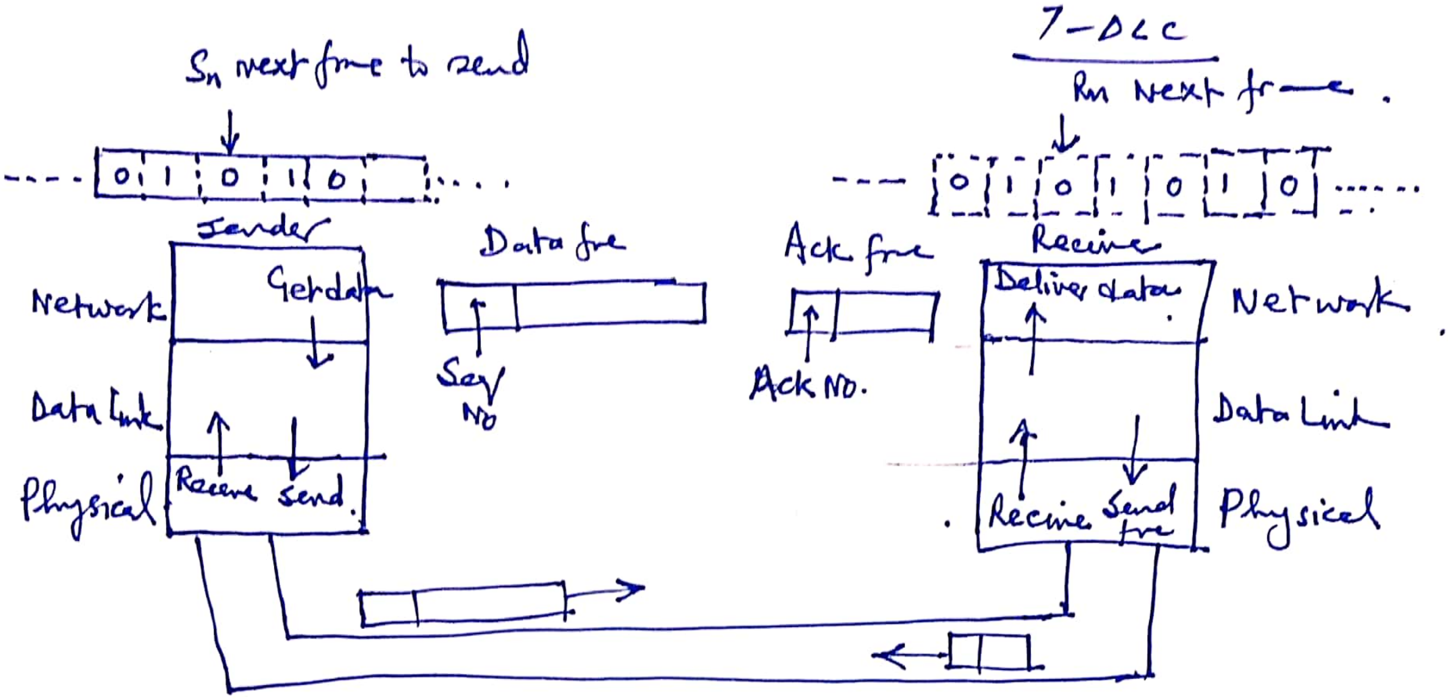
This is modulo-2 ∵ 0+1 = 1, 0+0 = 0

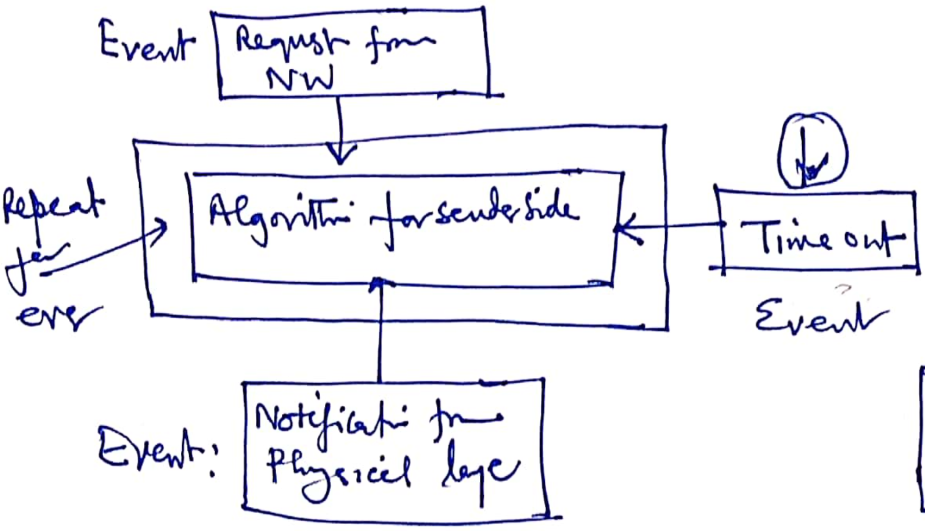
So Sequence no. must be suitable for both data frames & For Ack frames we use the convention:

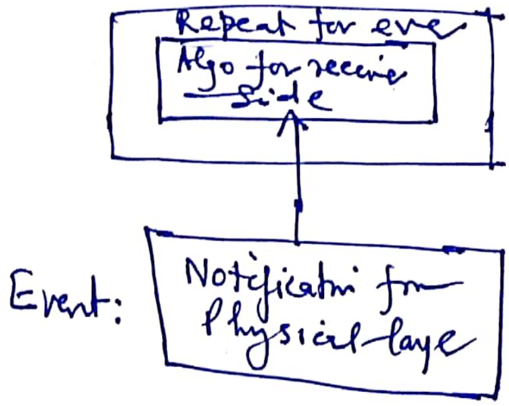
Ack no. announces seq. no. of next frame ~~received~~ expected by receiver.

If '0' has arrived safe, Receiver sends Ack (1) meaning frame 1 is expected next.

If '1' has arrived safe, receiver sends Ack (0) meaning '0' is expected







Design of Stop and Wait ARQ Protocol.

Here Sending frame keeps a copy of last frame Transmitted untill it receives an acknowledgment for That frame. Data frames use a Seq. NO. & ACK frame has an ack NO. The sender has a control variable Sn (Sender ~~acknowledgment NO~~ next frame to send), and holds ~~the~~ Sequence NO. for next frame to send (0 or 1).

The receiver has control variable Rn (Receive next frame expected)

When a frame is sent, the value of ~~Sn (mod 2)~~ Sn is incremented (modulo-2). Three events possible from sender side & one event possible ~~&~~ on receive side). Variable Sn points to slot that matches the sequence no of the frame that has been sent, but not acknowledged;
Rn points to the slot that matches the sequence no. of expected frame.

Efficiency: The Stop-and-Wait ARQ is v. inefficient if the channel is thick and long. (Thick ⇒ channel has large bandwidth; long ⇒ Round trip delay). This means if bandwidth delay product is high. i.e. channel is there, but we are using it inefficiently. Bandwidth delay product is a measure of the number of bits we can send out of our system while waiting for the news to receive; this is a case of poor utilization of channel / ~~channel~~ efficiency of Transmission.

To improve efficiency of x-mission, it is desirable that multiple frames are in transition while waiting for acknowledgement. We need to let more than one frame be outstanding to keep channel busy while sender is waiting for acknowledgement.

The first is called Go-Back-N Automatic Repeat Request.

In this protocol we can send several frames before receiving acknowledgement; we keep copy of these frames until acknowledgements arrive.

Seq. No.

Frames from sending station are numbered sequentially. ~~Since~~ we need to set a limit because we need to keep sequence No. corresponding to each frame.

For an m-bit Seq. No, the range of Seq No. 0 to $2^m - 1$

If m is 3; Seq Nos are from 0 – 7.

We can repeat Seq Nos. So Seq Nos are

0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0 . . . . . .

(Means Seq Nos. are modulo – 2^m.)

m is ~~Seq No~~ size of Seq No. field.

Sliding Window:

In this protocol sliding window is an abstract concept that defines range of Seq. Nos that is concern of sender and receiver.

The range of sender side: Send sliding window; Receiver side: Receive sliding window

Send sliding window is an imaginary box covering Seq Nos. that are in transit.

In each window position, some sequence Nos. define frames that can be sent; others define those that can be sent.

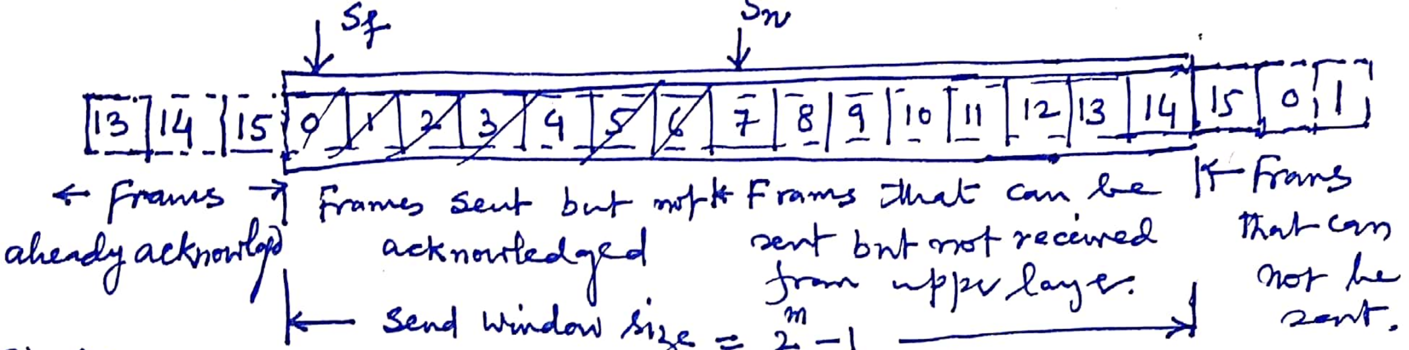
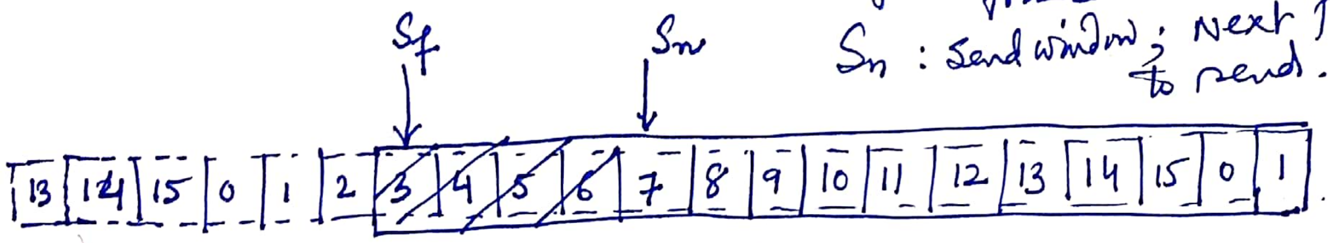


Fig-A-

(Send window before sliding): Sf: Send window, first outstanding frame
Sn: Send window; Next frame to send.



(Send window after sliding) Fig-B

The window at any time divides Seq. Nos. into four regions shown above:

1. Frames already acknowledged: Sender does not worry about these frames and does not keep copy of them.
2. The second region: Range of sequence nos. belongs to frames that are sent and have unknown status.
3. The sender needs to wait to see if these frames have been sent or were lost. (outstanding frames)
3. The frames that can be sent; however data packets have not been received by network layer.
4. The Seq. nos. that can not used unless window slides.

[Fig-A]

Fig. B shows how the send window can slide one or more slots to the right when an acknowledgement arrives from the other end.

Here acknowledgements are cumulative, meaning more than one frame can be acknowledged by ACK frame. In Fig B, frames 0, 1, 2 can be acknowledged so window has slid to right three slots.

∴ $S_f = 3$, because frame 3 is first outstanding frame.

The receive window makes sure that the correct data frames are received and the correct acknowledgement are sent.

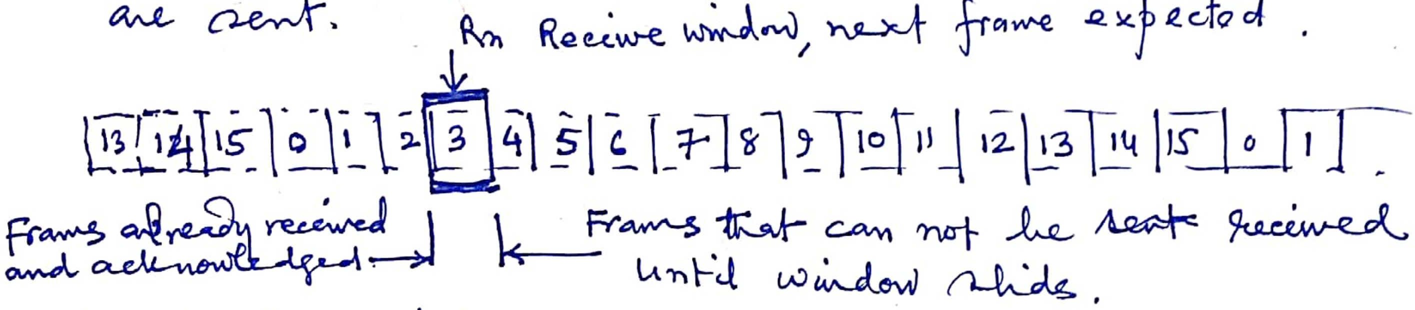


Fig: Receive Window

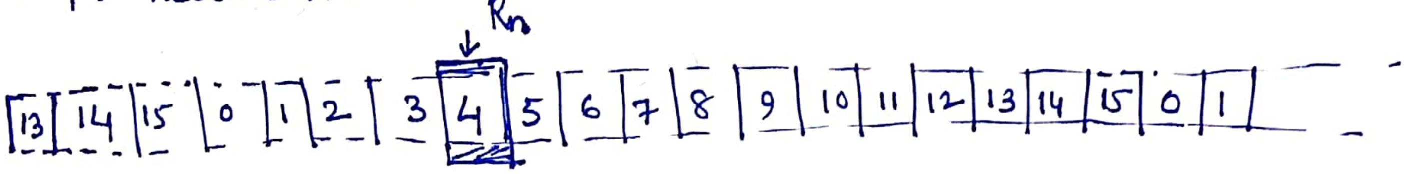


Fig: Window after sliding

The size of the receive window is always 1. Receive always looks for arrival of specific frame. Any frame arriving out of order is discarded & needs to be resent.

Only frame with a sequence no. matching the value of Rn is accepted and acknowledged.

The receive window also slides, but only one slot at a time. When correct frame is received (and the frame is received only one at a time), the window slide.

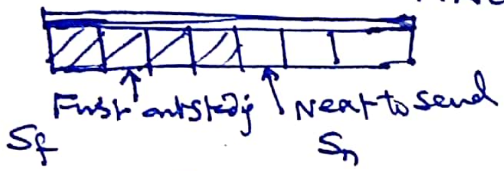
12-DLC

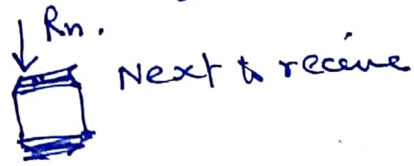
Timers

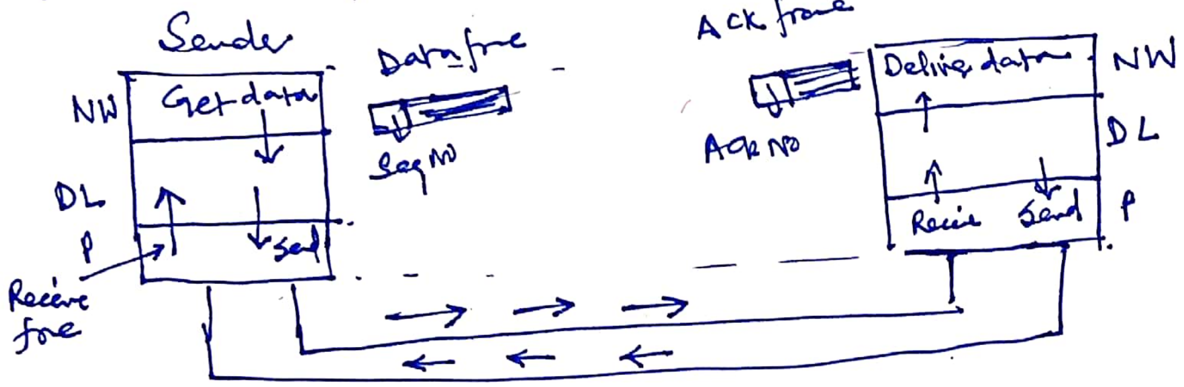
We normally use only one timer because timer for the first outstanding frame expires first; we send all outstanding frames when this timer expires

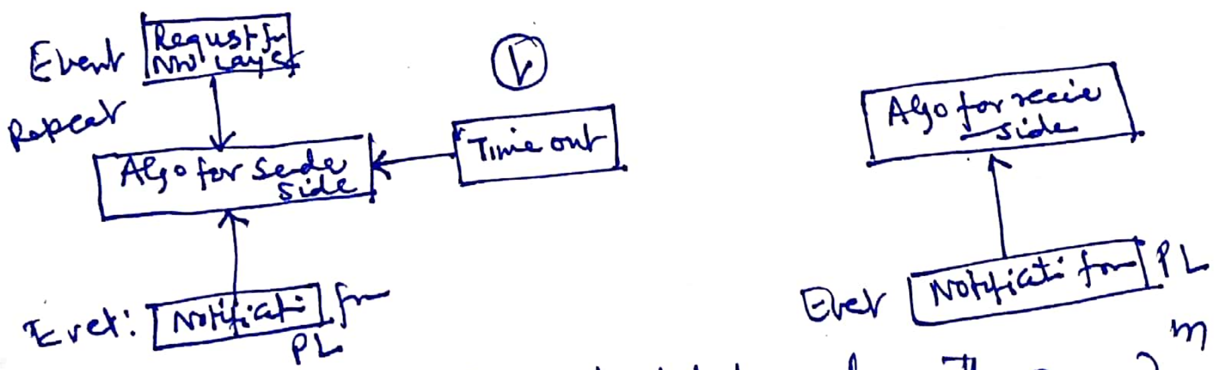
Acknowledgement: Receiver sends tive acknowledget if frame has arrived safe & in order.
Silence means: Frame is damaged or out of order all Subsequent frames are discarded unless expected
The Silence of receiver causes timer of unacknowledged frame receiv at sender side to expire. This causes Sender to Go Back and resend all frames, begining with one with expired timer.
The receiver need not send acknowledgement of each frame received. It can send cumulative acknowledget.

Resending frame: When timer expires, Sender resends all outstanding frames. For example suppose Sender has sent frame No. 6, but timer for frame 3 expires. This means frame 3 has not been acknowledged. Sender goes back and sends frames 3, 4, 5, 6 again. That is why Protocol is called Go-Back-N ARQ.









Q: Why Size of window should be less than 2^m.