Selective Repeat Algorithm:

→ Due to one packet loss, all following packets will need to be retransmitted, even if they have arrived at the destination. This leads to a great waste of bandwidth.

Concepts / process involved:

→ Similar to the window at the sender's side, a window is maintained at the receiver side. The purpose of window at receiver side is buffering of packets. (incoming frames)

⇒ Negative acknowledgement number for frame N will always be N. Used in case of data loss / delay.

⇒ positive acknowledgement number will indicate the sequence of next incoming frame.

→ For every frame sent, individuals timers are maintained.

N frames = N timers.

NOTE:

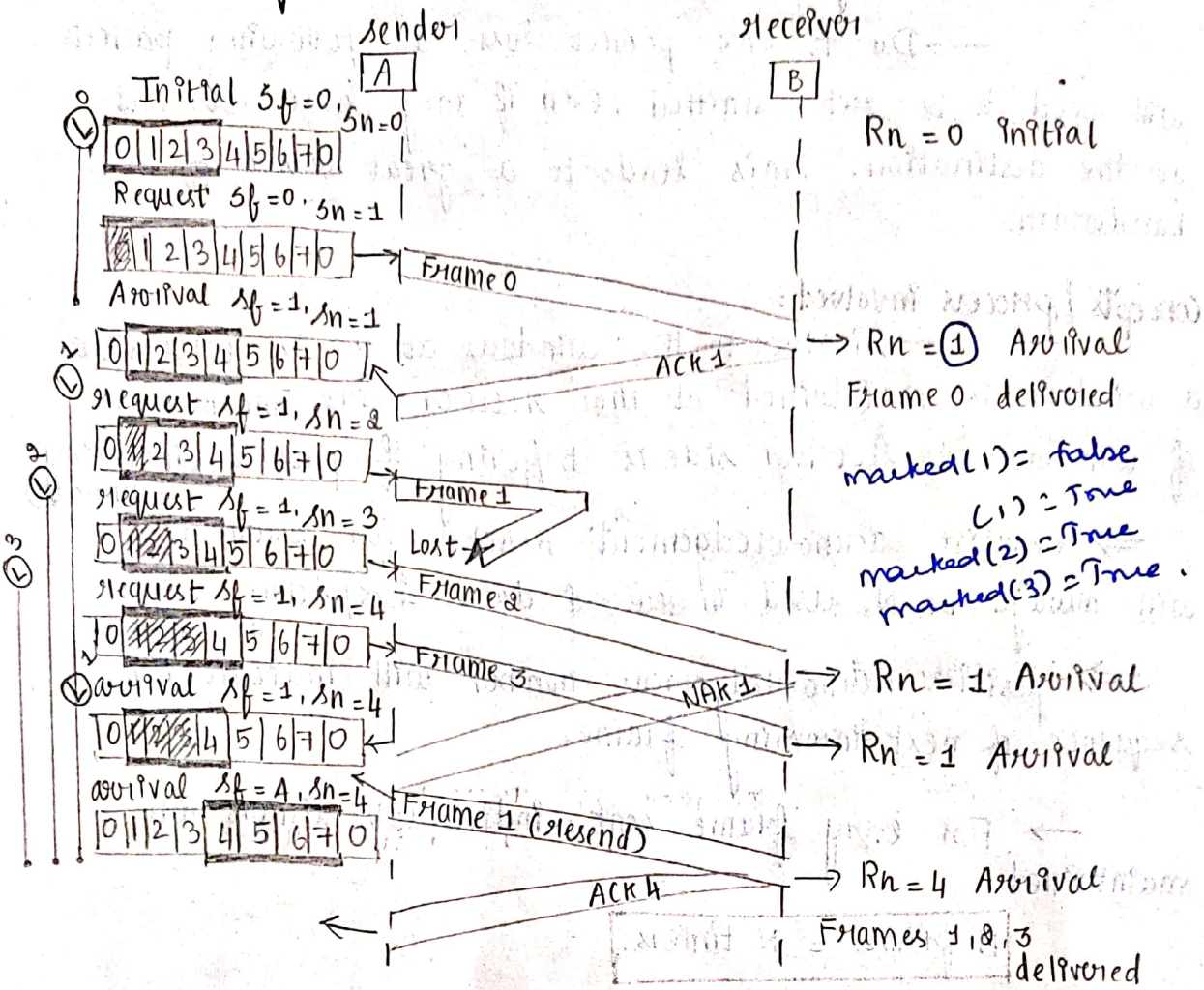
In selective repeat ARQ, the size of the sender and receiver window must be at most one half of 2^m.

Algorithm (sender site):

```
Sw = 2^m - 1;
Sf = 0;
Sn = 0;
while (true)                          // repeat forever
{ wait for Event();
  if (Event (Request To send))        // there is a packet to send
  {
    if (Sn - Sf >= Sw)                // if window is full
        sleep();
    Get Data ();
    MakeFrame (Sn);
    StoreFrame (Sn);
    Send Frame (Sn);
    Sn = Sn + 1;
    Start Timer (Sn);
```

.... continued

Flow diagram:



.... continuation:

```
if (Event (Arrival Notification)) // Ack arrives
{
    Receive (frame);                    // receives Ack or NAK.
    if (corrupted (frame))
        sleep();
    if (Frame Type == NAK)
        if (nakNo between sf and sn)
        {
            resend (nak No);
            startTimer (nak No);
        }
    if (Frame Type == Ack)
        if (ackNo between sf and sn)
        {
            while (sf < ack No)
            {
                purge (sf);
```

```
            stopTimer (sf);
            sf = sf + 1;
        }
    }
}

if (Event(TimeOut (t)))          // time expires
{
  startTimer (t);
  sendFrame (t);
}
}
```

Algorithm (receiver site):

```
Rn = 0;
NakSent = false;
AckNeeded = false;   (the previous frame is received, so there is no acknowledgement needed)
Repeat (for all slots)
    Marked (slot) = false;
while (true)                     // repeat forever
{
  waitForEvent();
  if (Event (Arrival Notification))     // data frame arrives
  {
    Receive (Frame);
    if (corrupted (Frame)) && (NOT NakSent)   (exclusive handling of corrupted frames)
    {
      sendNAK (Rn);
      NakSent = true;
      sleep();
    }
    if (seqNo <> Rn) && (NOT NakSent)   (not equal to)
    {
      sendNAK (Rn);
      NAKSent = true;
    }
    if (seqNo == Rn)
    {
      Extract data (Rn)
      Deliver data (Rn)
      Rn = Rn + 1
      send Ack (Rn)
    }
```

the receiver has not received the correct frame, but buffer it

↳ to indicate that the frame has been received but not yet acknowledge

```
if ((seqNo in Window) && (!Marked (seqNo))
{
    StoreFrame (seqNo)
    Marked (seqNo) = true;
    while (Marked (Rn))
    {
      DeliverData (Rn);
      Purge (Rn); (remove)
      Rn = Rn + 1;
      AckNeeded = true;
    }
    if (AckNeeded);
    {
      SendAck (Rn);
      AckNeeded = false;
      NakSent = false;
    }
  }
 }
}
```

↳ receiver will be able to buffer it

(Rn = 1)

Frame 1
Frame 2
Frame 3

(Rn = 4)

1. Using 5 bit sequence numbers, what is the maximum size of sent and received windows for the following protocol.

(i) Stop and wait
(ii) go back N
(iii) Selective repeat

(i) Stop and wait

Size of sender and receiver window = 1

(ii) go back N

Size of receiver window = 1

Size of sender window $= 2^m - 1$

$= 2^5 - 1$ $(m = 5)$

$= 32 - 1 = 31$

(iii) Selective repeat

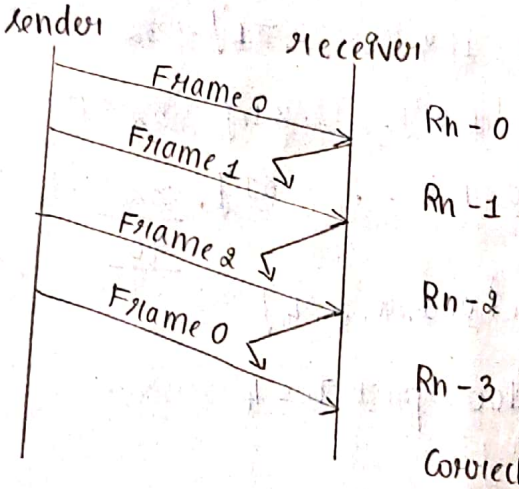
Size of sender window $= \frac{2^m}{2} = \frac{2^5}{2} = 16$

Size of receiver window $= \frac{2^m}{2} = 16$.

window size problems ⓧ ⇒ GATE

Why do we fix the size of window as 2^m in go-back N protocol?

Refer ppt Figure 11.13



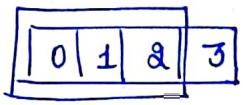
Correctly

if window size $> 2^m$ discarded

Size $< 2^m$

let $m = 2$

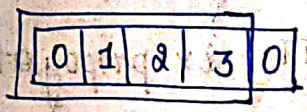
$2^m = 4$



window size = 3

Size $= 2^m$

let $m = 2$



window size = 4

Here all the frames are received, but the acknowledgements are lost. Since the ack are not received again it sends 0 but at receiver it interprets the frame next is sent and increment it.

Why do we fix size of window as $\frac{2^m}{2}$ in selective repeat protocol?

Refer ppt Figure 11.21

1. If The sender sends a series of packets to the same destination using 5 bit sequence number. If the sequence number started with 0, what is the sequence number after sending 100 packets.

Sequence number — $0, 1, \ldots, 2^m - 1$

$m = 5$

$2^5 = 32$

Range of seq no. — {0, 1, 2, ... 31} 32

{0, 1, 2, ... 31} 32

{0, 1, 2, ... 31} 32

96

{0, 1, 2, 3, 4}

Seq no. after sending 100 packets = 4.

(100) mod 32 = 4

(no. of packets) mod 2^m = seq no.

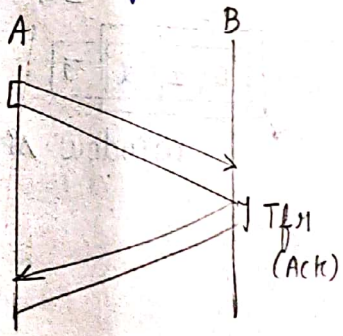
Round Trip Time (RTT)

sending a frame + receiving acknowledgement

RTT = 2x propogation time

Transmission time (Tfr)

↳ Time elapsed between the sending of first and last bit of a single frame.



Transmission time = $\frac{\text{framesize}}{\text{bandwidth}}$

(or) data rate

propogation time = $\frac{\text{distance}}{\text{speed}}$

total delay = RTT + transmission time.

2. A system uses stop and wait protocol. If each packet carries 1000 bits of data. How long does it take to send one million bits of data, if the distance between sender and receiver is 5000 km and the propogation speed is $2 \times 10^8 ms^{-1}$. (Ignore transmission waiting and processing delays). Assume no data frame (or) control frame is lost (or) damaged.

GATE

Frame size = 1000 bits

Total data to be sent = 1×10^6 bits

speed of transmission = 2×10^8 m/s

distance between sender and receiver = 5000 km

Time taken for "10^6" bits = ?

$$\text{Speed} = \frac{\text{distance}}{\text{time}}$$

$$\text{time} = \frac{5000 \times 10^3}{2 \times 10^8}$$

propogation time = 0.025 sec

(time taken by the data frame from source to destination)

RTT = 2 x propogation time

= 2 x 0.025

= 0.05 s

$$\text{no. of frames} = \frac{1 \times 10^6}{1000} \quad \frac{\text{(total data to be sent)}}{\text{(frame size)}}$$

= 1000 frames

Time taken for sending 1000 frames = 0.05 x 1000

= 50 sec

3. Repeat the previous problem using go-back N protocol with a window size of 7. Ignore the overhead due to header and trailer (Consider the propogation time)

4. Repeat the previous problem with selective repeat protocol.

Throughput:

No of bits that can be send including the delay.

(delay will not be considered) Bandwidth > Throughput

Throughput is given by window size / RTT (round trip time)

$$\text{Throughput} = \frac{\text{window size}}{\text{RTT}}$$

unit = bits/sec

5 Host A is sending data to Host B over a duplex ring A and B are using sliding window protocol. The sent and received windows are five frames each, data packets are 1000 bytes long and the transmission time for such a frame is 50 μsec. Acknowledgement packets are very small and require negligible transmission time. The propogation delay over the link is 200 μ sec. What is the maximum achievable throughput in this communication.

propogation delay = 200 μ sec

RRT = 2 x propogation delay

= 2 x 200 μ sec

= 400 μ sec

delay = (Tfr) + (2 x Tpr)

$= (50 \times 10^{-6}) + (2 \times 200 \times 10^{-6})$

$= 450 \times 10^{-6}$ sec

$$T.P = \frac{(5 \times 1000)}{450 \times 10^{-6}} = 11.12 \text{ bytes/sec (or) } 88.8 \text{ bits/sec}$$

6. Station A uses 32 byte frame to transmit message to B using a sliding window protocol. The RTT delay between A and B is 80ms and the bottle neck bandwidth is 128 kilobits/sec. What is the optimal window size that station A should use?

GATE

$$\text{Throughput} = \frac{\text{window size}}{\text{delay}}$$

$$(128 \times 10^3) = \frac{\text{window size}}{80 \times 10^{-3}}$$

$$\text{Window size} = (128 \times 80) \times 10^3$$
$$= 10240 \text{ bits}$$

frame size = 32 x 8 bits

$$\text{no. of frames} = \frac{10240}{32 \times 8} = 40 \text{ frames}.$$

7. Station A sends a message to station B which consist of 9 packets. The protocol used is go-back N. All the 9 packets are ready and immediately for transmission. If every fifth packet that A transmits gets lost. What is the number of packets that A will transmit for sending the entire message to B.

Ans 16 because every 5th packet is lost, so receiver will never send any ack, but sender will keep sending the frame available frame in the window.

3 2 1 (1 sent)
4 3 2 (2 sent)
5 4 3 (3 sent)
6 5 4 (4 sent)
7 6 5 (5 sent) --- 5th lost --- no ack
7 6 5 (6 sent) - no ack until receiver gets 5th
7 6 5 (7 sent) - no ack until receiver gets 5th

sender can no longer send anything now .... timeout for 5th.

7 6 5 (5 sent)
8 7 6 (6 sent)
9 8 7 (7 sent) – lost
9 8 7 (8 sent) – no ack
9 8 7 (9 sent) – no ack

timeout for 7

9 8 7 (7 sent)
9 8 (8 sent)
9 (9 sent) – lost

timeout

9 (9 sent)

Total = 16.

3. Answer.

propogation time = $\frac{D}{S} = \frac{5 \times 10^3}{2 \times 10^8} = 0.025$ sec

RTT = 2 x propogation time
= 2 x 25 x 10^{-3} sec
= 50 msec

Time taken for sending one frame = RTT
= 50 msec

no. of frames = $\frac{1 \times 10^6}{1 \times 10^3}$ = 1000 frames

window size = 7 (in 1000 frames, 7 frames can be sent without acknowledgement)

no. of windows = $\frac{1000}{7}$ = 143

Total time taken / delay = 143 x 50 x 10^{-3}
= 7.150 = 7.15 sec

4. Answer

propogation time $= \frac{D}{S} = \frac{5 \times 10^3}{2 \times 10^8} = 0.025$ sec

RTT = 2 x propogation time

$= 2 \times 25 \times 10^{-3}$

$= 50$ msec

Time taken for sending one frame = RTT = 50 msec

no. of frames $= \frac{1 \times 10^6}{1 \times 10^3} = 1000$ frames

window size = 4 (in 1000 frames, 4 frames can be sent without acknowledgement)

no. of windows $= \frac{1000}{4} = 250$

Total time taken / delay $= 250 \times 50 \times 10^{-3}$

$= 12.5$ secs.

Including Transmission Delay (TD) in flow control:-

$$TD = \frac{\text{size of the window (in bits)}}{\text{total bits to be sent}}$$

For the above problems,

3rd sum:

$$TD = \frac{7 \times 1000}{1 \times 10^6} = 7 \text{ msec}$$

4th sum:

$$TD = \frac{4 \times 1000}{1 \times 10^6} = 4 \text{ msec}$$

Framing :-

→ The simple act of inserting a letter into an envelope seperates one piece of information from another, the envelope serves as the delimiter.

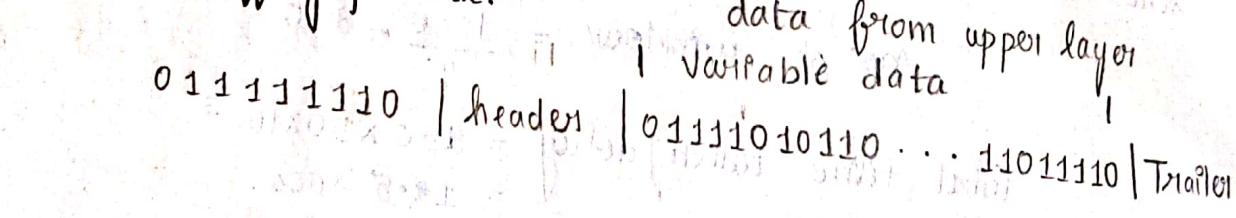
→ The datalink layer needs to pack bits into frames, so each frame is distinguishable from another.

It involves:

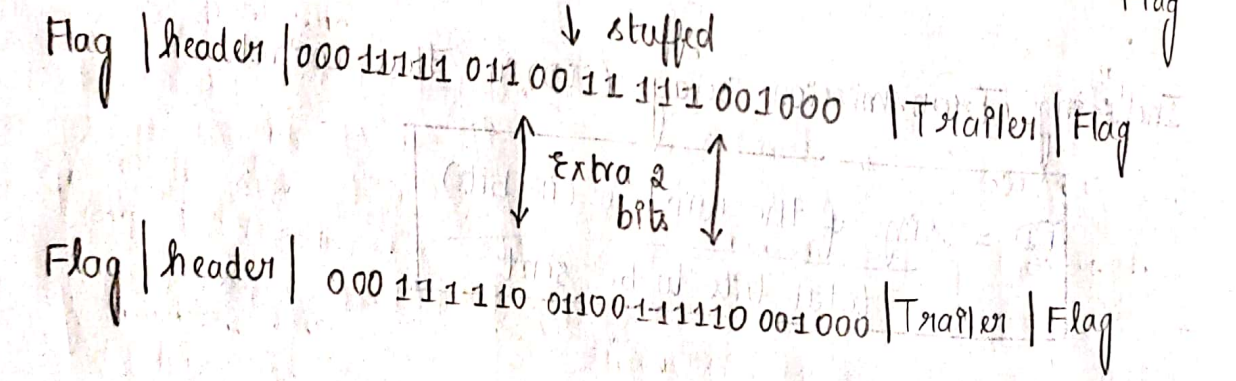
⇒ receives bits from the network layer

⇒ add headers, traplers (to distinguish the frame from another. we use a flag)

Bit stuffing protocol:



Frame sent



Bit stuffing (sender side):

A flag value is always 01111110 whenever 0 followed by 5 1's, append 0.

Bit unstuffing (receiver side):

whenever data is received starts from 0 followed by 5 1's it discards 0.

1000 11111|1100 11111|0100 11111|11111 0000 11111

100 111110 1100 111110 0100 111110

Unstuff the following frame payload:

000 11111|000 00 11111|0 1110100 1110 11111|00000 1111

000 111110 00000 111110 01110100 1110 111110 00000 1111.

Byte stuffing:-

Data from upper layer

| |Flag| | |Esc| |

↓ stuffed

Frame sent

|Flag| Header | |Esc|Flag| | |Esc|Esc| |Trailer|Flag|

escape sequence is added if there is a flag already present ↕ extra 2 byte ↕ escape sequence is added when already an Esc sequence is prevalent.

Frame received

|Flag| Header| |Esc|Flag| | |Esc|Esc| |Trailer|Flag|

↓ unstuffed (to remove the escape sequence

| |Flag| | |Esc| |

Byte stuff the following data:-

|Esc| | |Flag| | |Esc|Esc|Esc| |Flag| |

|Esc|Esc| | |Flag|Esc| | |Esc|Esc|Esc|Esc|Esc|Esc| |Esc|

Flag| |

Unstuff the following frame payload in which E is escape byte, F is flag byte, D is data byte

E | E | D | E | F | D | D | E | F | E | E | D | D | D

Multiple Access: (access control)

→ since there are more number of devices, the device to be transmitted is decided.

Data link layer (two sub layers)

- Data link control (framing, error control, flow control)
- ↓
- Multiple access resolution (Access control)

Only layer in which we add a trailer and can be subdivided into two functional layers.

Multiple access protocols

- random access protocols
  - no master, no slave
  - all the devices connected to the transmission medium have equal access
  - ALOHA, CSMA, CSMA/CD, CSMA/CA
- control access protocols
  - exclusive device as master, which controls entire access processes
  - eg:- reservation, polling token passing
- channelization protocols
  - based on multiplexing techniques
  - eg: FDMA, TDMA, CDMA

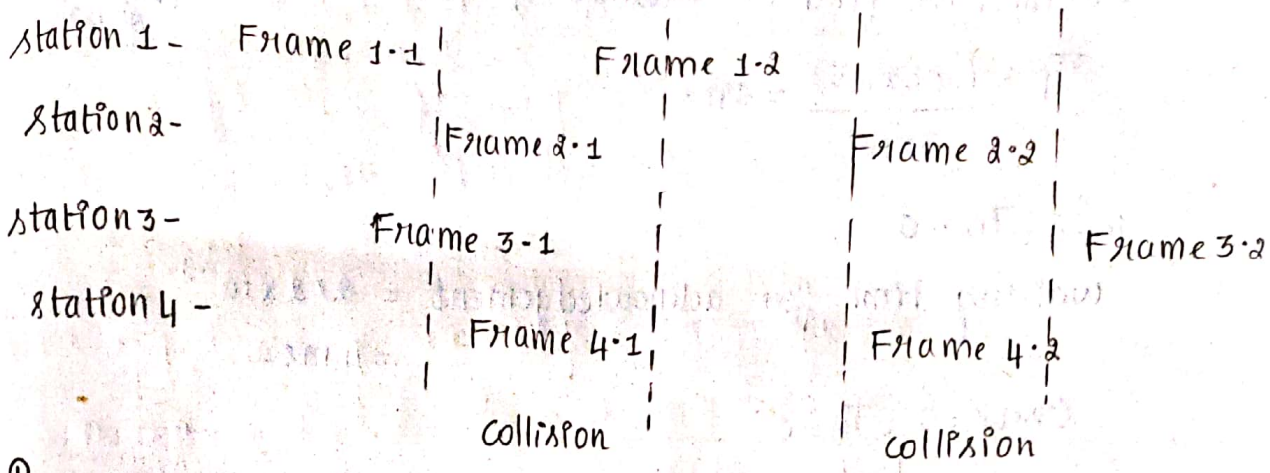
Frames in ALOHA network:

→ whenever some station has data to send, it sends the data, does not bother about the already present data.

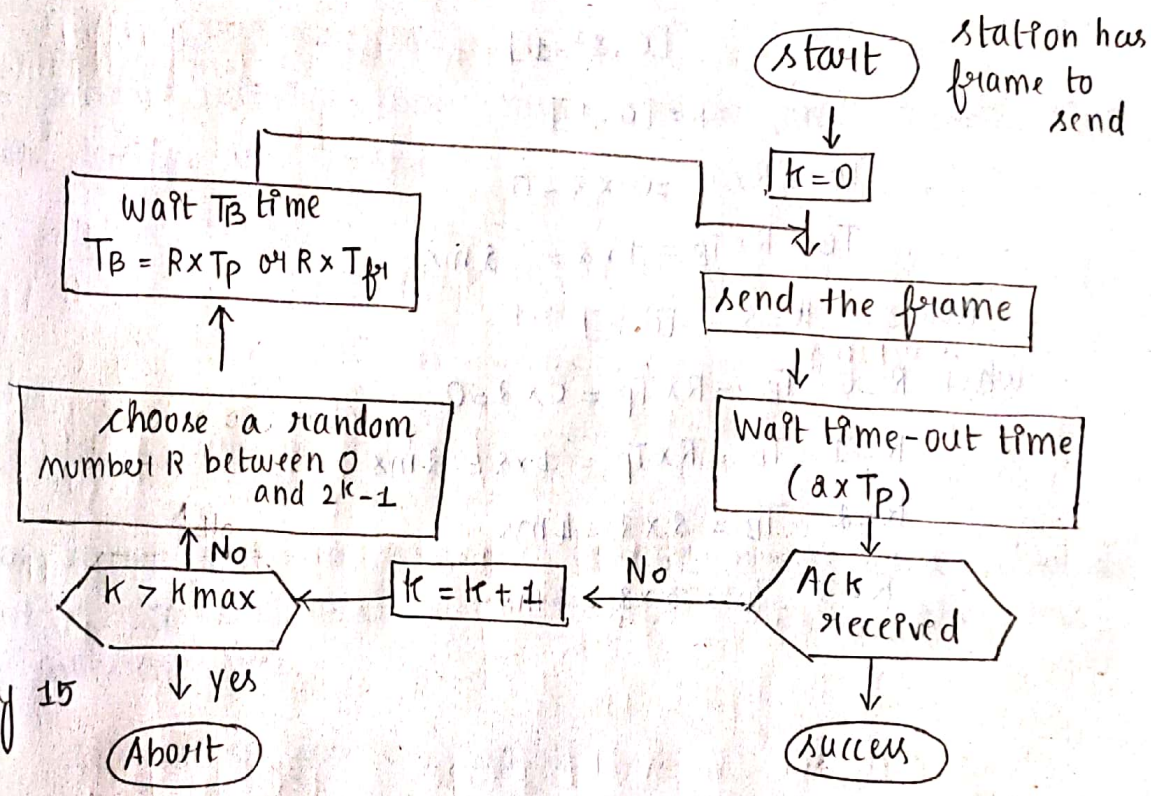
drawback:

→ two or more stations may transmit data at the same time, which will lead to collision.

→ To ensure correct transmission of data, we have acknowledgements.



Procedure for pure ALOHA protocol:



K - number of attempts

Tp - maximum propogation time

TB - back-off time

Tfr - Average transmission time for a frame.

1. The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propogate at 3×10^8 m/s, we find the value of T_B (for different values of k)

$d = 600\text{km}$

$s = 3 \times 10^8 \text{ m/s}$

$T = \frac{d}{s}$

$$T_p = \frac{(600 \times 10^5)}{3 \times 10^8} = 2 \text{ ms}$$

$k = 0 ; T_B = 0$

waiting time for acknowledgement $= 2 \times 2 \times 10^{-3}$

$= 4 \text{ msec}$

Case 1: $k = 0$, $T_B = 0$

Case 2: $k = 1$ 0 and $2^k - 1$

$[0, 2^1 - 1]$

$R = [0, 1]$

$T_B = R \times T_p = 0 \times 2 = 0$

$T_B = R \times T_p = 1 \times 2 = 2 \text{ ms}$

Case 3: $k = 2$ $[0, 3]$

when $R = 0$ $T_B = R \times T_p = 0 \times 2 = 0$

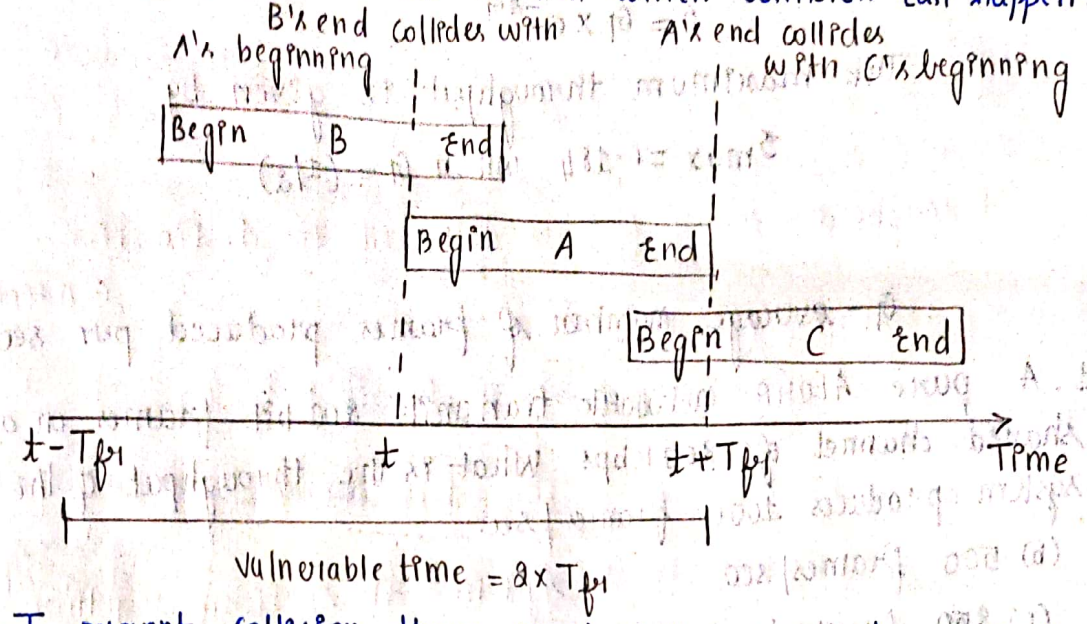
$R = 1$ $T_B = R \times T_p = 1 \times 2 = 2 \text{ ms}$

$R = 2$ $T_B = 2 \times 2 = 4 \text{ ms}$

$R = 3$ $T_B = 3 \times 2 = 6 \text{ ms}$

Vulnerable time for pure ALOHA protocol :-

→ maximum time within which collision can happen.



→ To prevent collision, there must not be any transmission between t - Tfr and t + Tfr.

Vulnerable time = (t + Tfr) - (t - Tfr)

t + Tfr
t - Tfr
(-) (+)
2 Tfr

1. A pure ALOHA network transmits 200 bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision free.

Length of frame = 200 bit

capacity of channel = 200 kbps

Vulnerable time = 2 Tfr

200×10^3 = 1 sec

200 bits = ?

$T_{fr} = (200) / (200 \times 10^3)$

= 1 msec

Vulnerable time = 2 Tfr

= 2 x 1 msec

= 2 msec

Note:

The throughput of pure ALOHA is given by

$$S = G \times e^{-2G}$$

The maximum throughput is given by

$$S_{max} = 0.184 \text{ when } G = (1/2)$$

Throughput - No of frames delivered to destination correctly

G - average number of frames produced per second.

1. A pure ALOHA network transmits 200 bit frames on a shared channel of 200 kbps. What is the throughput of the system, produces 1000 frames/sec

(b) 500 frames/sec system - all stations together.

(c) 250 frames/sec

Length of frame = 200

capacity of ~~frame~~ channel = 200 kbps

Throughput $S = G \times e^{-2G}$;

G is the average no of frames produced per second.

(a) 1 sec = 1000 frames

1 msec = ?

T_{fr} = 1 msec

(G) Average no of frames = $1000 \times 10^{-3} \times 1$

= 1 frame.

Throughput $S = G \times e^{-2G}$ (Frames delivered per millisecond)

$= 1 \times e^{-2}$

$= 1 \times 0.135$

$S = 0.135$

In seconds. (Frames delivered per second)

$S = 1000 \times 0.135 = 135$ / 1000 frames

(b) 1 sec = 500 frames

1 msec = $10^{-3} \times 500$

$G = 0.5$

Throughput $S = G \times e^{-2G}$

$= 0.5 \times e^{-1}$

$S = 0.183$ per ms

$S = 500 \times 0.183$

$= 91.96$

$S = 92$ (per 500 frames)

(c) 1 sec = 250 frames

1 msec = ?

$= 250 \times 10^{-3}$

$G = 0.25$

Through $S = G \times e^{-2G}$

$= 0.25 \times e^{-0.5}$

$= 0.6065 \times 0.25 = 0.1516$

$S = 250 \times 0.6065 \times 0.25$

$S = 250 \times 0.1516$

$S = 38$ (per 250 frames)

Frame in a slotted ALOHA network:

→ We donot have continuous sequence of time.

→ Data is sent only in the beginning of time slots.

→ Collision occurs when two stations try to transmit data in the same slot.

Vulnerable time = T_{fr}

Note:

throughput for slotted ALOHA is

$S = G \times e^{-G}$

The maximum throughput

$S_{max} = 0.368$ when $G = 1$ (due to vulnerable time)

CSMA - Carrier Sense Multiple Access

→ The carrier is sensed, and if the channel is free, only then it is transmitted.

types:

1. non persistent CSMA
2. 1 - persistent CSMA
3. p - persistent CSMA

Non - persistent CSMA:-

→ A stations with frames to be send, should sense the medium,

1. If ~~idle~~ medium is idle, transmit, otherwise go to 2.
2. If medium is busy, (backoff) wait a random amount of time and repeat 1.

→ non persistent stations are differential (respect others)

performance:

→ Random delays reduces probability of collisions.

1 - persistent CSMA:-

→ To avoid idle channel time, 1 - persistent protocol is used.

→ station wishing to transmit listens to the medium

1. If medium idle, transmit immediately.
2. If medium busy, continuously listen until medium becomes idle, then transmit immediately with probability 1-

p - persistent CSMA:-

→ Time is divided to slots where each Time unit (slot) typically equals maximum propogation delay.

→ station wishing to transmit listens to the medium.

1. If medium is idle.
   ⇒ transmit with probability (p). OR
   ⇒ wait one time unit (slot) with probability (1-p).

then repeat 1.

2. → If medium busy, continuously listen until idle and repeat step 1.

performance:

→ Reduces possibility of collisions like non persistent.

→ Reduces channel idle time like 1 persistent.

CSMA (Collision detection CD)

→ A station after| during transmission, it continuous senses the channel to check collision.

→ If collision, it stops current transmission and informs all other stations by sending jamming signals.

→ power of channel is monitored.

power ↑ collision occurs

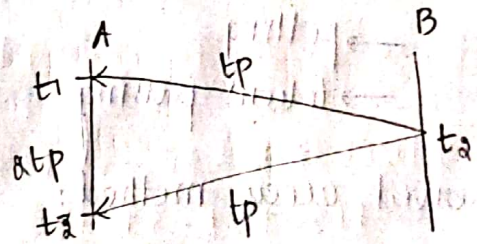
→ There is always a chance of collision even if the the channel detect due to delay in propogation time.

→ different devices connected to the network at different of times will detect collision.

Minimum duration in which a station must transmit data in order to detect collision = 2 (RTT) (Round Trip Time)

For diagram, refer ppt.

1. A network using CSMA CD has a bandwidth of 10 Mbps. If the maximum propogation time is 25.6 μs, what is the minimum size of the frame.



$t_p = 25.6\ \mu s$

Minimum duration for which 'A' should transmit $= 2 \times t_p$

$= 2 \times 25.6 \times 10^{-6}$

$= 51.2\ \mu s.$

1 sec = 10 Mb

51.2 μs = ?

$= 51.2 \times 10^{-6} \times 10 \times 10^{6}$

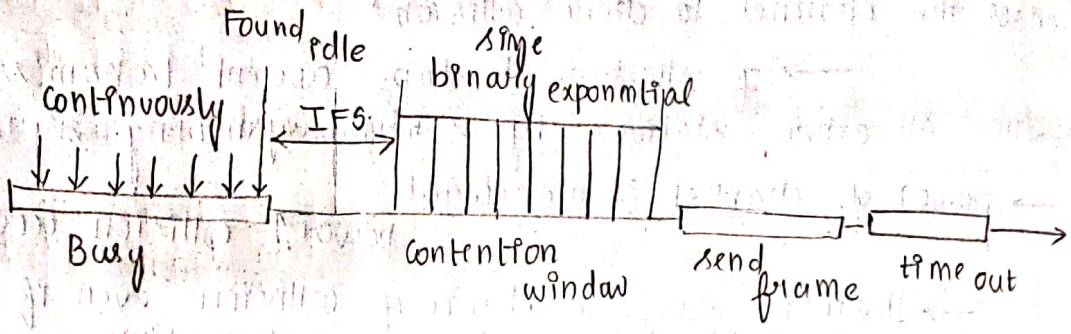
$= 512$ bps.

Timing in CSMA (Collision Avoidance - CA)

→ The collision is prevented from occuring.

Interface sequence time (IFST):

→ fixed time in which the channel waits, which is fixed for all devices.



→ There is least chance of collision, since they are two waiting time slots.

Controlled Access:

→ Controlling access is a mechanism of controlling the access to transmit and receive data.

→ There are three different methods involved

→ Reservation

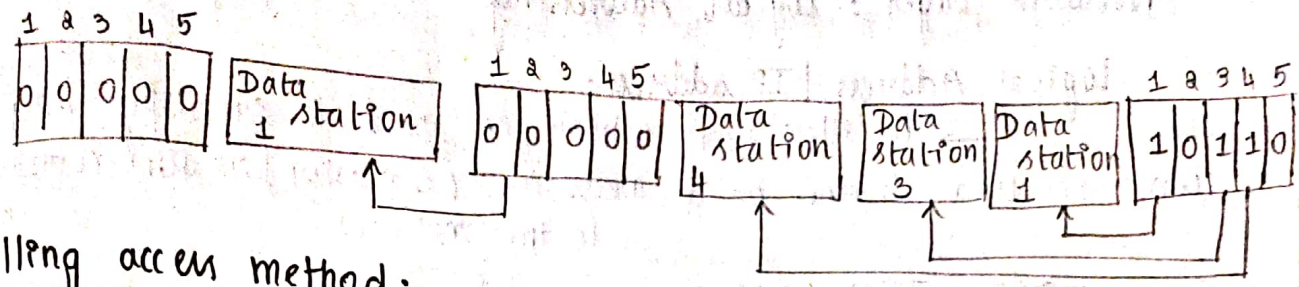
→ polling

→ token passing

Reserved access method:

→ The data is reserved for a particular duration, in which they are some equal time slots.

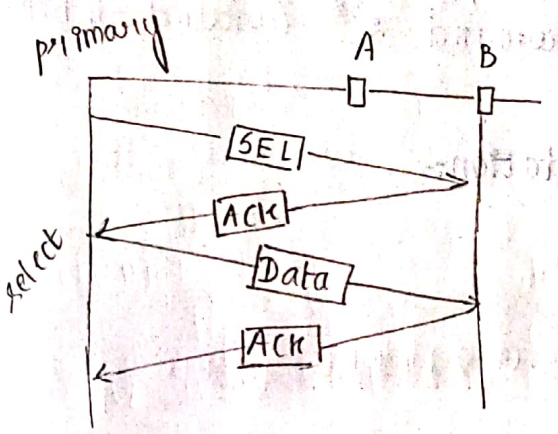
→ do a reservation before transmition, so other stations restrain from sending, thus avoiding collisions.

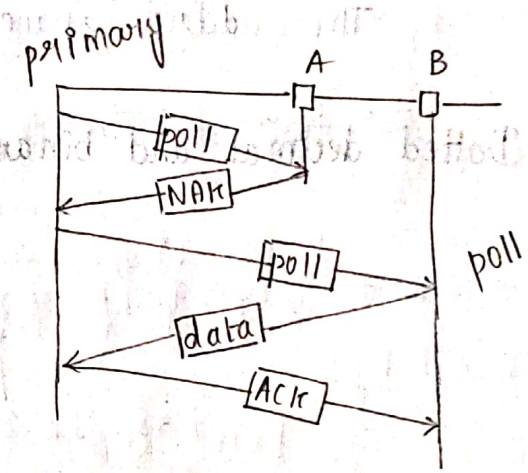


Polling access method:

→ Within a topology, we have one primary and one secondary station.

→ Primary station decides the data to be transmitted.





Token passing access method:

→ similar to a ring topology, devices are connected across each other.

→ "token" is a special frame pattern, which is circulated within the ring.

→ A device has some data to transmitted, capture the token, hold the token and then transmit the data.

→ After sending all the frames, the station will receive the token. Other devices must check for the intended receiver, and accept or pass it to the other.

Channelization : refer ppt

(similar to multiplexing)

Network Layer : Logical Addressing

Logical Address | IP address:

→ an identifier which is used to identify any communication device on a network. (Computer | router connected to the internet)

→ An IPV4 address is 32 bit long.

→ Logical address is always unique and universal.

An IP address which is 32 bit long, can address about 2^{32} people | users.

The address space is around 2^{32} (around 4 billion)

Dotted decimal and binary notation:-

10000000 00001011 00000011 00011111

188. 11. 3. 31

Every byte is followed by a 'dot'.

Change the following IPV4 address from binary to dotted decimal notation.

(a) 10000001 00001011 00001011 11111111

129. 11. 11. 239

(b) 11000001 10000011 00011101 11111111

193. 131. 29. 255

Convert the following IPV4 address from dotted decimal to binary:

(a) 111. 56. 45. 78

01101111 00111000 00101101 01001110

(b) 221. 34. 7. 82

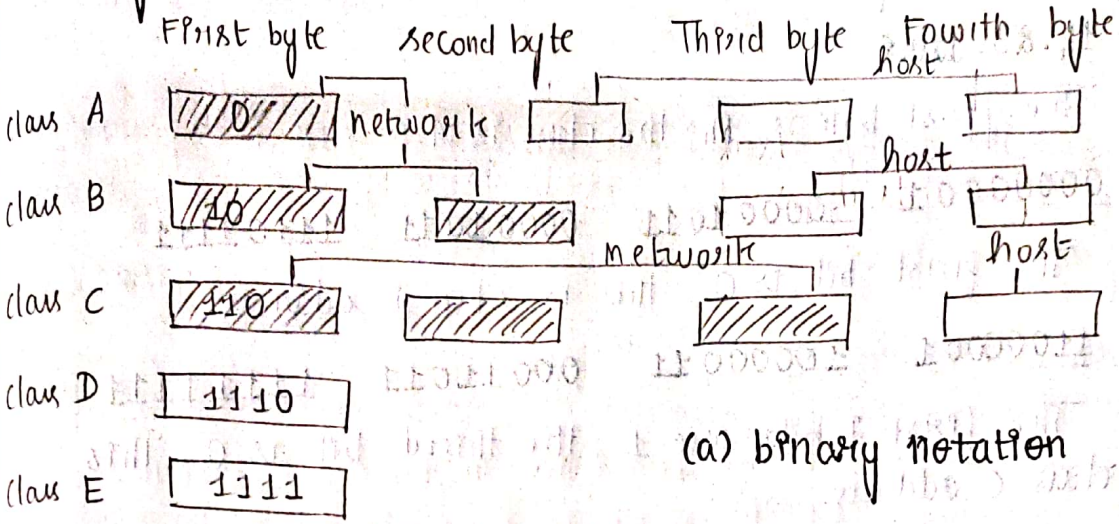
11011101 00100010 00000111 01010010

note:

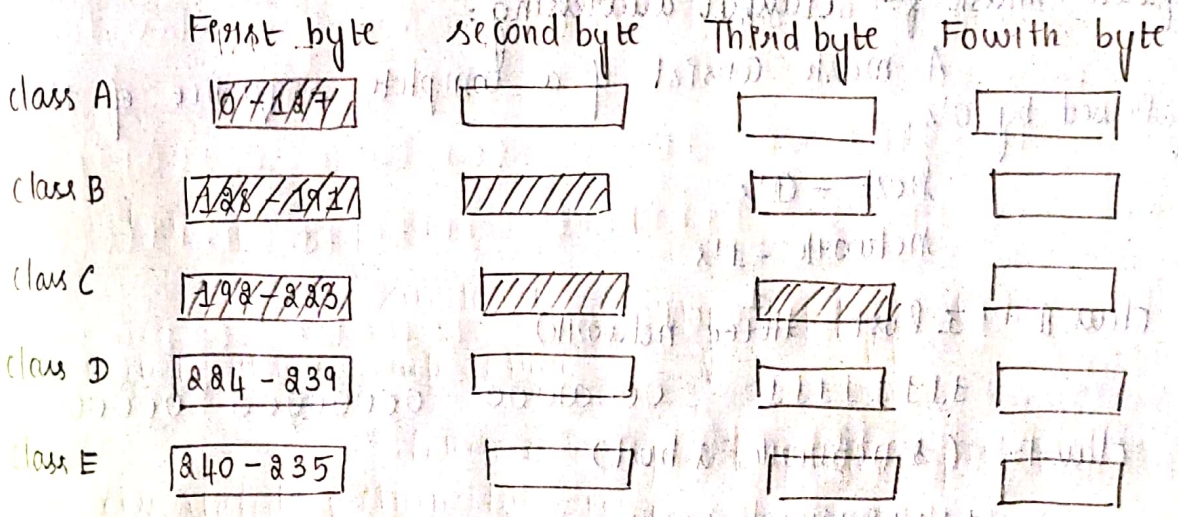
In classful addressing, the address space is divided into five classes:

A, B, C, D and E. (based on its ability to accomodate)

finding the classes:



(a) binary notation



(b) dotted decimal notation.

No. of posts to be connected.

class A - 2^{24}

class B - 2^{16}

class C - 2^{8}

If the first bit in the first byte starts with 0, then it is class A.

First byte - 10 - class B

110 - class C

1110 - class D

1111 - class E

Find the class of each address:

(i) 252.5.15.111

The first byte is 252, the class is E.

(ii) 14.23.120.8

The first byte is 14; the class is A

(iii) 00000001 00001011 00001011 11101111

The first bit is 0; this is class A address.

(iv) 11000001 10000011 00011011 11111111

The first 2 bits are 1, the third bit is 0. This is class C address.

Default mask for classful addressing:

A mask consist of a complete sequence of 1's followed by 0's.

host - 0's

network - 1's

class A (1 host, ~~three~~ 1 network)

11111111 00000000 00000000 00000000

class B (2 network, 2 host)

11111111 11111111 00000000 00000000

class C (3 network, 1 host)

11111111 11111111 11111111 00000000

class A 255.0.0.0 CIDR - /8

class B 255.255.0.0 CIDR - /16

class C 255.255.255.0 CIDR - /24

Classless addressing:

→ There are no classes, wherein the logical address is taken as a whole.

→ It is done since there is a lot of waste in classful addressing.

→ Since the requirements are not uniform, variable and are in powers of 2, there is a lot of wastage. ICAA (Internet Control Authority Agency) provides the IP address to configure to your machine

note:

In IPV4 addressing, a block of addresses can be defined as x.y.y.t/n (n-mask) binary notation of mask: 'n' 1's followed by 'n' 0's.

	Block	Block	
First	205.16.37.32	11001101 00010000 00100101 00100000	
	205.16.37.33	11001101 00010000 00100101 00100001	
	⋮	⋮	
Last	205.16.37.47	11001101 00010000 00100101 00101111	

1. Ip address given in 205.16.37.39/28. What is the first address in the block? What is the last address in the block?

Binary representation of given address

11001101 00010000 00100101 00100111

If we set 32-28 rightmost bits to 0, we get

11001101 00010000 00100101 00010000

(or)

205.16.37.32.

note:

The last address in the block can be found by setting the rightmost (32-n) bits to 1's.

If we set 32-28 rightmost bits to 1, we get

11001101 00010000 00100101 00100111

(or)

205.16.37.47.

Note:

The number of addresses in the block can be found by using the formula 2^{32-n}.

Method: 2

(a) Finding first address in the block

1. Represent the IP address in binary notation
2. Represent the mask in binary notation.
3. perform bitwise 'AND' operation of the above two.

Eg:-

Address:	11001101	00010000	00100101	00100111
mask:	11111111	11111111	11111111	11110000
First address:	11001101	00010000	00100101	00100000

(b) Finding the last address:

1. Represent the IP address in binary notation
2. Represent the mask in binary notation.
3. find the complement of the mask
4. perform 'OR' operation of mask complement and IP

Eg:-

Address:	11001101	00010000	00100101	00100111
mask:	00000000	00000000	00000000	00001111
last address:	11001101	00010000	00100100	00101111

(c) Computing the number of addresses:

1. Complement the mask
2. Represent the complement in binary
3. add 1 to it

Eg:

mask complement:

00000000 00000000 00000000 00001111

No° of addresses:

15+1 = 16

Conditions for classless addressing:

1. In case of classless addressing, the IP address must be continuous.
2. The number of addresses available will always be a power of 2.
3. The first address, when converted to its decimal notation, must always be divisible to 16.

Note:

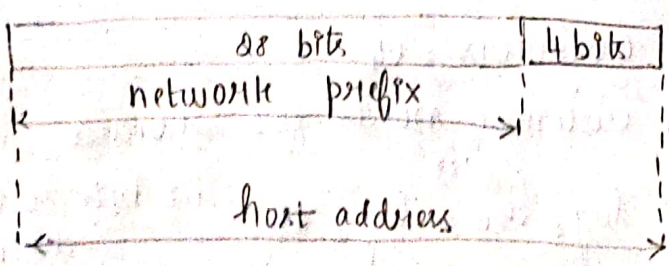
The first address in a block is not assigned to any device. It is used as network address that represents the organisation to rest of the world.

Subnetting:

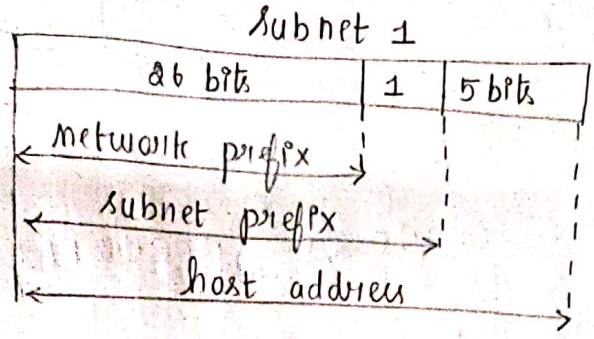
→ All subnetwork are interconnected to a network. Devices from different locations can be connected.

→ Process of dividing networks into subnetworks are called subnetting.

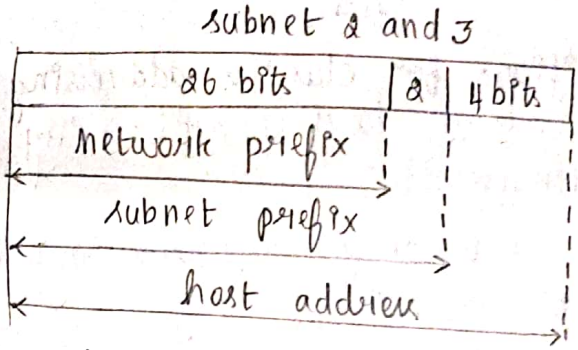
(Two level hierarchy)



Eg: 205.16.37.32
network host



(Three level hierarchy)



GATE

1. An ISP (Internet Service Provider) is granted a block of addresses starting with 190.100.0.0/16. The ISP needs to distribute these addresses to three groups of customers as follows:

(a) The first group has 64 customers and each needs 256 customers.

(b) The second group has 128 customers and each needs 128 addresses

(c) The third group has 128 customers and each needs 64 addresses

Design the sub blocks and find out how many addresses are available after these allocations.

Group 1:

No. of customers = 64

Each customer needs 256 addresses

(i) $\log_2 256 = \log_2 2^8$ (8 bits are needed to define each host)

(ii) 32 − 8 = 24 subnet masks

1st customer:-

190.100.0.0/24 to 190.100.0.255/24

(every customer will have unique address)

2nd customer:

190.100.1.0/24 to 190.100.1.255/24

3rd customer:

190.100.2.0/24 to 190.100.2.255/24

64th customer:

190.100.63.0/24 to 190.100.63.255/24.

group: 2

No of customers = 128

each customer need 128 address

(i) $\log_2 2^7 = \log_2 128$

$= 7$

(ii) 32-7 = 25 subnet masks (prefix length)

1st customer:

190.100.64.0/25 to 190.100.64.127/25

2nd customer:

190:100:64.128/25 to 190.100.64.255/25

3rd customer:

190.100.65.0/25 to 190.100.65.127/25

128th customer:

190.100.127.128/25 to 190.100.127.255/25

group 3:

no of customers = 128

each customer needs 64 address

(i) $\log_2 64 = \log_2 2^6$

$= 6$

(ii) 32 - 6 = 26 (prefix length)

1st customer: 190.100.128.0/26 to 190.100.128.63/26

2nd customer: 190.100.128.64/26 to 190.100.128.127/26

3rd customer: 190.100.128.128/~~64~~ 26 to 190.100.128.191/~~64~~ 26

4th customer: 190.100.128.192/26 to 190.100.128.255/26

128th customer: 190.100.159.192/26 to 190.100.159.255/26

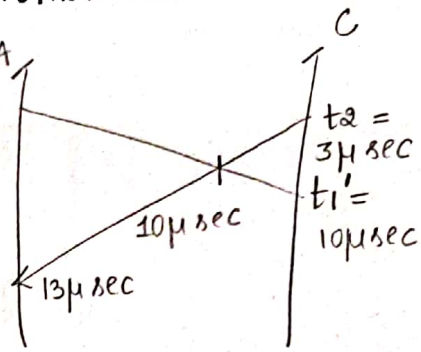
Total = 128 × 64 = 8192

2. An ISP is granted a block of addresses, ~~first~~ 150.80.0.0/16.

(a) first group has 200 customers, each need 128 address
(b) second group has 400 customers, each need 16 address
(c) third group has 2000 customers, each need 4 address
Design subblocks and find out how many addresses are available after allocations

In a CSNA CD network, the data rate is 100 Mbps. The distance between the stations A and C is 2000m and the propogation speed is 2×10^8 m/s. Station A sends a long frame at time $t_1 = 0$, C sends a long frame at time $t_2 = 3\mu s$. The size of the frame is long enough to detect the collisions. Find

(i) the time when 'C' hears the collision
(ii) the time when 'A' hears the collision
(iii) no. of bits sent by A before collision
(iv) no. of bits sent by C before collision



data rate = 100 Mbps

propogation speed = 2×10^8 m/s

distance = 2000 m.

(i) time taken (tp) = $\frac{D}{S} = \frac{2000}{2 \times 10^{-8}} = 10^{-5}$ s (or) 10 μs

$t_2 = 3\mu$sec

$t_1 = 10\mu$ sec

(ii) $t_2 + t_1' = 10\mu\text{sec} + 3\mu\text{sec} = 13\mu\text{sec}$

(iii) 1 sec = 100 Mb

$13\mu\text{sec} = \frac{13 \times 10^{-6} \times 100 \times 10^6}{1} = 1300$ bits

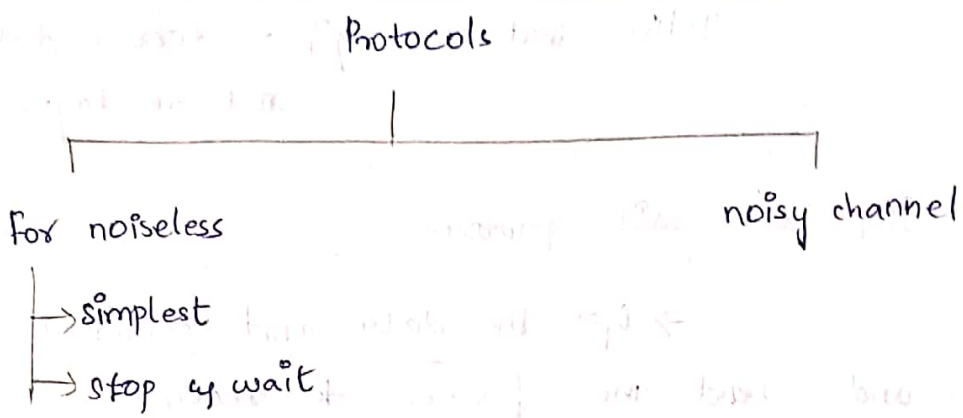
(iv) 1 sec = 100 Mb

$7\mu\text{sec} = \frac{7 \times 10^{-6} \times 100 \times 10^6}{1} = 700$ bits

Flow control:

Refers to a set of procedures used to restrict, the amount of data that the sender can send before waiting for acknowledgement.

Taxonomy:



Protocol with no error or flow control.

[Figure in ppt]

Algorithm for sender site:

```
while (true)
{
 wait for event();
 if (Event (Request to send))
 {
   Get data();  → Get data from network layer
   Make frame(); → making frames
   Send frame(); → Sending frame to receiver.
 }
}
```

Algorithm for Receiver site:

```
While (true)
{
  wait for Event ();
  If (Event (Arrival Notification))
  {
    Receive Frame ();
```

Extract Data();
Deliver Data(); } → deliver data to network layer.

Stop and wait protocol:

→ Get the data and converted to frame and send the frames to receiver.

→ Receiver receives the frames and extract data from it.

→ Receiver sends the acknowledgement frame to sender. Sender wait until the ack is received and then terminates.

Algorithm for sender

```
while (true)
 cansend = true
{
 Wait for Event();
if (Event (Request to send) AND cansend) // Arrival Notification
{
 Get data();
 Make frame();
 send frame();
 cansend = false;
}
 Wait for Event();
if (Event (Arrival Notification)
{ Receive frame();
   cansend = true;
}}
```

Algm for Receiver

```
// Receive frame();
// Extract data();
// deliver (data);
// Send frame();
```

→ No need for receiver

Cons:

→ The data may be lost

→ The frames may be corrupted

→ There is delay in acknowledgment.

Stop and wait ARQ:

cases: Fig 11.11 in ppt.

1) Frame send and ack received.

2) Frame sent is lost.

3) Frame is resent and ack received

4) Frame is send and ack is lost.

5) Frame is resend (duplication) and ack is received. The duplicate will be discarded.

Sequence number:

Sequence numbers are identifier. It is either 0 or 1. It is identifier to data frames and acknowledgement frames. Incase of stop and wait Protocol for noisy channel the sequence numbers is operated under modulo 2 arithmetic.

→ If frame 0 sent ack 1 will be received (it refers to the sequence of next expected frame).

→ If frame sent is 1, ack 0 will be received.

Algorithm Sender:

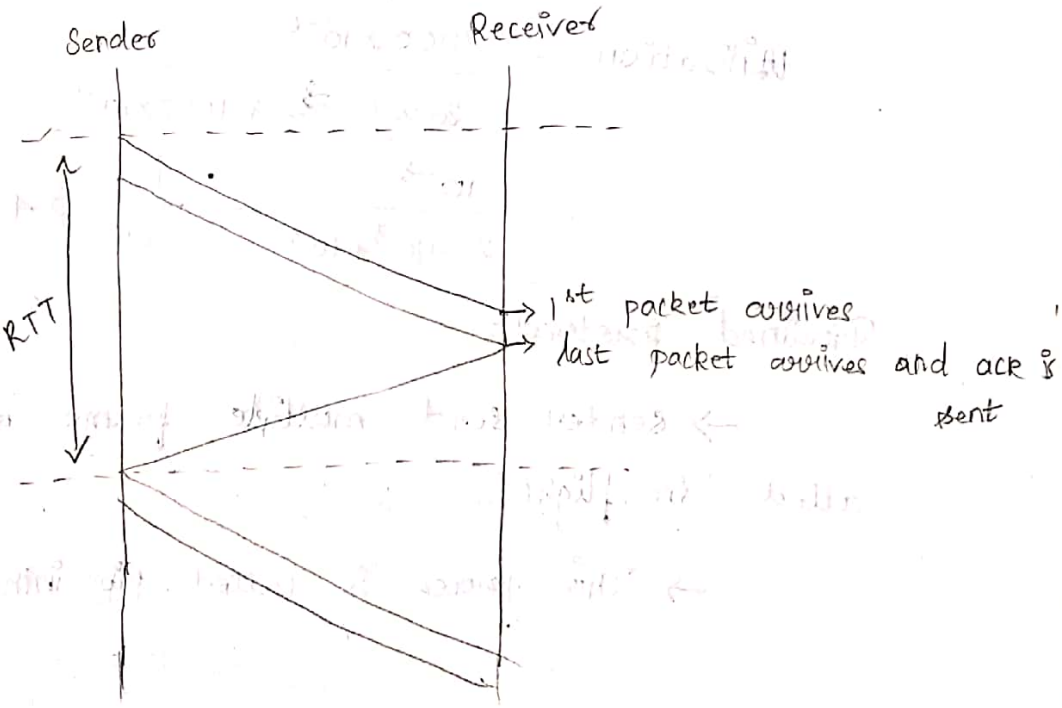
```
Sn = 0
cansend = true;
While (true)
{
Wait for Event();
If (Event (Request to send) AND cansend)
{
Get data();
Make frame(sn);
Store frame (sn);
Send frame (sn);
Start Timer();
sn = sn + 1
cansend = false;
}
Wait for Event();
if (Event (Arrival Notification)
{
Receive Frame (ack NO);
If (not corrupted AND ack NO == sn)
{
Stop timer();
Purge Frame (sn-1);
cansend = true;
}}
if (Event (time out) {
start Timel);
Resend frame (sn-1); }}.
```

Algorithm Receiver:

```
Rn = 0;
While (true
{
  wait for Event();
  if (Event (Arrival Notification))
  {
    Receive frame();
    if (corrupted (frame));
       sleep();
    if (seqNO == Rn)
    {
      Extract Data();
      Deliver Data();
      Rn = Rn+1;
    }
    Send Frame (Rn);
  }}
```

Rn → seq. no of next packet expected.

Bandwidth Utilization:



RTT - Time period of sending frame and receiving ack.

L - Length of frame.

Utilization $\propto$ L

Utilization $\propto \frac{1}{R}$ → Bandwidth of channel.

$$\text{Utilization} = \frac{\text{used time}}{\text{Total time}} = \frac{L/R}{RTT + \frac{L}{R}}$$

Fig: 11.4

Assume that in stop and wait ARQ sys the bandwidth of line is 1Mbps and 1 bit takes 20ms to make a round trip. If the system data frames are 1000 bits in length. What is Utilization Percentage of the link?

$R = 1\ Mbps = 10^6$, $L = 1000$

$RTT = 1000 \times 20 \times 10^{-3} = 20ms$

$$\text{Utilization} = \frac{1000 \times 10^{-6}}{20 \times (10^{-3}) + 1000 \times 10^{-6}}$$

$$= \frac{10^{-3}}{20 \times 10^{-3} + 10^{-3}} = \frac{1}{21} = 4.8\%$$

Pipelined protocol:

→ Sender send multiple frame which is called 'in-flight'.

→ This process is called pipelining.

Go Back N (pipeline protocol)

Window → refers to max no. of frames that can be sent without receiving an ack.
represented by S_{IO}.

If sw = 8, the seq nos = {0, 1, 2, ... 7}
= {0, 1, ... $S_w - 1$}

Sf - refers to first frame in window waiting for ack.

Sn - refers to next frame that is to be sent.

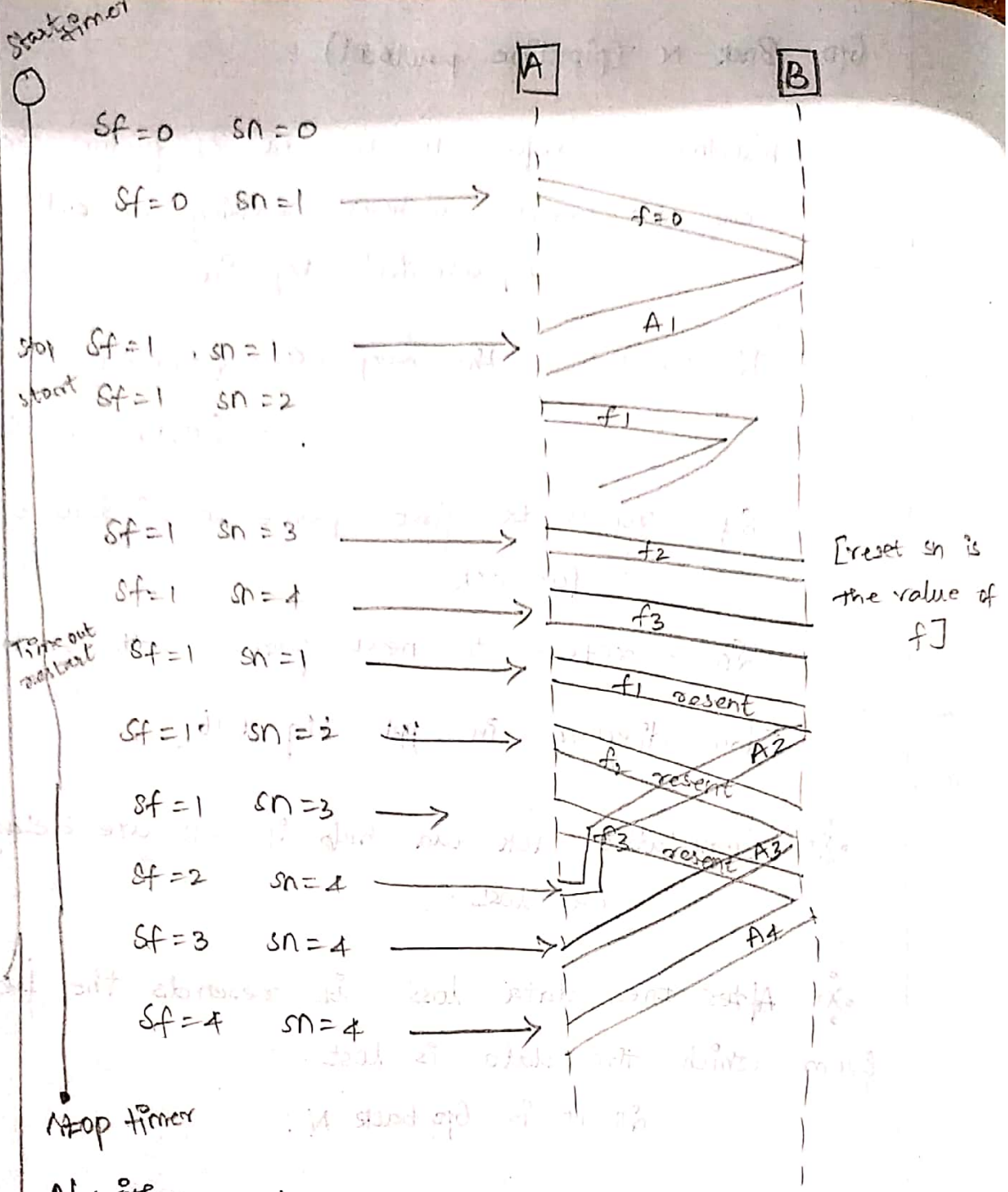
Flow diagram in ppt Fig 11.16.

∴ Cumulative ack can help if ack are delayed or lost.

∴ After the data loss it resends the frame from which the data is lost.
So it is Go back N.

Flow diagram for data loss 11.7

↳ Continued on back side



Algorithm refer ppt 11.7

$S_W = 2^m - 1$

```
Sf = 0;
Sn = 0;
While (true)
{
  wait for Event();
    if (Event( Request to send))
     {
       if (Sn - Sf >= Sw)
           Sleep();
       Get data();
```

```
        Make frame (Sn);
      Store frame (Sn);
      2nd Frame (Sn);
       Sn = Sn+1;
       if (timer not running)
            start timer();
  }
  if (Event ( Arrival Notification))
  {
    Receive (ACK);
    if (corrupted (ACK))
         sleep();
    if ((ack.NO >sf) && (ackNO <=sn))
    {
      while (sf <= ackNO)
      {
        purge frame (sf);
        sf = sf +1
      }
      If (sf == sn )
          Stop Timer();
  } }
  If (Event (Time out ))
  {
   start Time();
   Temp = sf
   while (temp < Sn)
   Sendframe (tmp)
   tmp++
  }
}
```