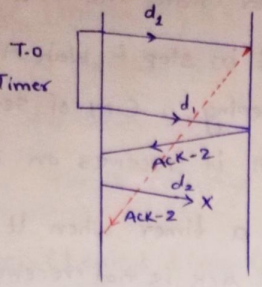
3.) Delay Ack :



→ We solve this problem by assigning Next Expected frame Number With Ack.

$$\eta = \frac{T_T}{T_T + 2T_p}$$

→ Total Waiting time = Stop & Wait Time + Sender time out + Receiver time out

$$\eta = \frac{1}{1+2a}$$

Throughput = $\eta \times BW$

• Go Back - N ARQ :-

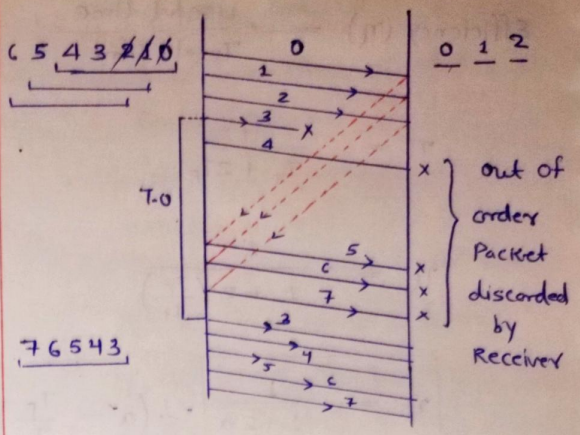
→ In the sliding Window Concept instead of sending one packet and wait for the ack, We sent W packet and. Wait for the Ack.

[W : Sender Window Size)

→ GB-N (N>1)

1) Sender Window size = N

2) Receiver Window size = 1



Note :-

1) out of order packet is not Received by the receivers.

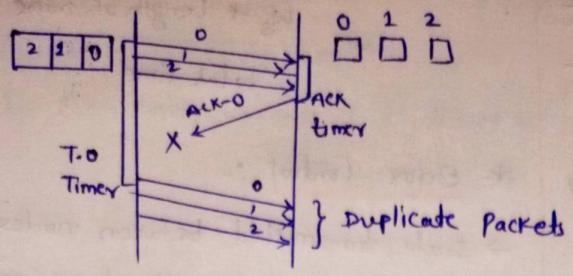
2) Timer is maintained only for the first frame (Right most) in the window because If its timer expire then sender assume that rest of the frames are Not received by receiver (Because out of order delivery is rejected)

Note :-

1) G.B.N uses Commulative Ack and Ack Number defines the Number of next expected frame.

ACK timer < T.O. Timer

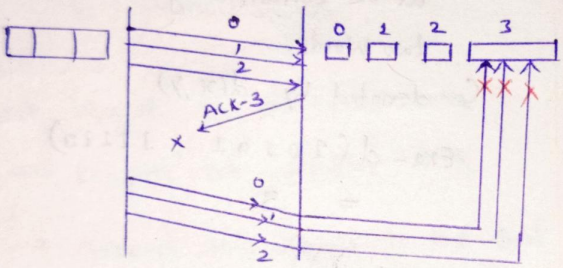
• Relationship b/w Window size & seq. No.



→ Duplicate packet problem can be solved by increasing the seq. No. or Applying this formula–

$$W_S + W_R \leq ASN$$

↓ Available Sequence Number



Discard duplicate Packet

Waiting for Packet-3

Using formula– $W_S + W_R \leq ASN$

$3 + 1 \leq 4$

$4 \leq 4$ ✓

$\Rightarrow W_S \leq ASN - 1$

→ if seq No. = K bit

Total seq No. = 2^K

W_S	W_R
$2^K - 1$	1

$$\Rightarrow \eta = \frac{\text{useful time}}{\text{total time}}$$

$$= \frac{N * T_{d(frame)}}{T_{d(frame)} + 2 \times P_d + Q_d + P_{rd} + T_{d(ACK)}}$$

$$= \frac{N}{1+2a}, \quad a = \frac{P_d}{T_d}$$

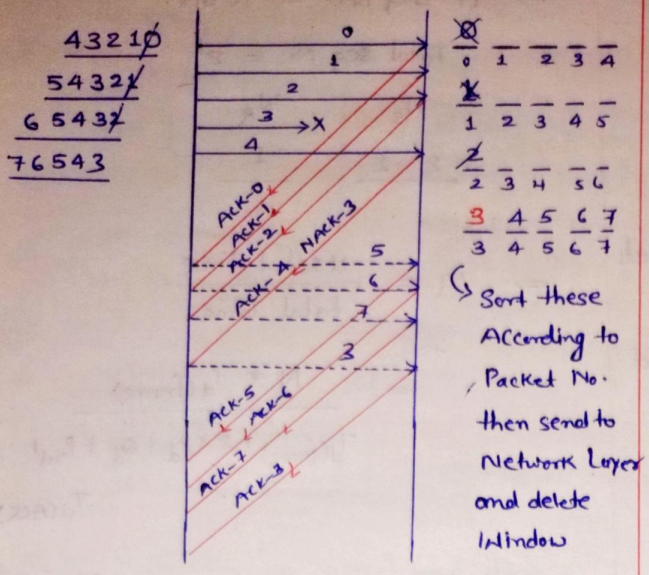
$\Rightarrow$ Throughput $= \eta \times BW$

OR

$$\text{Throughput} = \frac{N * \text{Length of frame}}{\text{Total Time}}$$

* Selective Repeat ARQ :-

- In SR – $W_R = W_S$
- SR Protocol uses Independent ACK. and ACK Number defines Number of Error free packet Received.
- It Can Receive out of order packet but packets are delivered to upper layer in order.
- Searching and Sorting Algorithm required. Searching is done by sender. Sorting is done by receiver.
- Timer is maintained for Each and Every frame in the Window at sender side.



↳ Sort these According to Packet No. then send to Network Layer and delete Window

→ For 1st out of order delivery or if Packet received is Corrupted then NACK for respective Packet is sent by receiver to Sender.

→ Relationship b/w window size and sequence No.

- Duplicate Packet problem Can be solved by Increasing the sequence No. or Decreasing the sender window size.

- $W_S + W_R \leq A.S.N$

- if seq No. – K bit

$$\frac{2^{K-1}}{W_S} \qquad \frac{2^{K-1}}{W_R}$$

$$\eta = \frac{W_S * T_d(\text{frame})}{T_{d(\text{frame})} + 2P_d + Q_d + Pr_d + T_d(ACK)}$$

Throughput $= \eta \times BW$

$$= \frac{W_S * \text{Length of frame}}{\text{total time}}$$

* Error-Control :-

→ Data transmitted between nodes Can sometimes be altered or Corrupted during transmission.

→ This Corruption Can manifest as either a single-bit error or burst error.
↳ (multiple bit change)

- ~~Hamm~~ Hamming distance :-
  ↳ Total Number of dissimilar bits at the Corresponding Positions in two Words.
  ↳ denoted by $d(x,y)$
  Exa- $d(10101, 11110)$
  $= 3$

- Error detection
  - → Simple Parity check
  - → Cyclic Redundancy check
  - → checkSum

- Error Correction
  - ↳ Hamming Code

- Simple Parity check / single Parity check

DataWord → m bits

Code Word → (m+1) bits

for Even Parity → No. of 1's should be Even.

Exa-

1010 ⟶ 10100

1110 ⟶ 11101

- Cyclic Redundancy check (CRC)

→ If message has m bits then transmitter sends (m+r) bits, r ~~repers~~ represents redundant bits.

→ This Can detect all odd Errors, single bit, burst Error of length Equal to polynomial degree

→ Length of dataword = n

→ Length of divisor = K

→ Append K-1 Zero's to the End of the msg.

→ Perform modulo 2 division

→ Remainder of division = CRC

→ Code Word = (n+K-1) bits

CodeWord = dataWord with Appended Zero + CRC

Exa- Data = 1001001

Divisor = 1101

```
1101 ) 1001001000
       1101
       0100001000
        1101
        010101000
         1101
         01111000
          1101
          0010000
            1101
            01010
             1101
             0111
               ↳ CRC Remainder
```

CodeWord = 1001001111

- Algebric Properties to choose the Polynomial

1) It should not be divisible by x.
   ↳ This Cond guarantee that all the burst Error upto the length equal to the length of polynomial are detected.

2) It should be divisible by $(x+1)$.
   ↳ To detect odd No. of Error

Note :-

1) CRC Can detect all Single bit Error.
2) CRC Can detect all double bit Error if polynomial does not divided by x^K+1 for any K.
3) CRC Can detect any odd No. Error if polynomial should be divisible by $x+1$.

4) CRC can detect all burst Errors if Polynomial G(x) has degree n and is not divisible by x.
detects all burst Errors of length $\leq n$

5) CRC can detect most of large bit Errors with high Probability.

✱ CheckSum :-

- It works by adding up all data units (Packet / Words), taking the Complement, and appending it as the checksum at the Sender side.
- The receiver performs the same addition, and if the result is all 1's (no Error), Else an Error is detected.

Steps :-

1) Sender Side :-
   - 1) m bit checksum
   2) n bits in transmitting message
   3) n bits are divided into fixed size segment equal to m bit
   4) Sum to all m bit segments
   5) 1's Complement of final output
   6) output is called checksum

2) Receiver side -
   1) Transmitted bits + checksum
      ↳ sum of m bit segments
   2) 1's Complement
      if Result = 0 No Err
      Result ≠ 0 Err

Exa-

Sender Wants to transmit -

msg- 10011001 11100010 00100100
10001100

1) Sender -

Add them all -

```
  10011001
  11100010
 ----------
 101111011
      + 1
 ----------
  01111100
+ 00100100   ← Next word
 ----------
  10100000
  10000100   ← Next word
 ----------
 100100100
      + 1
 ----------
  00100101
```

1's ↳ 11011010 → check sum

2) Receiver side

```
  00100101
  11011010
 ----------
  11111111
```

1's ↳ 00000000 ✓ (No Err)