2 Data Link Layer.

- It is layer in osI layer placed between the physical layer & network layer

=

Network
Data link
Physical

- It is interface between network & physical layer.
- It transform raw bit coming from physical layer to a layer it means hop to hop.
- Responsibility of data link layer is framing, addressing, flow control, error control.
- The main responsibility of data link layer is to form a frame.
- Data link layer divides a stream of bits coming from the network layer into manageable data unit called as frame

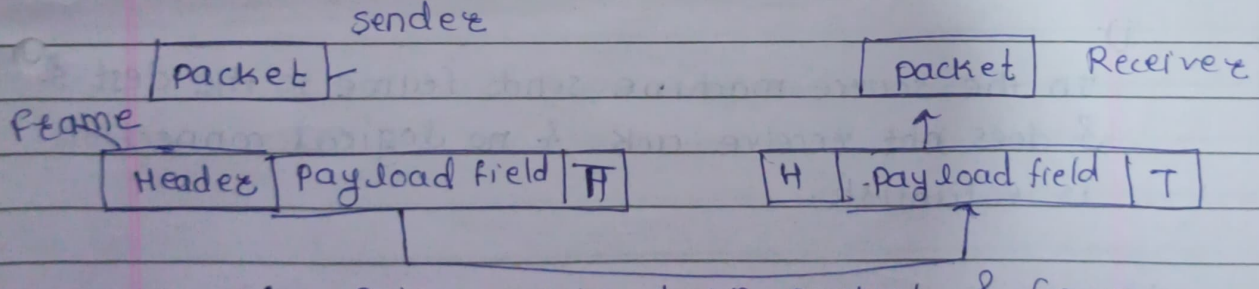


fig. Relationship betn packet & frame

- Header is used to ~~source~~ store info. for source & destination & Trailer is used for error correction

corrupted

Function of DLL

i) service to nlw layer
ii) Deals with transmission error
iii) flow control.

Design issues

1) Services provided to nlw layer.
2) Framing
3) Error control
4) flow control

1) Services provided to nlw layer

- The main responsibility is to transform data from sender to receiver
- 3 possibilities

i) unacknowldge connectionless service
ii) Acknowldged connectionless service
iii) Acknowldged connection oriented service

i)

In the source machine sends frame to the dest & & does not receive ack & no logical connection is establish.

ii)

whatever frame we sending, get the ack from recerver, means we knows the data is received, or not by ack.

If any kind of ack not receive in a particular time interval then the sender resend the data

iii)

1st we building the connection between two devices & then transfering data.

whatever packet send to the receiver, then destn send ack to the sender.

Framing

- ~~Del~~ Breaking the bit stream into framing is called as framing.
- Dividing data into manageable data.

i)

ii)

iii)

iv.)

character count method

PAGE :
DATE : / /

Data can be corrupted during transmission, some appn require that errors be detected & corrected.

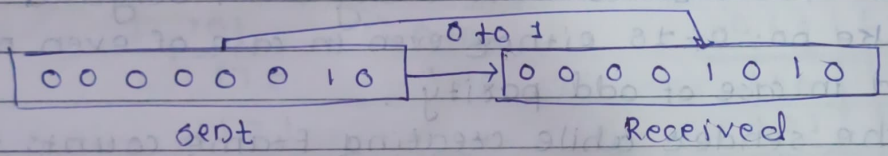
1) Error Detection & correction
   a) single bit error
   b) Burst error
   c) Redundancy

a) single bit error

i) The term single bit error means that only 1 bit of given data unit (such as byte, character/packet) is changed from 1 to 0 & from 0 to 1

ii) In single bit error only 1 bit in the data unit has changed

e.g → 10.1,00000010 (ASCII STX) was sent but 00001010 (ASCII LF) was received



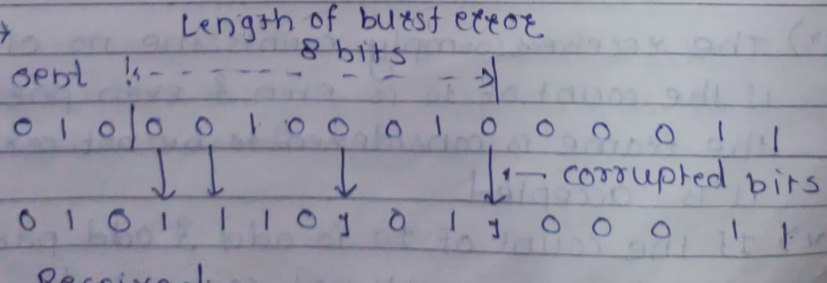
b) Burst error

i) The term burst error means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1

ii) 0100010001000011 was sent but 0101110101100011 was received.

iii) The length of the burst is measured from first corrupted bit to the last corrupted bits

e.g →



iv) Frame contains more than 1 consecutive bits corrupted

3) Redundancy

i) To detect or correct errors we need to send some extra bits with our data

ii) The redundant bits are added by sender & removed by the receiver.

2) Error Detection

i) Errors in the received frames are detected by means of parity check & cyclic Redundancy check (CRC).

ii) In both cases few extra bits are sent along with actual data to confirm that bits received at other end are same as they were sent.

iii) If the counter check at receiver & end fails the bits are considered corrupted.

Parity check

i) One extra bit is sent along with original bits to make no. of 1s either even in case of even parity or odd in case of odd parity.

ii) The sender while creating frame counts the no. of 1s in it.

iii) e.g if even parity is used & no. of 1s is even then one bit with value 0 is added. this way no. of 1s remains even, if no. of 1s odd to make it even bit with value 1 is added.

Data Bits → Even parity

1 0 0 1 0 0 1 ⟶ 1 0 0 1 0 0 1 [1]

iv) The receiver simply counts the no. of 1s in a frame if the count of 1s is even & even parity is used the frame is considered to be not corrupted & it is accepted

v) If the count of 1s is odd & odd parity is used the frame is still not corrupted

CRC

i) CRC is different approach to detect if the received frame contains valid data

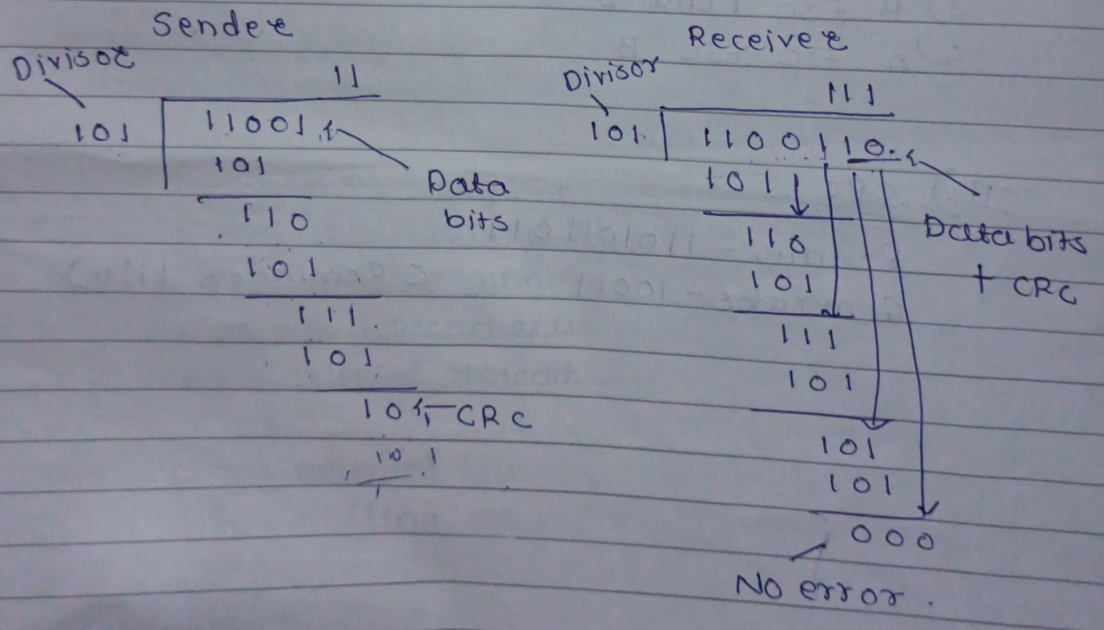
ii) This technique involves binary division of the data bit being sent.

iii) The divisor is generated using polynomials.

iv) The sender performs division operation on the bits being sent & calculates the remainder.

v) Before sending actual bits, the sender adds the remainder at the end of actual bits.

vi) Actual data bits plus the remainder is called codeword. The sender transmits data bits as codeword.



- At other end receiver performs division operation on codewords using same CRC divisor
- If the remainder contains all zeros the data bits are accepted, otherwise it is considered as there some data corruption occured in transit

Error correction

1) Backward error correction

when receiver detects an error in the data received it requests back the sender to retransmit data. an

2) Forward error correction

when the receiver detects some error in data received, it executes error correcting code, which helps it to auto recover & correct some kinds of errors

Example (Byte stuffing)

1) A flag B →

2) A ESC B

3) A ESC FLAG B

4) A ESC ESC B

II) CRC

Frame - 11010110 11

Generator - 10011 (Reminder 1110)

PAGE :
DATE : / /

Framing

i) Framing in data link layer separates a message from one source to a destination, or from other messages to other destination, by adding a sender address & destination address.

ii) The dest. address defines where the packet is to go, the sender address help the recipient ack the receipt.

iii) In data link layer, needs to pack bits into frames, so that each frame is distinguishable from another

i) Fixed size framing

a) Frames can be of fixed or variable size

b) In fixed size framing, there is ~~no~~ need to defining area networks the boundaries of the frames, the size itself can be used as delimiter

c) Example of this type of framing is the ATM wide area network, which frame uses fixed size called cells.

ii) Variable size framing

a) In variable size framing, we no need a way to define the end of the frame & beginning of the next

b) In that two approaches were used for this purpose (a) a character oriented approach & (b) bit oriented approach.

a) character oriented framing / byte stuffing

i) In byte stuffing or character stuffing a special byte is added to the data section of frame when there is a character with same pattern as flag.

ii) The data section is stuffed with an extra byte.

iii) This byte is called the ESC means escape character which has predefined bit pattern.

iv) whenever receiver encounters ESC character it removes it from data section & treats the next character as data not delimiting flag.

v) Byte stuffing by esc character allows presence of flag in the data section of the frame, but it creates another problem.

vi) If the ESC is part of the text an extra one is added to show that the second one is part of the text.

- Data from upper layer

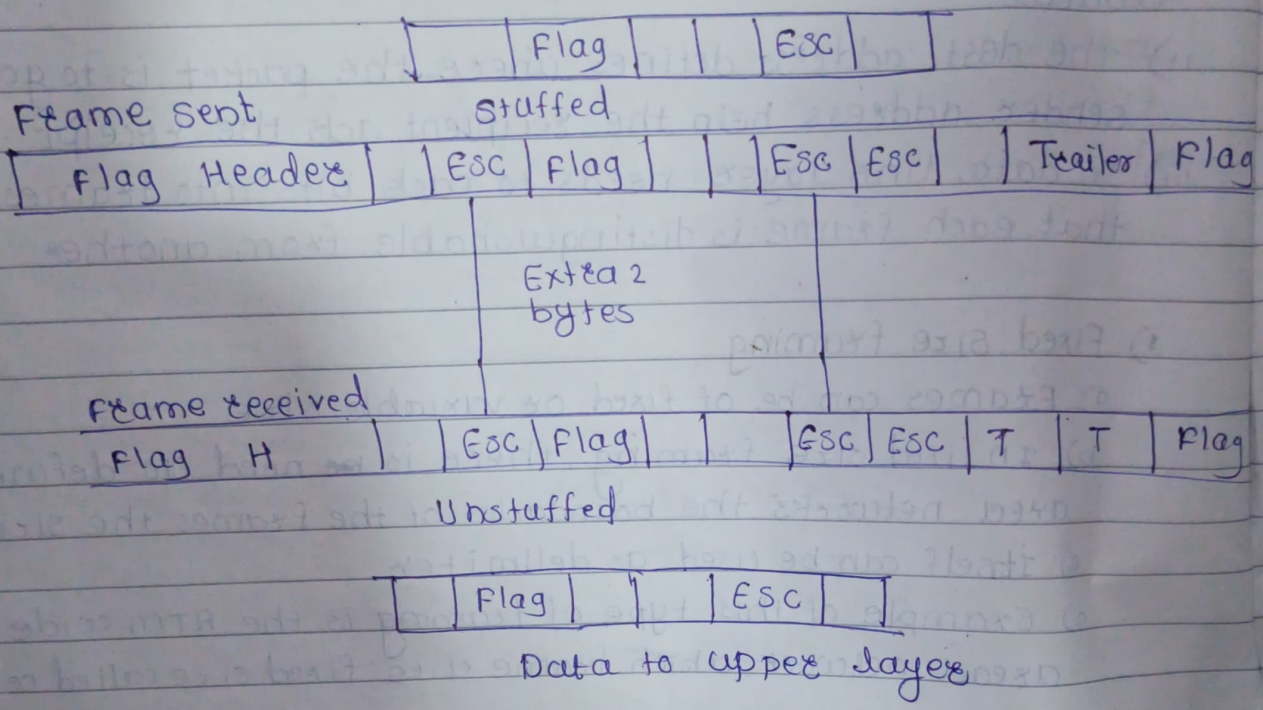
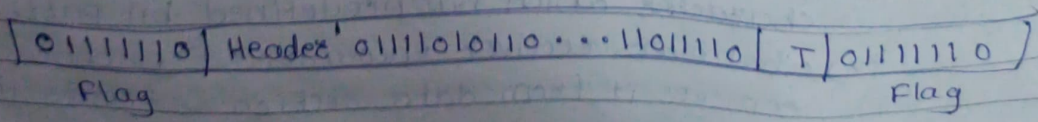


fig → byte stuffing & unstuffing

vi) Byte stuffing is the process of adding 1 extra byte whenever there is flag / escape character in the text.

② Bit stuffing

- Most protocol use a special 8 bit pattern flag 01111110 as the delimiter to define the beginning & the end of the frame.



1) If the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame.

iii) we do this by stuffing 1 single bit to prevent the pattern from looking like flag this is called bit stuffing.

iv) In bit stuffing if a 0 & 5 consecutive 1 bits are encountered an extra 0 is added.

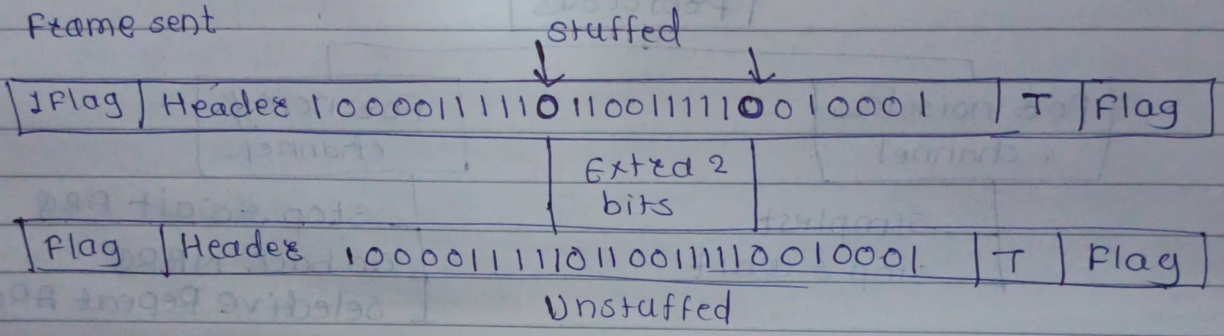
v) This extra stuffed bit removed from the data by receiver.

vi) Bit stuffing is the process of adding one extra 0 whenever 5 consecutive 1 follow a 0 in the data, so that receiver does not mistake the pattern 01111110 for a flag.

Data from upper layer

1000011111100111110010001

Frame sent



1000011111100111110010001

Data to upper layer

fig → Bit stuffing & unstuffing

Flow & error control

The most important responsibilities of data link layer are flow control & error control. these fun: are known as data link protocol

Flow control

i) Flow control co-ordinates the amount of data that can be sent before receiving an acknowledgement & is one of the most important duties of data link layer.

ii) Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for ack.

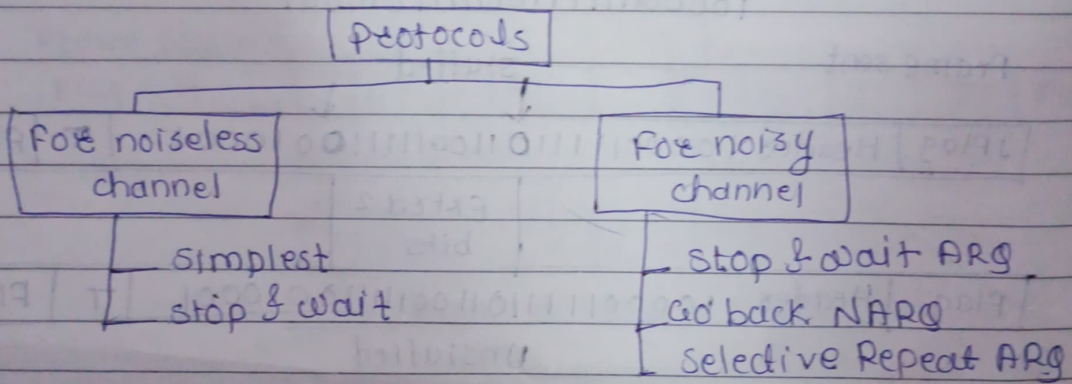
Error control

i) It is both error detection & error correction.

ii) It allows the receiver to inform the sender of any frames lost or damaged in transmission & co-ordinate the retransmission of those frames by the sender.

iii) Any time error is detected in an exchange, specified frames are retransmitted this process is called ARQ (Automatic Repeat Request)

Protocols.



- The noisless channel is also called as error free & noisy channel is called the error creating.

① Noisless channels.

a) simplest protocol

i) It is a unidirectional protocol in which data frames are travelling in only one direction from the sender to receiver.

ii) The data link layer at the sender site gets data from it's network layer, makes frame out of data & sends it.

iii) The data link layer at the receiver site receives a frame from it's physical layer, extract data from the frame & delivers the data to it's network layer.

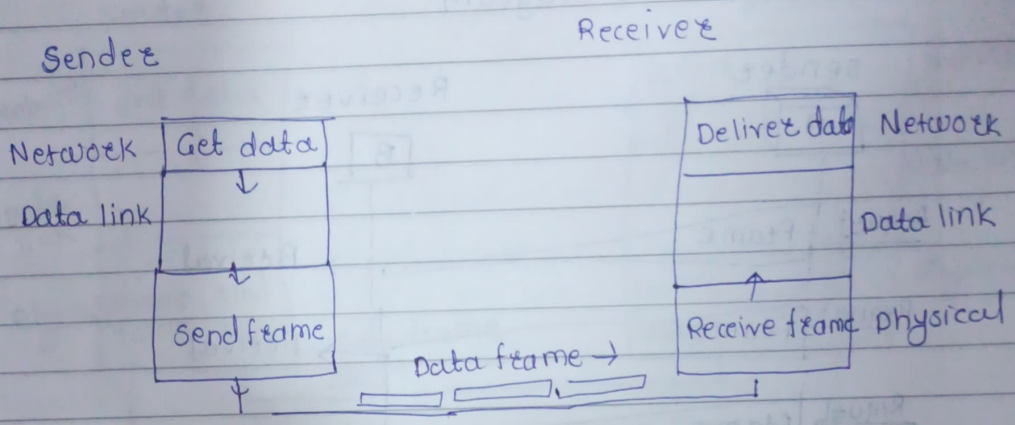


fig → The design of the simplest protocol with no flow or error control.

Algorithm

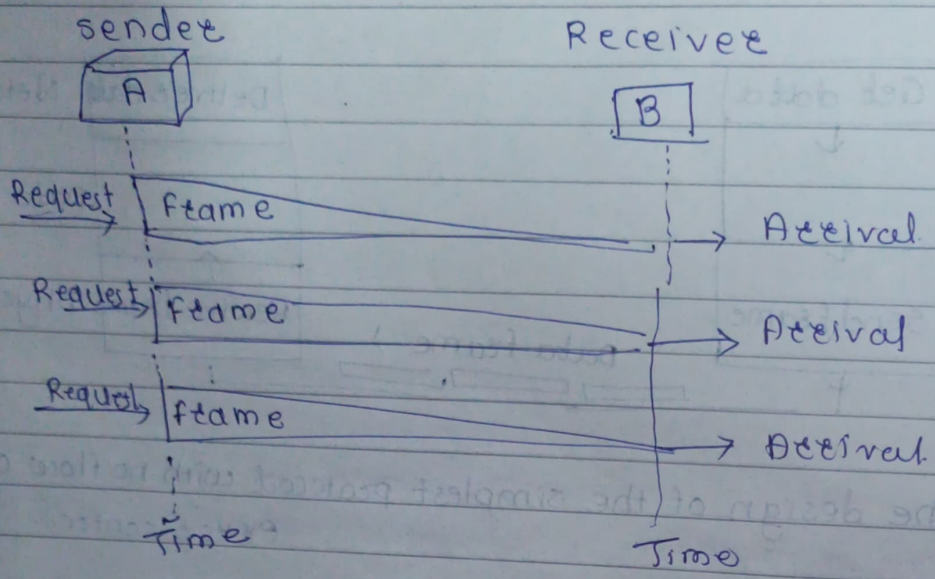
* Sender site

```
while (true)                    // Repeat forever.
{
    WaitForEvent();             // sleep until an event occurs
    if (Event(RequestToSend))   // There is packet to send
    {
        GetData();
        MakeFrame();
        SendFrame();            // send frame
    }
}
```

* Receiver site

```
while (true)                    // Repeat forever
{
    waitForEvent();             // sleep until event occurs
    if (Event(ArrivalNotification))   // Data frame arrived
    {
        Receive frame();
        Extract Data();
        Deliver Data();         // Deliver data to Network layer.
    }
}
```

Example → Flow diagram



i) The sender sends a sequence of frames without even thinking about the receiver.
ii) To send three frames, three events occur at the sender site & three events at the receiver site
iii)

Stop & wait

i) The sender sends one frame, stop until it receives confirmation from the receiver & then sends the next frame.
ii) We still have unidirectional communication for data frames, but auxiliary Ack frames (simple tokens of acknowledgment) travel from the other directions.
iii) In below figure we can see the traffic on the forward channel & reverse channel. At any time, there is either one data frame on the forward channel or one Ack fram on the reverse channel. We therefore need a half duplex link

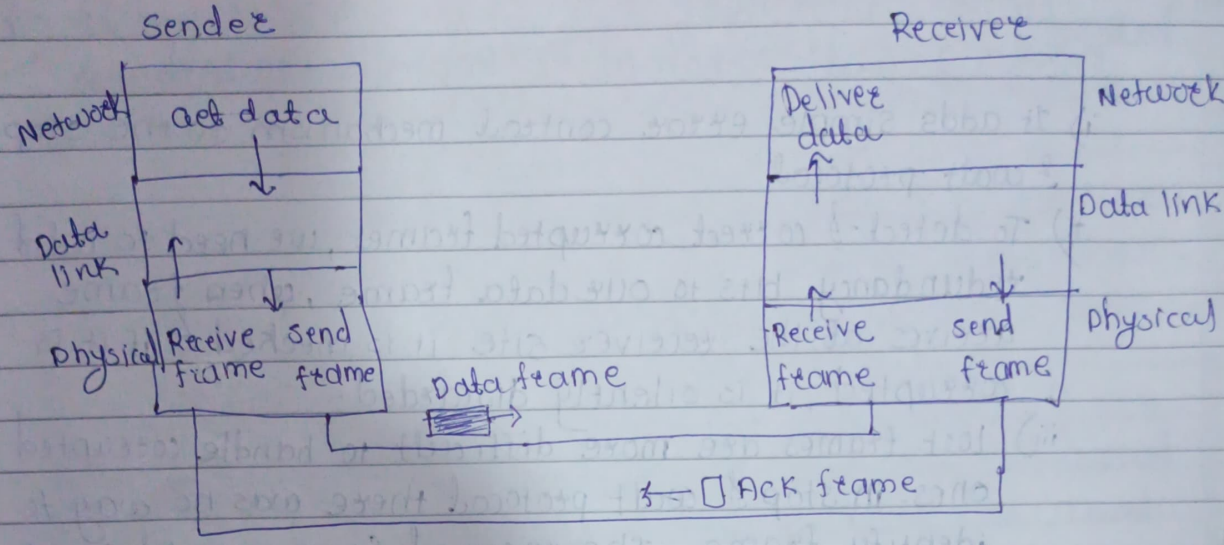
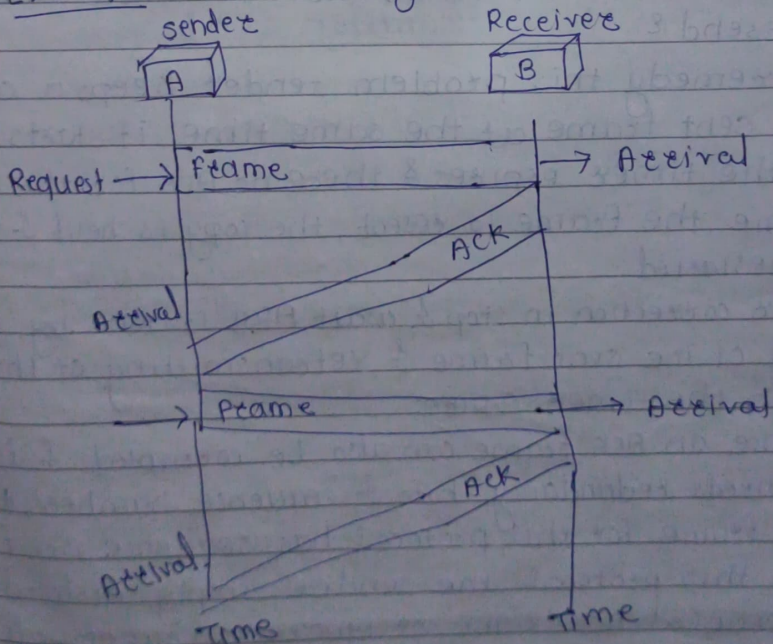


fig → Design of stop & wait protocol

3 problems occured during stop & wait protocol

1) Due to lost data
2) Due to lost Ack
3) Delayed ack

Example → Flow diagram



- The sender sends one frame & waits for feedback from the receiver, when Ack arrives, the sender sends the next frame.
- Sending two frames in the protocol involves the sender in four events & receiver in two events

NOISY CHANNEL

1) Stop & wait Automatic Repeat Request

i) It adds simple error control mechanism to the stop & wait protocol.

ii) To detect & correct corrupted frames, we need to add redundancy bits to our data frame, when frame arrives at the receiver site, it is checked & if it is corrupted, it is silently discarded.

iii) Lost frames are more difficult to handle corrupted ones. in stop & wait protocol there was no way to identify frame, The received frame could be the correct one or a duplicate, or a frame out of order, the solution is to number the frames, when the receiver receives data frame that is ou of order, this means that frames were either lost or duplicated.

iv) The completed & lost frames need to be resent in this protocol, if the receiver does not respond when there is an error, how can the sender know which fra to resend?

v) To remedy this problem sender keeps a copy of the sent frame, at the same time, it starts timer If the timer expires & there no Ack for the sent frame, the frame is resent, the copy is held & timer is restarted

vi) Error correction in stop & wait ARQ is done by keeping copy of the sent frame & retransmitting of the fram when the timer expires.

vii) Since an Ack frame can also be corrupted & lost it needs redundancy bits & sequence number, the Acks frame for this protocol has sequence no. field In this protocol, the sender simply discard a corrupted ACK frame or ignores an out of order one

Sequence No.

i) The protocol specifies that frames need to be numbered. This is done by using sequence numbers A field is added to the data frame to hold the sequence no. of that frame

ii) If we decide that the field is m bits long, the sequence no. start from 0, go to $2^m - 1$ & then repeated

iii) Assume that sender has sent frame numbered x three things can happen

a) The frame arrives safe & sound at the receiver site, the receiver sends an ack. The ack. arrives at the sender site, causing the sender to send next frame numbered x+1

b) The frame arrives safe & sound at the receiver site, receiver sends an ack, but ack corrupted The sender resends frame after time out. That the frame here is duplicate, The receiver can recognize this fact because it expects fram x+1 but frame x was received.

c) The frame is corrupted or never arrives at the receiver site, the sender resends the frame after the timeout.

iv) In stop & wait ARQ we use sequence no. to no. the frames, the seq. no. are based on modul 0-2 arithmetic

Acknowldgement No.

i) The acknowldgement no. always announce the sequence no. of the next frame expected by the receiver. e.g → if frame 0 has arrived safe & sound, receiver sends an Ack frame with ack. 1 means frame 1 is expected next

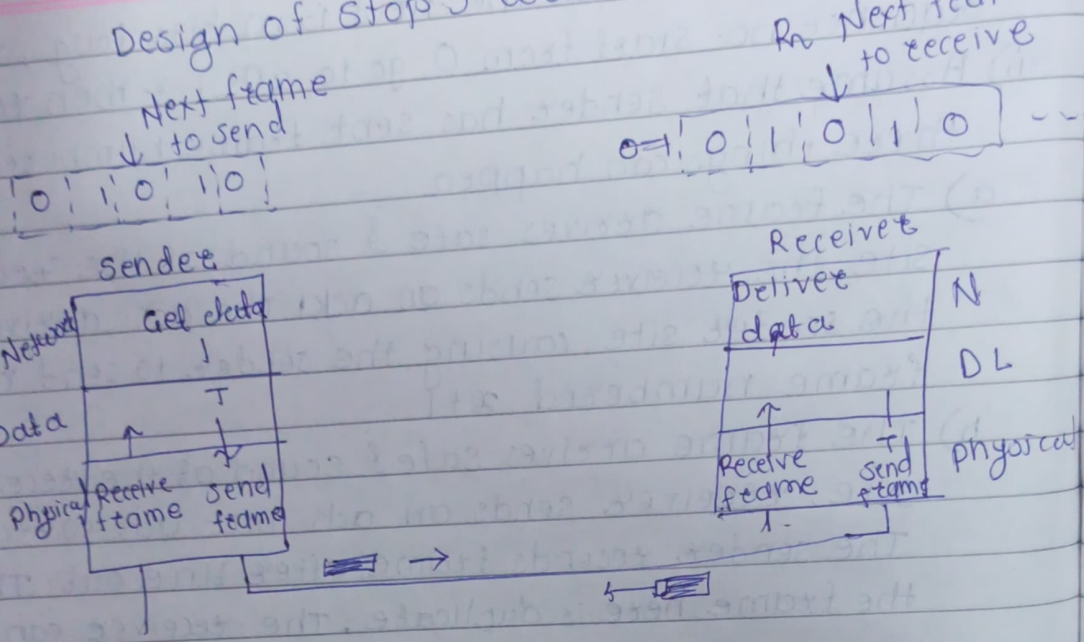
ii) If the frame 1 has arrived safe & sound, receiver sends an Ack frame with ack 0

iii) In that ack no. always announces in modul 0-2 arithmetic the sequence no. of next frame expected.

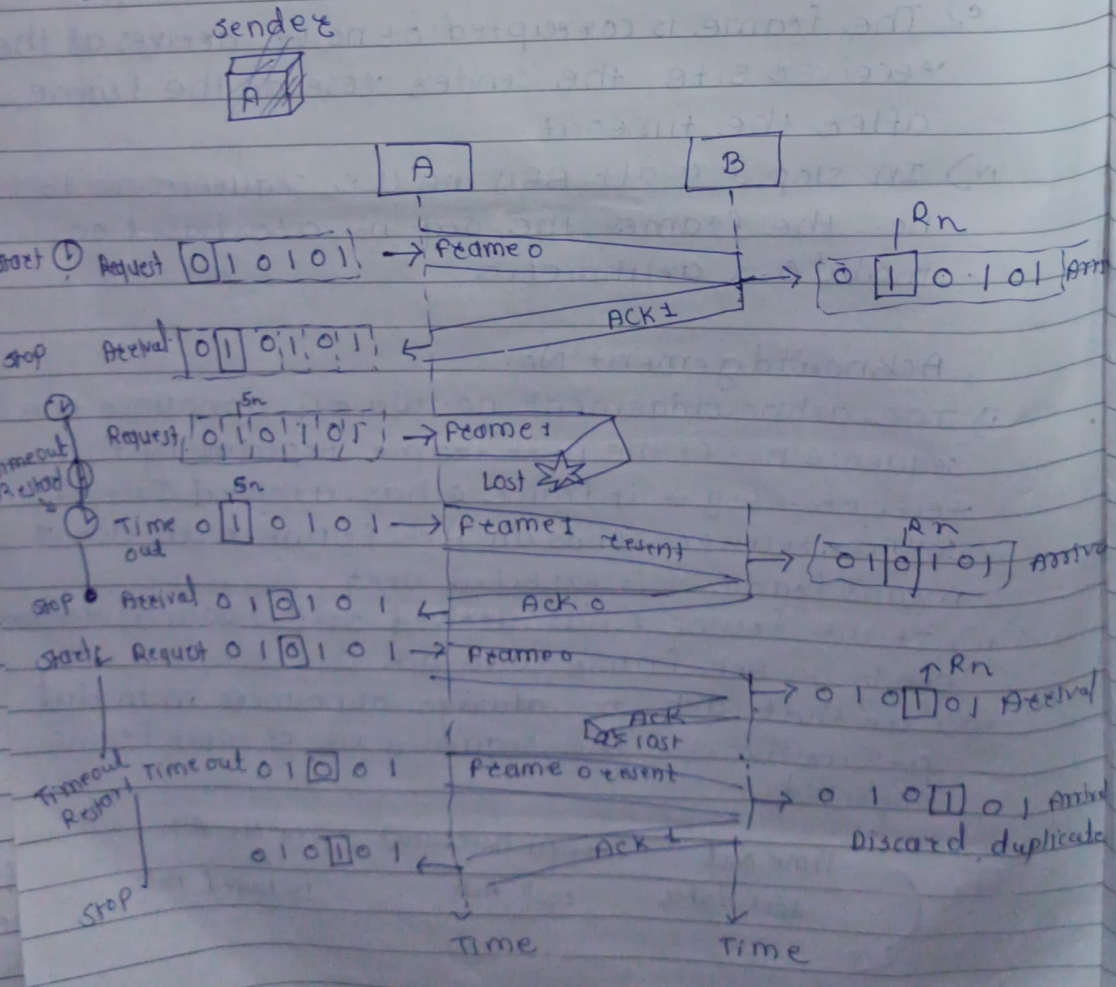
Time out — Sequ no.(Data) — Seq. No. Ack

lost data — Lost Ack — Delayed Ack

Design of stop & wait ARQ protocol



Example.



DATE

PAGE:
DATE: / /

* Go Back N Automatic Repeat Request

i) In this protocol we can send several frames before receiving ack, we keep a copy of these frames until the ack arrive.

Sequence Number

i) We need to include sequence number of each frame in the header, we need to set limit.

ii) If the header of the frame allows m bits for the sequence no., the sequence no. range from 0 to $2^m - 1$
e.g → m = 4 then seq. no = 0 to 15

iii) In the Go-Back N protocol, the sequence no. are modulo 2^m, where m is the size of sequence no. field in bits.

Sliding window

i) The sender & receiver need to deal with only part of the possible sequence numbers. The range which is concern of the sender is called the send sliding window, the range that is concern of the receiver is called receiver sliding window.

ii) The send window is an imaginary box covering the sequence number of the data frames which can be in transit. In each window position some of these sequence no. of the data define the data frame that have been sent, others define those that can be sent. The maximum size of window is $2^m - 1$.

- In fig. range of sequence no. belonging to the frames that are sent & have unknown status. The sender needs to wait to find out if these frames have been received or were lost, that frame is called outstanding frames.
The 3rd range white in fig. defines the range of sequence no for frames that can be sent however corresponding data packets have not yet been received from the network layer.

& Finally fourth region defines sequence no. that can be used until the window slides as we see next.

Three variables define it's size & location at any time we call these

Sf → send window, 1st outstanding frame

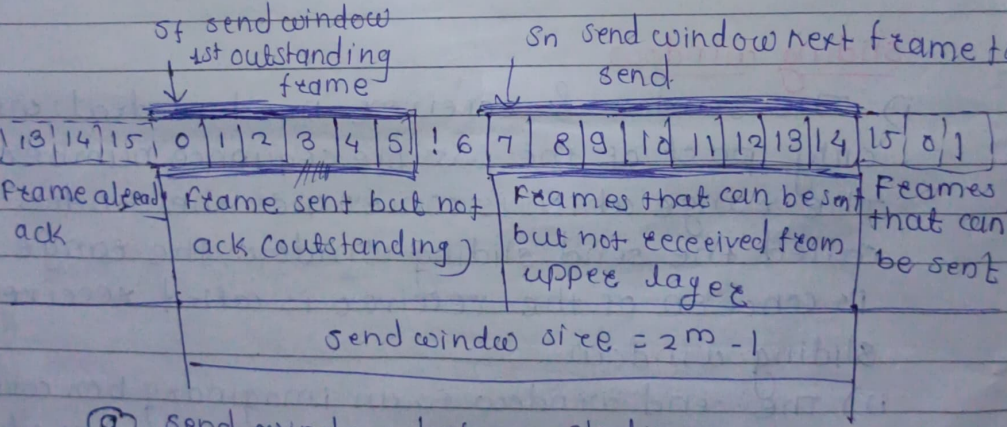
Sn - send window, the next frame to be sent

& Ssize - send window, size

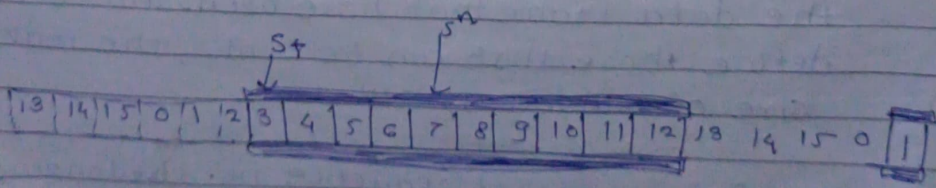
i) The variables Sf defines the sequence no. of the 1st outstanding frame

ii) The variable Sn holds the sequence no. that will be assigned to the next frame to be sent

iii) Finally the variable Ssize defines size of window which is fixed in our protocol.



(a) send window before sliding



(b) send window after sliding

In fig (b) frame 0, 1, 2 are ack, so the window has slid to the right 3 slots. Note that value of Sf is 3 because frame 3 is now 1st outstanding frame.

Receive window for Go-back NARQ

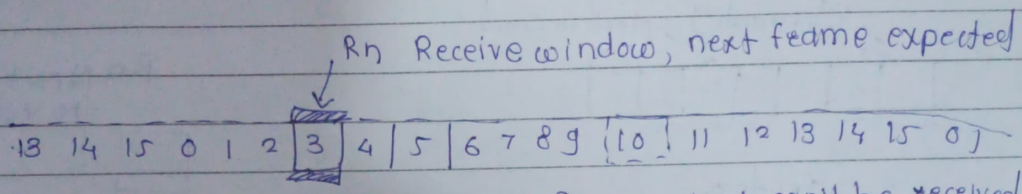


Fig. (a) Receive window

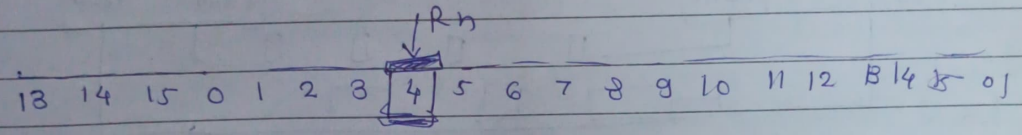


fig (b) window after sliding

The receive window is an abstract concept defining an imaginary box of size 1 with one single variable Rn the window slides when correct frame has received, Sliding occurs one slot at a time.

Timers

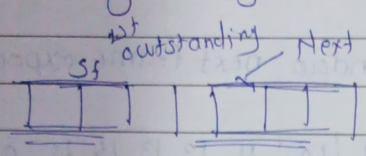
In this protocol we use only one timer, the reason is that timer for first outstanding frame always expires yet we sent all outstanding frames when this timer expires.

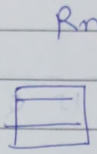
Acknowledgment

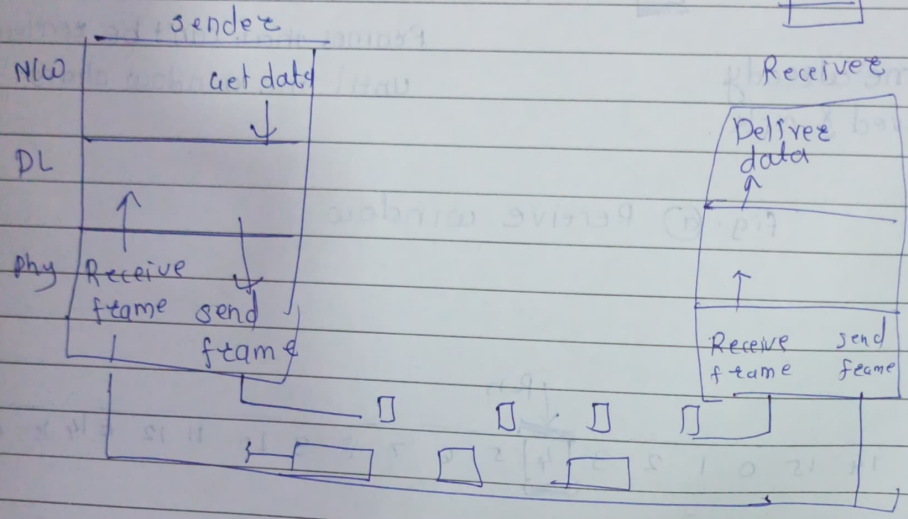
- Receiver sends +ve ack if a frame has arrived safe & sound & in order. If frame damaged, the receiver is silent & will discard all subsequent frames until it receives one it is expecting.

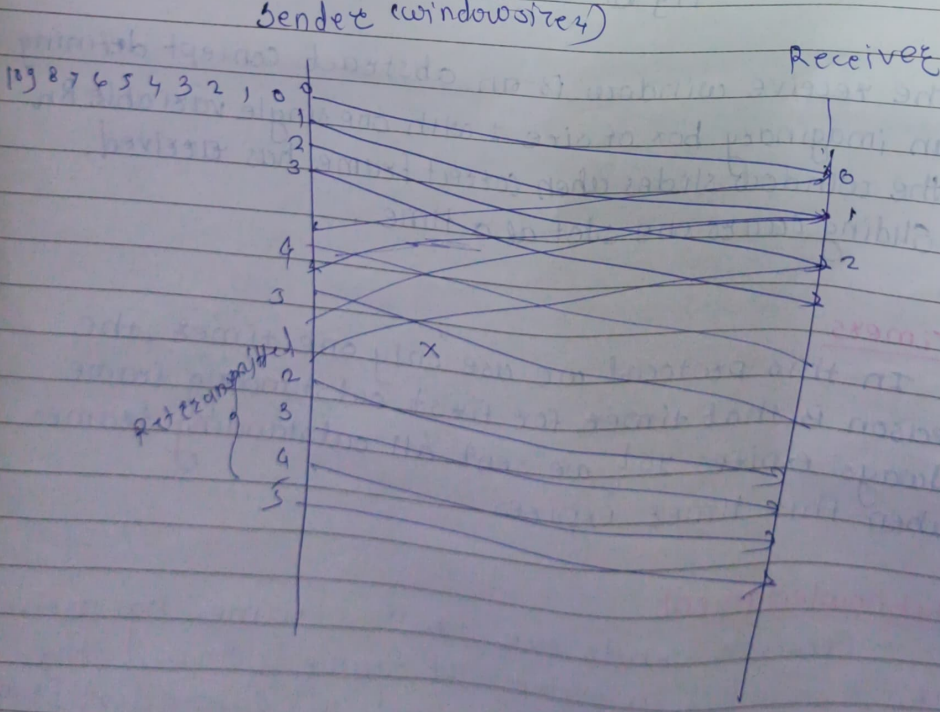
If the ack of frame is not received within an agreed upon period, all frames in the ~~current~~ window are transmitted

Design of go back N ARQ









ii) In Go back N ARQ if one frame is discarded then we need to resend ~~all~~ windows so ~~all~~ frames are repeated.

iii) There is another mechanism that does not resend N frames when just one frame is damaged only, the damaged frame is resent this mechanism is called as selective Repeat ARQ