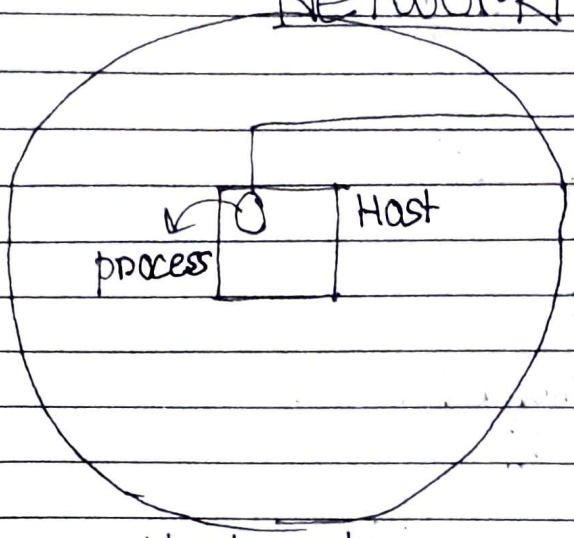
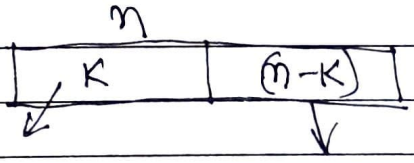
# Networking




(i) Destination network
(ii) " host
(iii) " process

Domain name
↓ DNS (ISP)
ip address
↓
Network + Host
id id
port no : (80)
HTTP

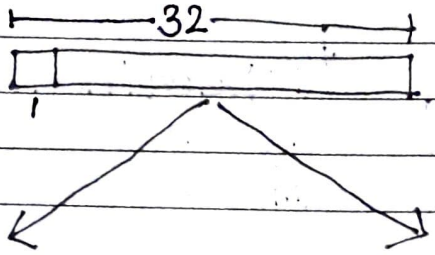
DNS overhead.
→ recent IP addresses are cached




number of networks

size of each network
$2^{n-K}$

# Classful IP addressing:




First bit 0
$2^{31}$
Class A

First bit 1
$2^{31}$

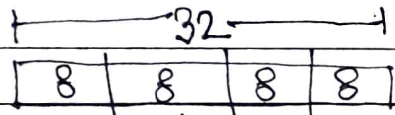
2nd bit 0
$2^{30}$
Class B

2nd bit 1
$2^{30}$

prefix code
↓
unique prefix

3rd bit 0
$2^{29}$
Class C

3rd bit 1
$2^{29}$



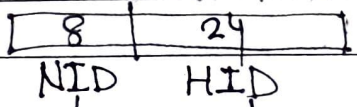

Dotted decimal representation

4th bit 0
$2^{28}$
Class D

4th bit 1
$2^{28}$
Class E

## Class A:

$2^{31}$ IP

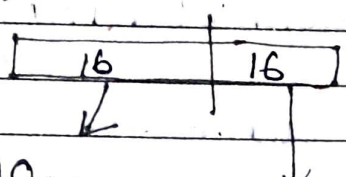



First bit 0.
$2^7$ → networks
$2^{24}$ → each network size

→ all 0 and all 1 not used

Date: / / Page

## Class B




10...

$2^{14}$ → NID $2^{16}$ → HID

First and Last HID are not used for any @ network.

10 ..... First octate

10000000 — 128

10111111 — 191

NID: 128.0.

128.1

128.2

⋮

128.255

129.0

⋮

191.0

⋮

191.255

## Class C


110

$2^{21}$ NID $2^8$ HID

110 00000 192

110 11111 223

Class D — 1110

Class E — 1111 → military purpose

no division NID/HID

multicasting / group broadcasting

Casting → unicast
→ broadcasting → limited
→ directed

HID → all 0s (1st address)
↓
network i'd

Limited broadcasting → to all host in the same network
255.255.255.255
↓
all 1s ~~(last address)~~

Directed broadcasting → to all host in other network
↓
NID + all 1s in HID (Last add)

IP : 1.2.3.4 ClassA
NID : 1.0.0.0
DBA : 1.255.255.255

IP : 130.1.2.3 Class B
NID : 130.1.0.0
DBA : 130.1.255.255

IP : 200.1.10.100 (128 - 72, 64) Class C
NID : 200.1.10.0
DBA : 200.1.10.255

# Networking

## Subnetting

(I) Big network maintenance problem
(II) Security issues
(III) Loss of ip addresses

Internal router

Subnet mask : 1's : NID + SID part
0s : HID part

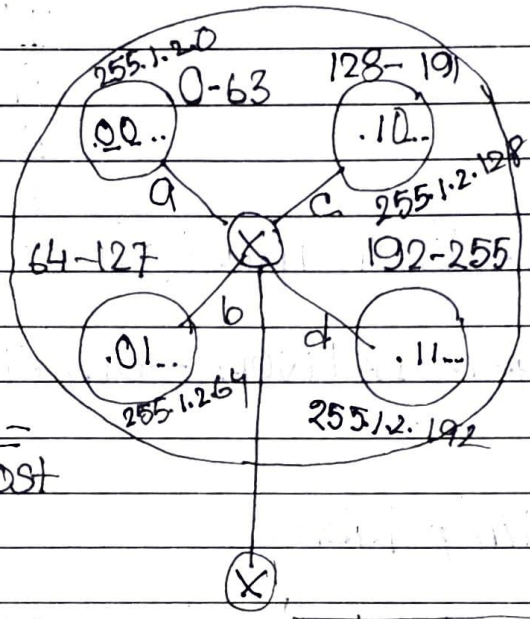
255.1.2.0

Subnet

[8].[8].[8].--- ------

Subnet ↓1 Host 0

255.255.255.192
↳ subnet mask




[m | SA | DA]

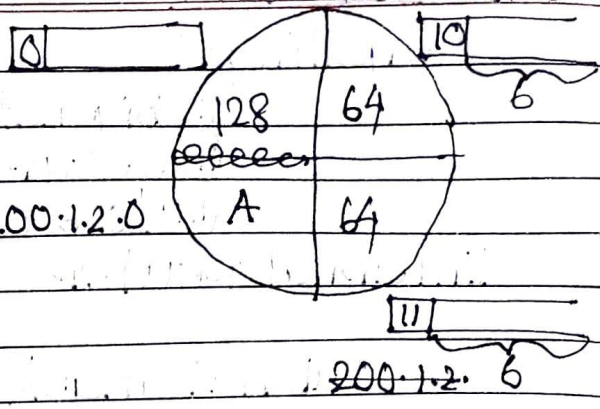
DA : 200.1.2.230

Routing table:

| NID | SM | Interface |
|---|---|---|
| 200.1.2.0 | 255.255.255.192 | a |
| 200.1.2.64 | 〃 | b |
| 200.1.2.128 | 〃 | c |
| 200.1.2.192 | 〃 | d |
| 0.0.0.0 | 0.0.0.0 | e |

Fixed length subnet masking
equal partitioning

# Variable length subnet masking (VLSM)




A: Subnet mask
255.255.255.128

B, C: Subnet mask
255.255.255.192

If more than one match in the routing table, send the packet to the subnet with
more 1s in mask ← longer subnet mask.

Bigger network → smaller subnet mask

Routing table

| NID | SM | Interface |
|---|---|---|
| 200.1.2.0 | 255.255.255.128 | A |
| 200.1.2.128 | 255.255.255.192 | B |
| 200.1.2.192 | " | C |
| 0.0.0.0 | 0.0.0.0 | e |

IANA - Internet assigned numbers authority → IP addresses

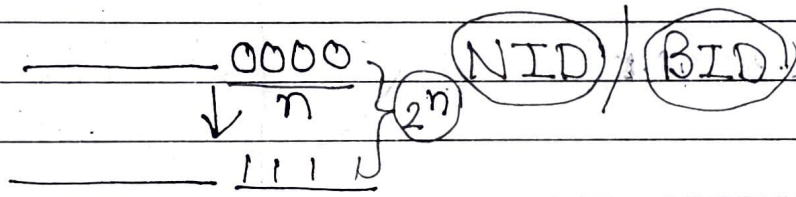
Classless : (CIDR) Classless Inter Domain Routine

(Block)

a.b.c.d/(n) → no of bits used in the NID part

Rules :

(i) contiguous
(ii) block size must be power of 2
(iii) FIRST IP address in the block must be divisible by size of the block.




BID ← 20.10.30.32
:
DBA ← 20.10.30.63
} 32 = $2^5$

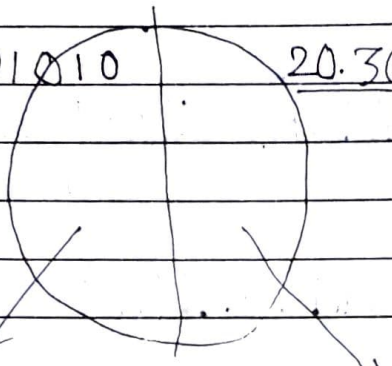
Representation :
20.10.30.32/27

20.10.30.001 00000

Any number in the range

# Subnetting in CIDR : Classless Inter Domain Routing

20.30.40.10 / 25



20.30.40.00001010
NID 0

20.30.40.00001010
NID 1

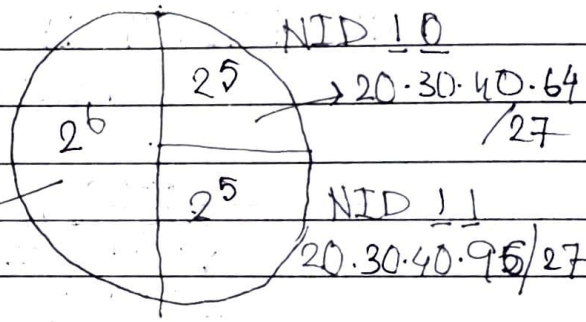
20.30.40.0/26

20.30.40.64/26

Subnet mask

Subnet mask

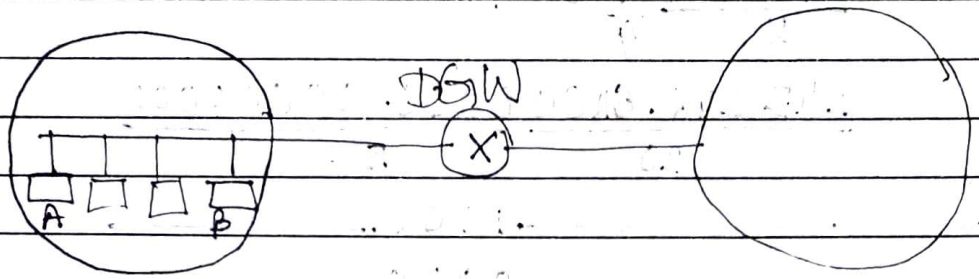
20.30.40.10/25
(27)




20.30.40.0000 1010
NID

0100 0000
(64)

# Networking




Subnet mask - used to determine destination B - same network or not

Default gateway - default router it is connected to

## Supernetting / Aggregation:

Too much subnetting
routing table
increasing exponentially

F ① Illusion that same network

Rules

(i) contiguous
(ii) size same for all networks
(iii) First ~~add~~ NID must be divisible by total block size (sum of the sizes)

→ supernet id
→ supernet mask — Fixed part 1
Variable part 0

100.1.2.0 / 25
100.1.2.128 / 26 } supernet
100.1.2.192 / 26

245. 248. 128. 0 / 20

16 4

245. 248. 1000 0000. 00000000

20 12

(1/2)

1000.

0000.

0100

0000

One way

~~1001000~~

A: 10001000 → (128+8) = .136. / 21

B: 10000100 (128+4) = .132 / 22

OR 10000000 128 128 / 22

Date: / / Page

# Delays in CN:

## Transmission delay ($T_t$)

(Host to channel)

$$T_t = \frac{L}{B}$$

$L$ → Length of the packet

$B$ → Bandwidth

Data: 1 KB = 1024 b

BW : 1 Kbps = 1000 bps

$$T_t = \frac{1024}{1000} = 1.024$$

## Propagation delay:

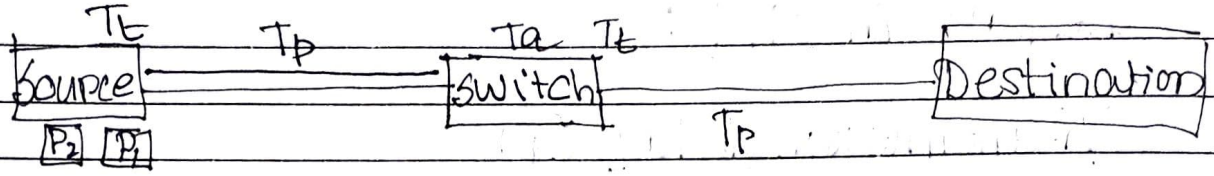
Last bit travel time to receiver

$$T_p = \frac{d}{v}$$

$d$ → length of the channel

$v$ → velocity of transmission

**Queing delay**: Time spent by the a data packet in receiver buffer.




$P_1$ = 5000 bits

$P_2$ = 5000 bits

BW = $(5000/10^7)$ bits/sec

$T_p = 20 \times 10^{-6}$ s. = 20 μs

$d = \left(\frac{5000}{10^7} \times \frac{20}{10^6}\right)$ bits

$T_t = \left(\frac{5000}{5000} \times 10^7\right)$ sec $= 10^7$ sec

$T_t = (5000 \times 10^{-7})$ sec = 500 μs

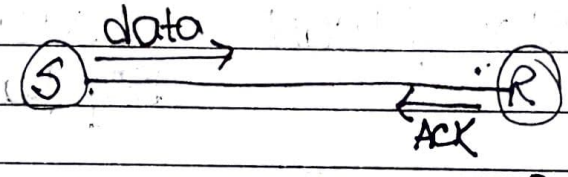
1 packet: $2(T_t + T_p) + T_a$

$T_1$ : 0

$T_2$ : 500 μs

Date 8/7 Page

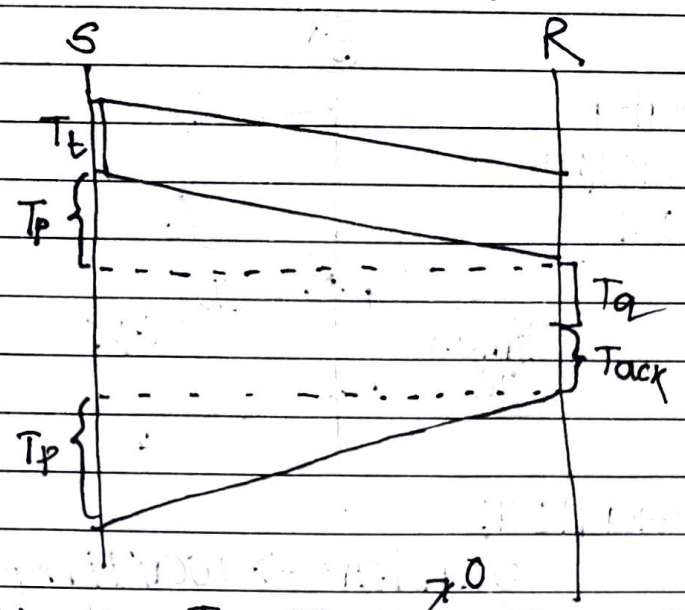
# Flow Control




Simplest flow control method

↓

Stop and wait



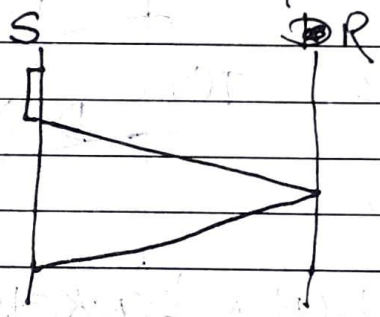

Round Trip Time (RTT) $= 2T_p$

Total time $= T_t + T_p + \cancel{T_q}^{\to 0} + T_{ack} + T_p$

$T_t$ ↓ data, $T_{ack}$ ↓ ack

ACK packet is quite small.

$= T_t + 2T_p$




$\eta$ = useful time / total cycle time

$= \frac{T_t}{T_t + 2T_p} = \frac{1}{1+2a}$

Throughput = bits per second

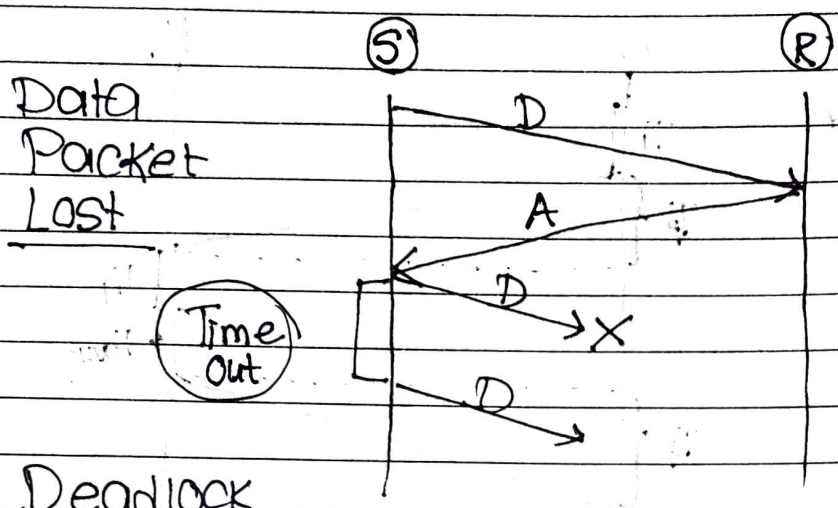
Effective b/w

$= \frac{L}{T_t + 2T_p}$

$= \frac{L/B \times B}{T_t + 2T_p} = \frac{T_t}{T_t + 2T_p} \times B$ (Bandwidth)

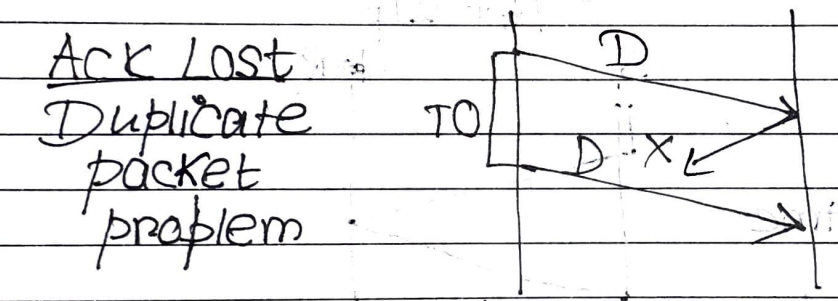
$= \eta \times B$

Stop & Wait  LAN ✓  WAN ✗
Big packet ✓

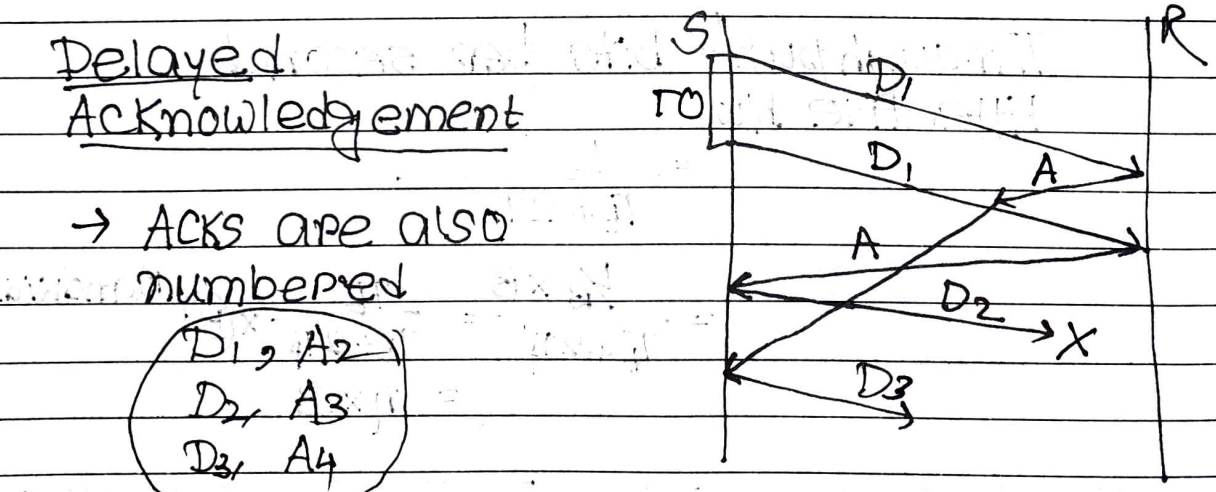



Deadlock
Sender → waiting for ACK.
Receiver → waiting for data.

↳ Time out timer
↳ ARQ (Auto Repeat Request)

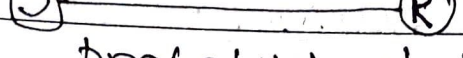



Data packets are numbered



8-






probability of losing a packet = $p$

Total $n$ packets sent

Total no of transmission req

$$n + np + np^2 + np^3 + \cdots$$

(lost, lost, lost)

$$= n\left(\frac{1}{1-p}\right)$$

$n = 400 \quad p = 0.2$

$$400 \times \frac{1}{0.8} = 500$$

Q 2015

$$\eta = \frac{1}{1+2\left(\frac{T_p}{T_t}\right)} \geq 50\% \; 0.5$$

$$\frac{1}{1+2a} \geq \frac{1}{2}$$

$$1+2a \leq 2$$

$$a \leq 1/2$$

$$\frac{T_p}{T_t} \leq 1/2$$

$$\frac{T_p \times B}{L} \leq 1/2$$

$$L \geq 2 \times T_p \times B$$

$$2 \times 20 \times 10^{-3} \times 64 \times 10^{3}$$

$$\frac{2 \times 20 \times 64}{8} \text{ Byte}$$

$$320A$$

## Capacity of channel :

Capacity = BW × $T_p$ [total no of bits put in the channe

↓

Half duplex

Full duplex, capacity = 2 × BW × $T_p$

## Pipeline :

Sliding window protocol

Maximum utilization : Sender side buffer size = (1+2a)

min no of seq no required = $\lceil \log_2(1+2a) \rceil$

$\eta = \left(\frac{64}{100}\right)$

~~No of bits~~

Date: 9/7/2020 Page

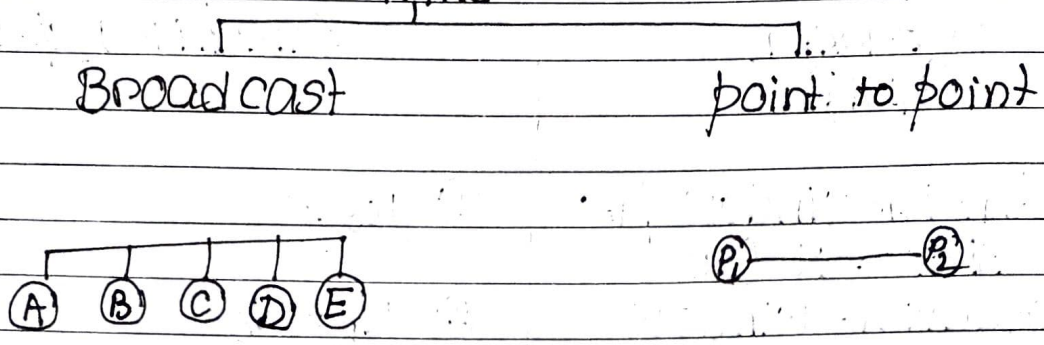
## Networking

### Sliding window protocol implementation

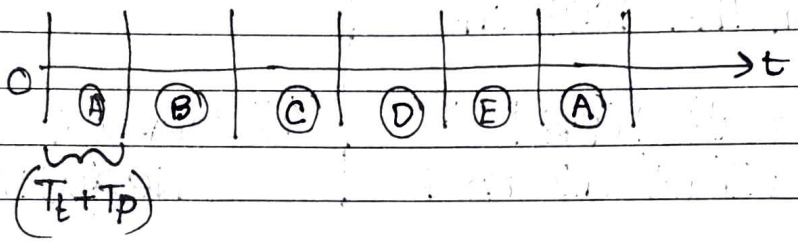
| Go Back n (GBN) | Selective Repeat |
|---|---|
| ① Sender window size = N (>1) | ① $W_S > 1$ |
| ② Receiver window size = 1 | ② $W_R = W_S$ |
| ③ Cumulative ACK | ③ Discrete ACK Retransmit only the lost packets |
| Problem ① One packet lost ↳ go back and retransmit all from the lost one | |
| ④ Corrupt packet → silent discard | ④ Corrupt package → Negative ACK (NAK) |

| | Stop n Wait | GBN | SR |
|---|---|---|---|
| Efficiency | $\frac{1}{1+2a}$ | $\frac{N}{1+2a}$ | $\frac{N}{1+2a}$ |
| Buffer | S① + ①R | S(N) + ①R | S(N) + (N)R |
| Seq no | (1+1) = 2 | N+1 | N+N |
| Retransmit (1 packet lost) | 1 | N | 1 |
| Bandwidth | low | high | medium |
| CPU | low | moderate | high |
| Implementation | simple | moderate | complex |

## Access control methods:




### ① Time Division Multiplexing: (TDM)



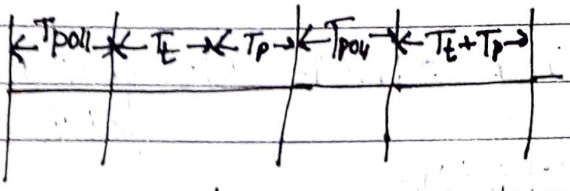

$T_t$ may vary for each station

$$\eta = \frac{\text{useful time}}{\text{cycle time}} = \frac{T_t}{T_t + T_p} = \frac{1}{1+a}$$

Disadv ↘ A station may be idle during its alloted t slot.

## ② Polling :

Ask who has wants to send ✓




$$\eta = \frac{\text{useful time}}{\text{cycle time}} = \frac{T_t}{T_p + T_t + T_p}$$

$EBW = \eta \times BW$ (Effective)

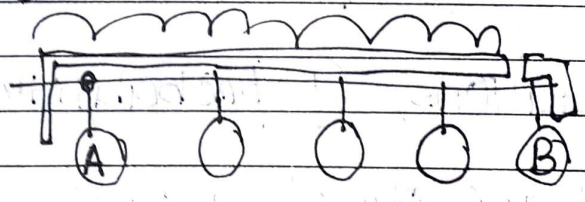
Disadv : Starvation

## ③ CSMA / CD

CSMA → Carrier sensing, multiple access

CD → collision detection

~~No~~ (No ACK)




① Sense if the carrier is busy before transmission

② Can be sure that collision if collision signal comes back whie A is still transmitting (collision signal)

Worst case

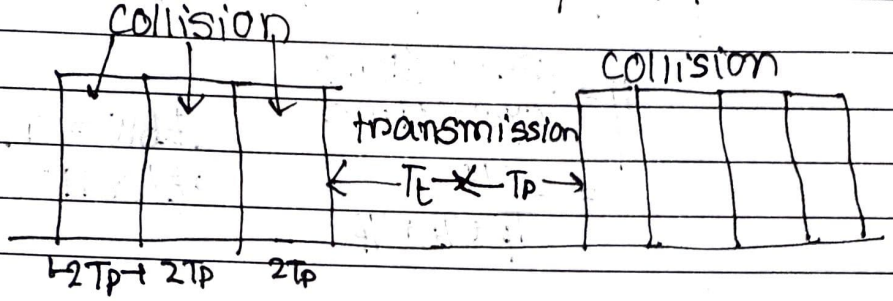
$T_t \geq 2T_p$ [condition for collision detection]

Date: / / Page

$\Rightarrow L/B \geq 2T_p$

$\Rightarrow \boxed{L \geq 2T_pB}$

Padding → increase data size packet




Cycle = collision + $T_t$ + $T_p$

(c) no of

$$\eta = \frac{T_t}{C \times 2 \times T_p + T_t + T_p}$$

C = ?

① n stations, (p) probability of transmission

② probability of success

$$P_{success} = {}^nC_1 \times p \times (1-p)^{n-1}$$

transmit → does not transmit

$$\frac{dP_{succ}}{dp}, \quad \frac{d^2P_{succ}}{dp^2} < 0$$

$p = 1/n$ → max success probability

$$P_{max} = \left(1 - \frac{1}{n}\right)^n \quad \underset{n \to \infty}{Lt} P_{max} = \frac{1}{e}$$

(maximum prob of succ

## Networking

### CDMA/CD

contd

③ No of tries before the first success (Poison distribution)

$$\frac{1}{P_{max}} = e$$

$$\eta = \frac{T_t}{e \cdot 2 \cdot T_p + T_t + T_p}$$

$$= \frac{1}{1 + 6.44\,a}$$

$$a = \frac{T_p}{T_t}$$

LAN ✓ WAN ✗

Large packet ✓

Too large packet → monopolization

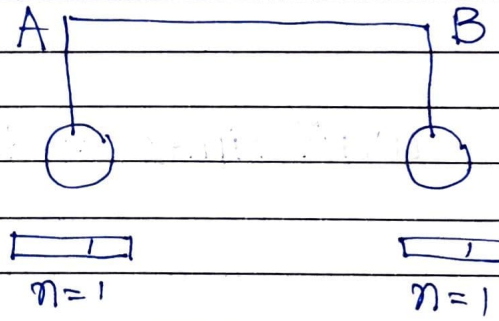
### Back off Algorithm:

waiting time

collision no of the data packet (n)

$[0, 2^n - 1]$

Random no




$P(Win_A) = 1/4$

$P(Win_B) = 1/4$

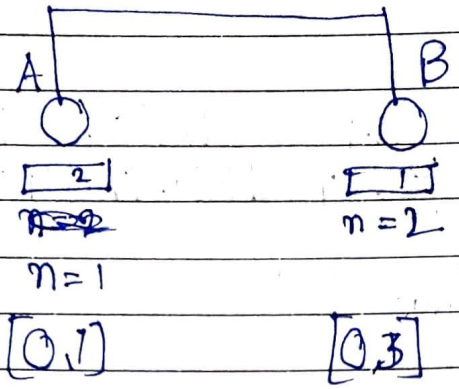
$P(collision) = 2/4 = 1/2$

[0,1]

| A | B | A's wait time |
|---|---|---|
| 0 | 0 | $= 0 \times T_{slot}$ |
| 0 | 1 | B's wait time. |
| 1 | 0 | $= 1 \times T_{slot}$ |
| 1 | 1 | |

(0,1) → A won.

Date: 18/7 Page 186




| A | B | |
|---|---|---|
| 0 | 0 - C | A's winning chance = 5/8 |
| 0 | 1 - A | |
| 0 | 2 - A | B's " " = 1/8 |
| 0 | 3 - A | |
| 1 | 0 - B | P(collision) = 2/8 |
| 1 | 1 - C | |
| 1 | 2 - A | |
| 1 | 3 - A | |

If a channel wins, its wining probability in next transmission ~~becom~~ increases exponentially

capture effect

wait time = K x Tslot

$K \in [0, 2^n - 1]$

n = collision no of the data packet to be sent

Applicable only for two stations.

Binary exponential backoff algorithm

Date: / / Page

## Token passing:

$x$ bit time = time taken to transmit $x$ bits

$x$ bittime = $\left(\frac{x}{BW}\right)$ secs

meters $\xrightarrow{(v)}$ sec

→ Ring Topology

→ unidirectional
→ station holding the token transmits the data

Total time taken by a bit to start from a point and come back to the same place : Ring Latency

$$= \left[\left(\frac{d}{v}\right) + \left(\frac{N \times b}{BW}\right)\right]$$

$\left(\frac{d}{v}\right) = T_p$

holding time in a station

Cycle time : Time taken by a token to start from a point and come back

$\left(\frac{d}{v} + N \times THT\right)$

Token Holding Time : (THT)

$\eta = \frac{\text{useful time}}{\text{cycle time}} = \frac{N \times T_t}{T_p + N \times THT}$

→ every station sends one packet

# Token passing

- Delayed Token Reinsertion (DTR)
- Early token reinsertion (ETR)

## ① DTR:

(i) Hold the token
(ii) Transmit data packet
(iii) wait till the transmitted packet complete one round
(iv) Release the token

$$THT = T_t + RL$$

$$= T_t + T_p + \underbrace{N \times b}_{=0 \text{ assuming}}$$

$$\eta = \frac{N \times T_t}{T_p + N \times THT}$$

$$= \frac{N \times T_t}{T_p + N \times (T_t + T_p)}$$

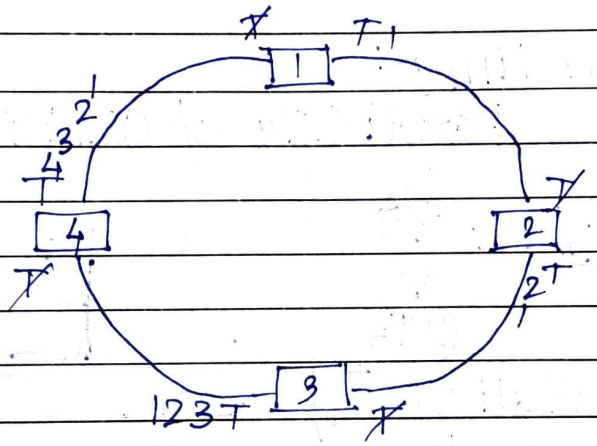
$$= \frac{1}{1 + \left(\frac{N+1}{N}\right) a} \quad \left(a = \frac{T_p}{T_t}\right)$$

# Networking

## ETR :

1. Take the token
2. Transmit data packet
3. Release the token

Station must remove its data packet after it comes back completing one cycle.




$THT = T_t$

$$\eta = \frac{N \times T_t}{T_p + N \times T_t}$$

$$= \frac{1}{1 + \frac{a}{N}} \quad \left(a = \frac{T_p}{T_t}\right)$$

# ALOHA:

1. Transmit any time
2. No carrier sensing
3. ACK present
4. Retransmission
5. Backoff

① Pure aloha : ② Slotted aloha

## Vulnerable time:

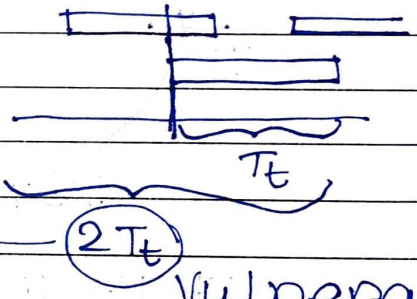
Possibility of collision

## Pure aloha:

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

$2T_t$

no of station that wants to transmit in $T_t$



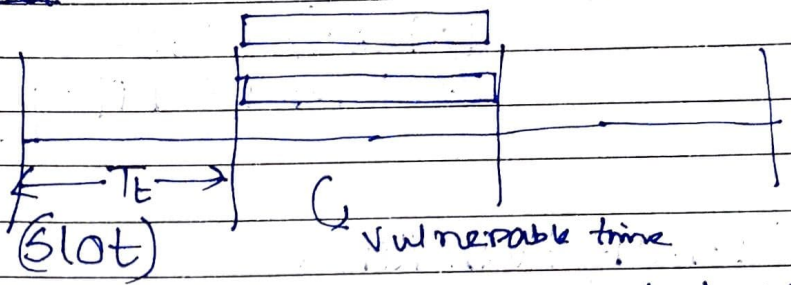

Vulnerable time

$$\frac{d\eta}{dG} = 0 \Rightarrow G = 1/2$$ max efficiency

So, if 1 station wants to transmit in $2T_t$ time, then max eff achieved

$$\eta = \frac{1}{2} e^{-1} = \frac{1}{2e} = 0.184$$

## Slotted aloha:




If a station misses one slot, it has to start in next slot.

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

$$\frac{d\eta}{dG} = 0 \Rightarrow G = 1$$

$$\eta = \frac{1}{e} = 0.368$$

Flow control: Target is: receiver's buffer is not overloaded

Access control: No collision. Sender gets the access of the channel.

## Error Control :

① Packet loss
② Corrupt bit ⊢ Burst error
└ Single bit

## Error Handling

| Error Detection | Error Correction |
|---|---|
| Retransmi-ssion | ① Hamming code |
| ① Send a copy. (D+D) | |

② Pairity checking : Even 0's/1's / Odd
③ CRC
④ Checksum

## ① CRC (Cyclic Redundancy Check)

(32 bit polynomial)

D: 11010
CPCGP: $x^3+x+1 \Rightarrow 1011$

```
      11010 000
1011 )11010 000 (
     1011
     -----
     01100
      1011
      ----
      1110
      1011
      ----
      1010
      1011
      ----
      0010
```

Receiver
11010 010

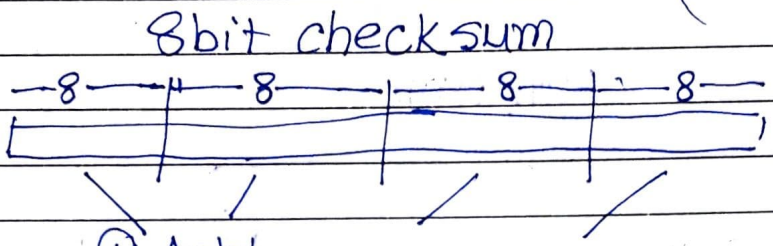
GATE

Networking

Date: 18/7/2020 Page 193

## Checksum :

(Read from book)




① Add.

② 1's complement of the addition resu

wrap around ⇒ avoid * overflow

Meaningful error – cannot be detected

# ISO/OSI Layer:

Communication b/w two process

Function

| mendatory | Optional |
|---|---|
| ① Error control | ① Encryption |
| ② Flow control | ② Checkpoint |
| ③ Access control | ③ Routing |
| ④ Multiplexing & Demultiplexing | ④ |
| ⑤ Addressing | |

Model of implementation
① ISO-OSI
② TCP/IP

## ISO-OSI model:

Application
Presentation
Session
Transport
Network
Data Link
Physical

Layering Adv ① Divide & conquer
② Encryption
③ Abstraction
④ Testing - easy

## ① Physical Layer

Hardware

Encoding
- ① manchester
- ② Differential "

## ② Data Link Layer :

① Flow control
② Error control: CRC
③ Framing
④ Physical addressing
⑤ Access control

### Framing :

- Fixed length
- Variable length
  - source and destination address
  - SFD Start Frame Demiliter
  - Ending : ① length
    ② End delimiter ED
      - A part of data may match ED
      - Byte/ Character Stuffing
      - Bit Stuffing

Why am I feeling so hungry

Date: / / Page

## Byte Stuffing / Character :

Escape character in the data part if it matches with ED

## Bit Stuffing :

ED : 01111

(Data part) ↓0

01011101

Data Sender

01111 → 011110

01101 → 011001

## Physical Addressing :

↓

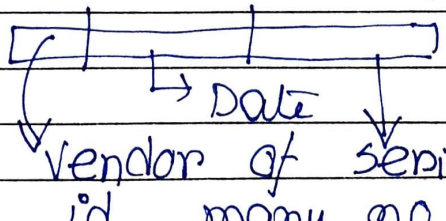
unique only in a network

Logical address : unique in entire (Network layer) : world wide web

Globally unique

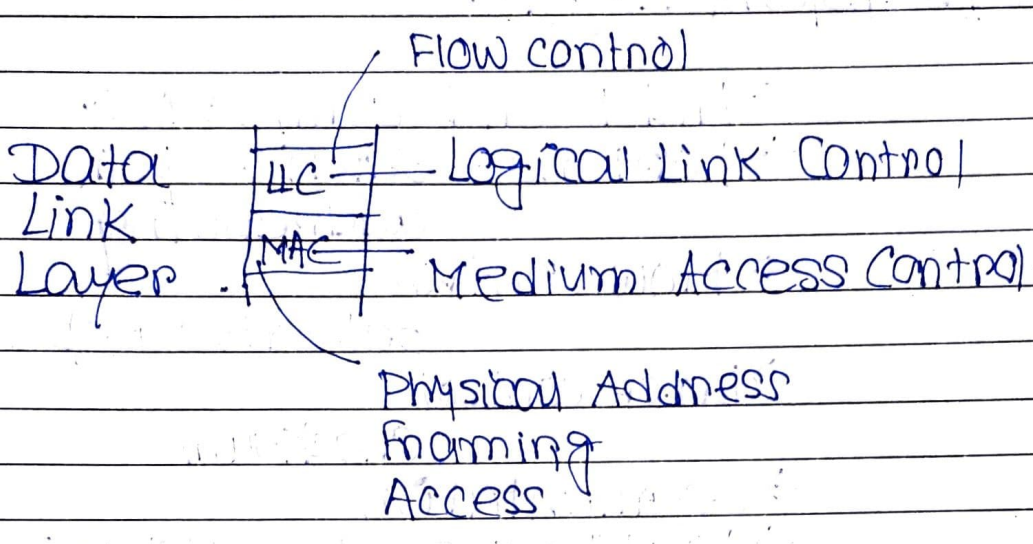
MAC
48 bit
Physical
NIC - ROM

IP
32 bit, Logical



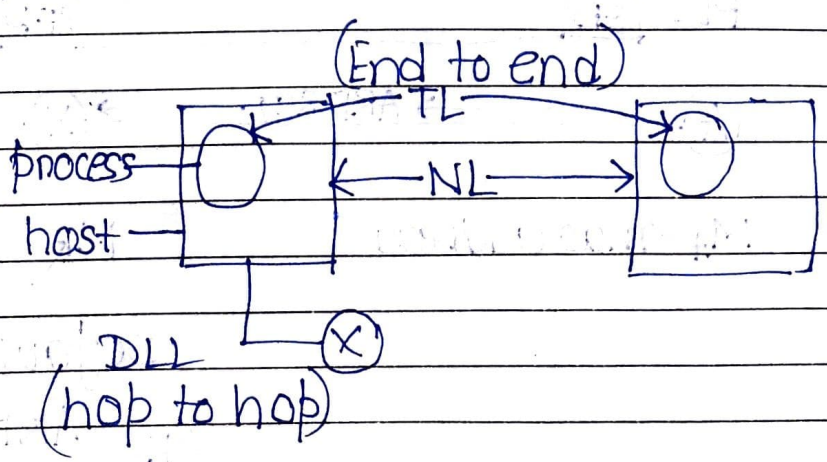
Date: 18/7 Page

# Networking



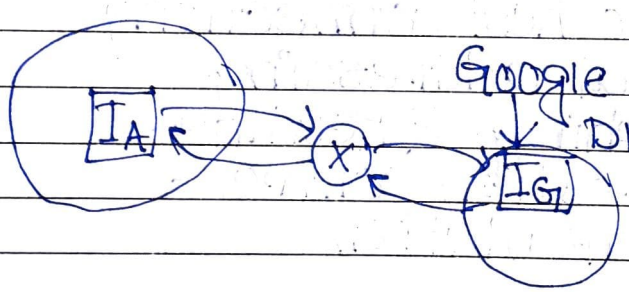

## ③ Network Layer

→ Host to host connectivity
→ Logical addressing
→ Switching & Routing
→ congestion control
→ Fragmentation



## ④ Transport Layer:

→ End to end connectivity
→ Port no - service point addressing
→ Flow control — Selective Repeat
→ Error control — Checksum
→ Segmentation
→ Multiplexing - demultiplexing
→ congestion control
→ Reliable (if required)




AL m → source port
TL [m | x | 80] → dest " (well Known)
NL [m | x | 80 | $I_A$ | $I_G$] → DNS
DLL [m | x | 80 | $I_A$ | $I_G$ | $M_A$ | $M_R$] → ARP
PL [EB ... SFD]

A → ARP Req → X → Google

$M_A$ - mac address

$M_R$

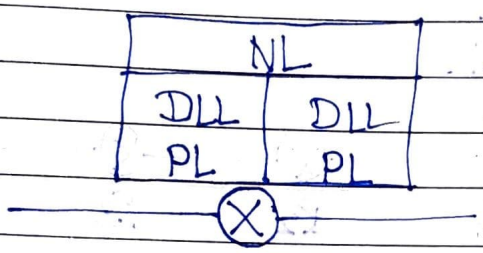
Default Gateway Router

$I_R$ ARP $M_R$

DLL → Hop to hop acknowledgement.

TL → ACK depending on the protocol

Flow control ~~b/w~~ in both DLL & TL




Both side networks can be of different type.

⑤ Session Layer

→ Authentication and authorization - username, password

→ Checkpointing

→ Synchronization

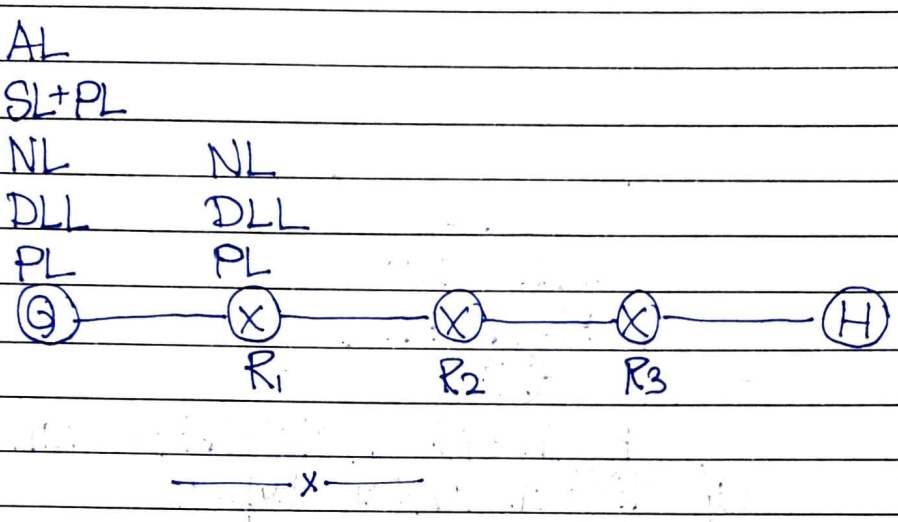
→ Dialog control

→ Logical grouping ◎ (Atomicity)

Date: / / Page

## ⑥ Presentation:

→ Character translation
→ encryption & decryption
→ compression




## Ethernet:

(IEEE 802.3)

### Traditional

(I) Topology: Bus
(II) Access control: CSMA/CD
(III) No ack
(IV) 10 mbps, 100 mbps (FAST), 1GBY
(V) Encoding: Manchester

# Networking

AL message
TL segment
NL datagram
DLL frame
PL Single protocol data unit (I-PDU)

## Ethernet Frame Format (802.3)

| | | |
|---|---|---|
| PL Header | Preamble | 7 Byte: 10101010 ... 10 > Sync |
| | SFD | 1 Byte: 10101011 > Frame start |
| DLL Header | DA | 6 " : mac |
| | SA | 6 " : mac |
| | Length | 2 " : ~~fram~~ data length |
| | Data | |
| (DLL Tail) | CRC | 4 " : 32 bit CRC |

max data : 1500B
max frame : (1500+18) (1518) B

$L \geq 2 \times T_p \times BW$
$L \geq 64B$ (min Frame size)
Data $\geq 46B$ (DA — CRC)

### MAC

① unicast
LSB of 1st byte = 0

② multicast
LSB of 1st byte = 1
Create a group of the receivers and assign the group with a multicast address

③ broadcast
All bits : 1
48 1s
FF:FF:FF:FF:FF:FF

Date: / / Page 20

Disadv

(i) Real time works not possible.
Due to collision

(ii) Max-min limitation
Chat/interactive

(iii) no priority
Client-server application

To solve these problems: Token ring was proposed

802.4 : Token bus
802.5 : Token ring

Token ring:

→ Ring topology
→ Access control: Token passing
→ unidirectional
→ Datarate/bandwidth: 4 mbps, 16 mbps
→ Ack : piggybacking
→ Encoding : Differential manchester

ETR/DTR

→ Sender pulls out its sent packet after full cycle

Source Problem
→ Orphan packet problem : Sender is down
→ Stray packet : Sent data so corrupted the sender fails to recognize it

## Solution

① monitor station: put a mark when sees a packet. Pulls out next time.
(monitor bit) ✓ orphan packet

② CRC. Delete if not 0.
stray packet ✓

## Destination Problem:

① Down — Resend ✗
② Busy — Resend ✓
③ Error data — Cannot & Retransmit ✓

| 3 ~~Two~~ bits: | Available | Copied | Error |
|---|---|---|---|
| Initial: | 0 | 0 | 0 |
| ~~OK~~ Okay: | 1 | 1 | 0 |
| (Resend) Busy: | 1 | 0 | 0 |
| (Cannot & Resend) | 1 | 0 | 1 |
| (✗Resend) Down: | 0 | 0 | 0 |

Retransmitting: monitor bit ← 0

## Token Problem:

① Monopolization
Captured token
⇒ Max THT.

② Token lost
⇒ monitor → calculate time
- min token return time: RL
- max " " " : (RL+NX mTHT)

after that generate token

(iii) Token corrupted
⇒ monitor does not recognize it
⇒ removes it as stray
⇒ after max token return time
↳ new token

Monitor Problem
→ If monitor down, everything gone
⇒ Heartbeat message
AMP frame
Active monitor
presence

↳ If monitor down
polling → new monitor

→ Monitor malfunctioning / hacked
⇒ Human intervension require

Adv
(i) Real time application (no collision
(ii) Interactive " (no limit on da
(iii) Client server " (priority ✓)

# Networking

## Token ring frame format:

(Differential manchester encoding)

**Token**

| | |
|---|---|
| SD | 1 Byte |
| AC | 1 " |
| ED | 1 " |

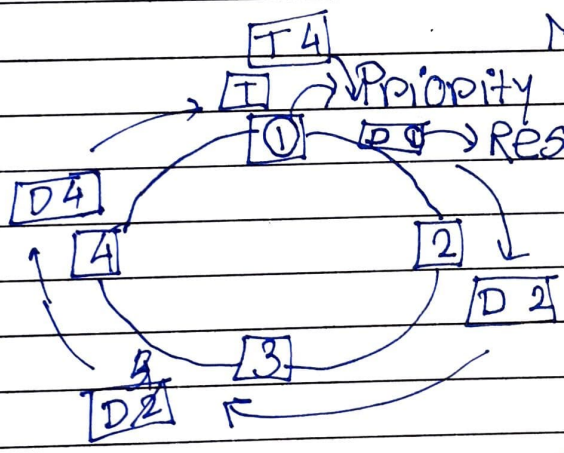
**Data frame**

| | Byte |
|---|---|
| Start delim : SD | 1 |
| Access Control : AC | 1 |

0 ⊓⊔ Represen
0 ⊔⊓
1 ⊔
1 ⊓

High Low
JK00JK00
PPPTMRRR
Priority, Reservation, Monitor

Token bit
T←1 : Token frame
T←0 Data frame
M←1 monitor has seen it




Delayed Token Reinsertation

| | | Byte |
|---|---|---|
| Frame control : FC | | 1 |
| Destination add : DA | | 6 |
| Source add : SA | | 6 |
| Mac address | | |
| CRC | : CRC | 4 |
| End delim | : ED | 1 |

Data 0 0 1 0 1 0 1 0
Control : 1 1
- AMP
- Polling
- Purge
- Beckon

JK11JKIE → Error Into

I ← 1 more info following (Data too large cannot be sent in MTHT)

E ← 1 Destination detects error

Frame status : FS A C 0 0 A C 0 0

↙ Available ↓ copy

Two copies: since CRC is not computed on this. & CRC is ~~use~~ computed on sender side. They cannot determine the ~~curr~~ correct status of A & C.

# Switching (Done in network layer)

- Circuit Switching (Not in network layer)
- Packet Switching
  - Virtual Circuit
  - Datagram
- Message switching

## Circuit Switching:

Telephone exchange
- Manual switching

- Applied at physical layer.

$$Time_{taken} = \text{Setup time} + T_t\left(\frac{M}{B}\right) + \left(\frac{d}{v}\right)T_p + \text{tear down time}$$

($M$: message size; $T_t$: transmission time; $d$: total length of net; $T_p$)

## Packet Switching:

Multiplexing – to determine which path to choose

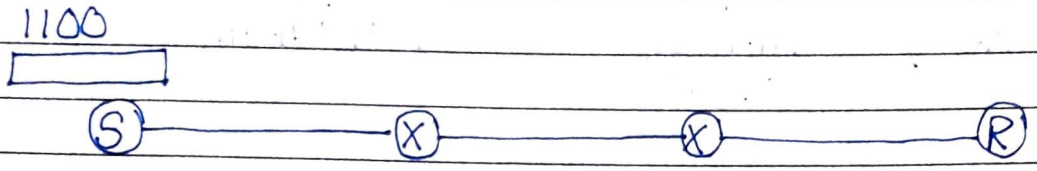
$$Time_{taken} = \left(x\,\frac{M}{B}\right) + \left(\frac{d}{v}\right)$$

($x$: No of switches; $\frac{M}{B}$: $T_t$; $\frac{d}{v}$: $T_p$)

Date: / / Page

## Pipelining in packet switching:

Data = 1000 Bytes
B/w = 1 mBps
= $10^6$ Bps
Header = 100 Bytes




$T_p = 0$

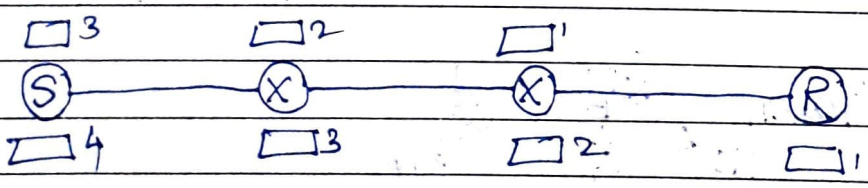
$T_t = \left(\frac{1100}{10^6}\right) = 1.1$ ms

Time taken = $(3 \times 1.1)$ ms = 3.3 ms

### Packetization

5 say. Packet size = $\left(\frac{1000}{5}\right) = 200$ Byte
+ Header
= (300 Byte) each packet

$T_t = \left(\frac{L}{B}\right) = \left(\frac{300}{10^6}\right) = 0.3$ ms




Time Taken = {
1st packet = $(3 \times T_t) = 3 \times 0.3 = 0.9$
2nd " = $T_t$
3 = $T_t$
4 = $T_t$
5 = $T_t$
} $4 \times T_t = 4 \times 0.3 = 1.2$

2.1 ms

# Networking

Say

# Packets = 10

Data = 1000/10 = 100

Header = 100

(Packet) = 200

$T_t = (200/10^6) = 0.2$ mSec

1.st Packet = $3 \times 0.2 = 0.6$

Next 9 " = $9 \times 0.2 = 1.8$

2.4 msec

Due to increased header overhead.

| Virtual Circuit (Reliable) | Datagram (Not reliable) |
|---|---|
| (i) First packet forms a path | (i) Each packet may take seperate path → out of order |
| (ii) Reserving buffer and memory | |
| (iii) All other packets take the same path → in order | |
| (iv) Charged: According to time & resoure reservation | Charged: based on data |
| → Packet I: Global header<br>Next: local " | → All packets require global header |
| → Connection oriented (Reservation) | → Connectionless (no reservation) |

# IP (Internet Protocol)

## IPv4

## Header

~~Version(4) HL(4) Type of Service(8) Total Length(16)~~

| Version(4) | HL(4) | Type of service (8) | Total length (16) | | | | 32 |
|---|---|---|---|---|---|---|---|
| Identification (16) | | | 0 | DF | MF | Fragment offset (13) | 32 |
| TTL(8) | | Protocol (8) | Header checksum (16) | | | | 32 |
| Source IP (32) | | | | | | | |
| Dest IP (32) | | | | | | | |
| Options (0-40 Byte) | | | | | | | |
| Data | | | | | | | |

HL : Header Length

~~3×5~~ (32×5) bits = 20 Bytes (static header)
+ (0-40) B
= (20-60) Byte

1111 = (15)

(scaling factor 4)

Header size = 30
HL = ⌈30/4⌉ = 8
(1000)

2 Byte ~~2bit~~ padding (00)

Version : 1 – 6
IPv4 → 0100

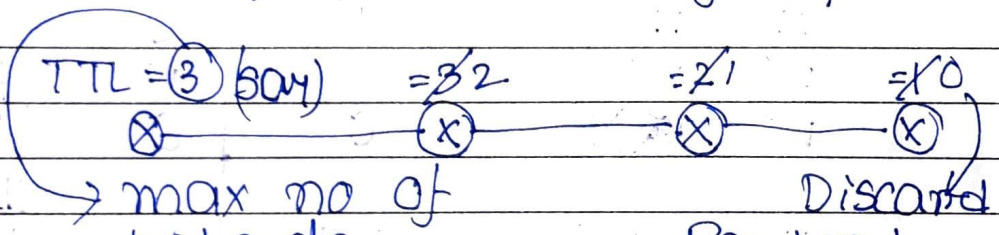
## Identification no:

Number every ~~packet~~ datagram going out of host.
Fragments – same identification no.

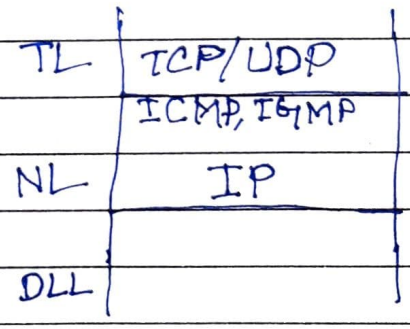
MF ← More fragment (1)
DF ← Do not fragment

Fragment offset ← No of data bytes ahead of this fragment in this datagram

TTL (8) Default router/entry
- Infinite loop problem
Restrict the no of hops




## Protocol:



## Header Checksum:

Only on header

It changes in every router (may)

- TTL
- Fragment-offset
- MF
- Options

## Source IP and Destination IP:

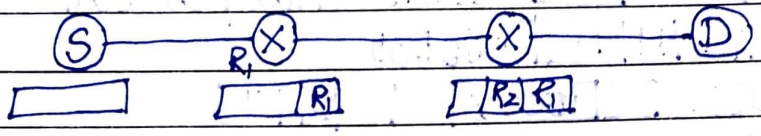
| Source | Destination | NID | HID | |
|---|---|---|---|---|
| ✓SIP, | ✓DIP | ✓ | ✓ | Valid id |
| X | X | ✓ | 0 | NID |
| X | ✓ | ✓ | 1 | DBA |
| X | ✓ | 1 | 1 | LBA |
| X | X | 1 | 0 | Subnet mask |
| ✓ | ✓ | 0 | ✓ | Host in a n/w |
| ✓ | X | 0 | 0 | (Asking for IP address) |
| X | ✓ | 127 | ✓ | Loop back address |

# Networking

## IP Header

### Options

#### (i) Record route

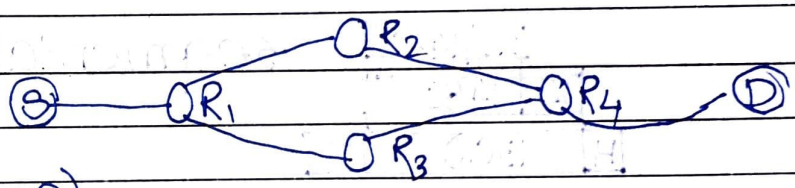



B/w two addresses : gap

Max 9 routers can be recorded.

#### (ii) Source Routing

Specify the path the packet has to take




(max 9)

→ Strict source routing
→ Loose source routing

#### (iii) Padding

Make header size ~~%4 = 4~~ = 0 (mod 4)

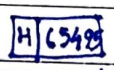
## Total Length:


16 bit

$(2^{16}-1) = 65,535$

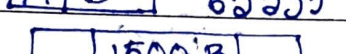
max size possible

mess AL
segm TL
datagram NL
frame DLL
PL



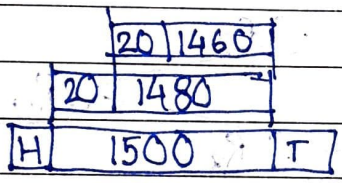

= 65515 (max payload)


= 65535




MTU

(AL) data size not limited
TL ← segmentation (65495 each, max)
NL ← fragmentation

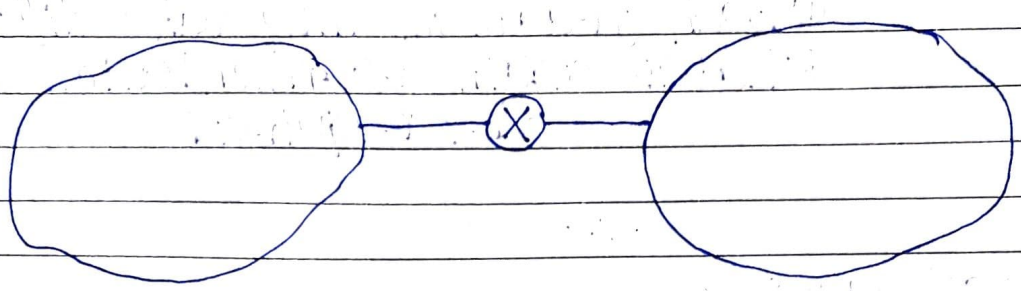



segmentation (source)

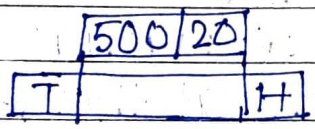
Source tries to avoid fragmentation

# Segmentation

Router



MTU = 520B




Id No = 100

MTU = 200B

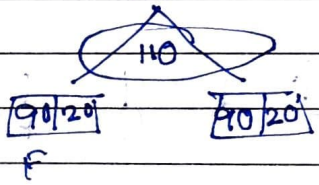
Id no = 100


Fragments

MF = more fragment

Fragment offset (13)
= no of data bytes ahead of it

| FO | FO | FO |
|---|---|---|
| 360 | 180 | 0 |




Data 46

So, scaling factor $(2^3) = 8$

Data size after fragmentation — Multiple of (8)
[Closest ~~no~~ multiple ≤ the allowed]

↳ Overhead at the network layer due to fragments ↓
each fragment has its own header

Date 21/7/20 Page

Reassemble done at destination

Not at routers:
① Each fragment may take different path
② Later again fragmentation may be required

—— x ——

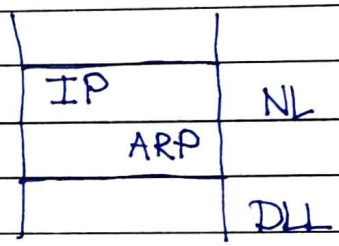
Broadcasting

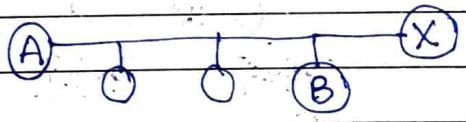
LBA IP: 255.255.255.255
MAC: FF:FF:FF:FF:FF:FF

Direct Broadcast: converted to LBA at the default router of dest network

Address Resolution Protocol:
IP → mac

ARP




A: | $I_B$ | ? | $MAC_A$ | FF:FF:FF... | ARP Request Broadcasting

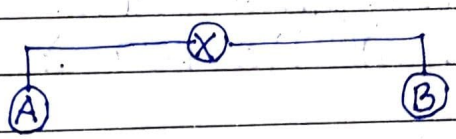
B: | $I_B$ | $m_B$ | $m_B$ | $m_A$ | ARP reply unicasting

# Networking

## Special Addressing

127 (Loop back address)

✗ 127.0.0.0
✗ 127.255.255.255



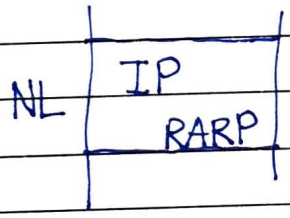

## Self connectivity:


ping 127.x.y.z

## RARP (Reverse ARP):

(mac → IP)

mac – ROM
IP – RAM




RARP server

| MAC | IP |
|---|---|
| $M_1$ | $I_1$ |
| $M_2$ | $I_2$ |

RARP Req packet (multicast)

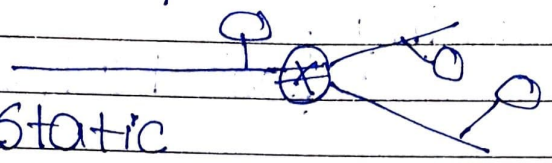
| $M_A$ | 0.0.0.0 | $M_A$ | FF:FF... |
|---|---|---|---|

RARP server (unicast)
↳ Reply packet

(Works at network layer AL, TL not needed)

Date: / / Page 218

Dis → Every network needs RARP

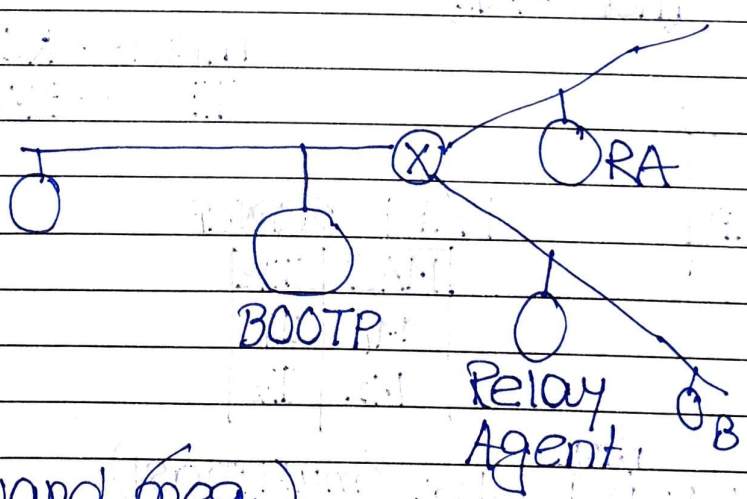


→ Static mapping table.
No of IPs req ≥ no of Host..

50 people day shift
50 " night "
(100 IP needed)

**Bootp:** (MAC → IP) (Bootstrap Protocol)
Application Layer

AL Rm
NL m x y
NL 0.0.0.0 255.255.255.255
DLL MA FF:FF
PL




**RA:** Forward (req_b) packet to BOOTP (unicast)

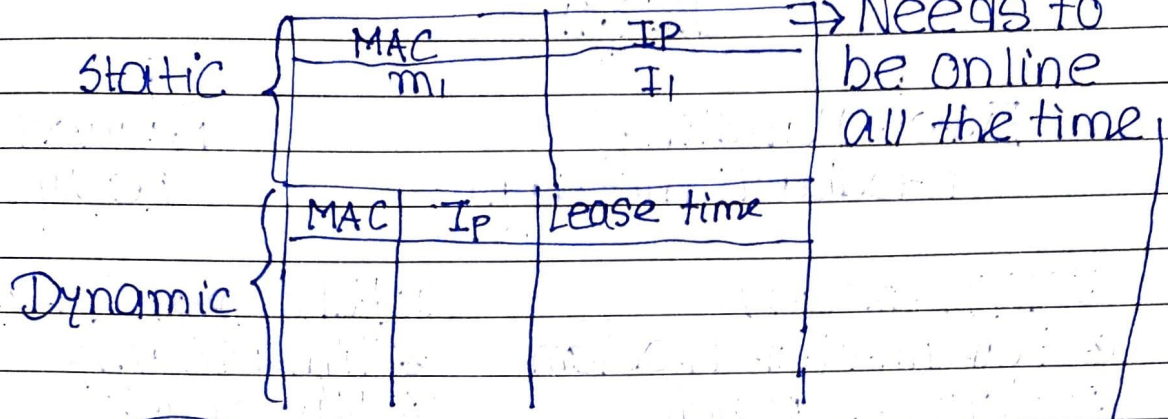
→ Stating mapping table

Date: / / Page

# DHCP :

Dynamic
Host
Configuration
Protocol

Relay agents




→ Needs to be online all the time

permanent ip assigned
→ mail server
→ web "
→ file "

Pool of IP address

Renew request after lease time

① Dynamic.
#Online hosts = req. # IP

DHCP backward compatible with BOOTP.
→ Same port no.

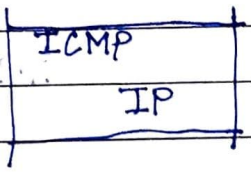
→ DHCP is implemented on router (sometimes) to avoid relay agents

Date: / / Page

## ICMP:

Internet Control Message Protocol

Network Layer




ICMP

- Error Handling / Feedback messaging
  1. TTL exceed
  2. Parameter problem
  3. Source quench
  4. Source redirect
  5. Destination unreachable
- Request & Reply
  1. Echo req & reply
  2. Time stamp
  3. N/w mask
  4. Router solicitation & advertizement

## Error Handling:

→ A ~~packe~~ IP packet discarded → ICMP generated
→ ICMP discarded (congestion) → no more ICMP

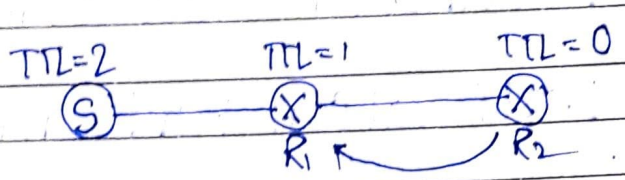
Date 21/7/1 Page

# Networking

## ICMP

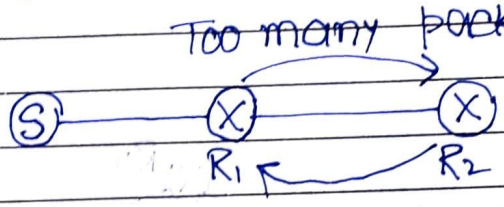
### Error Handling / Feedback messaging:

### TTL Exceed:




ICMP type = TTL exceed

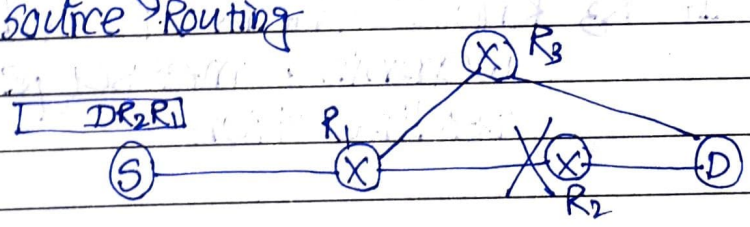
### Source quench:




ICMP packet = source quench

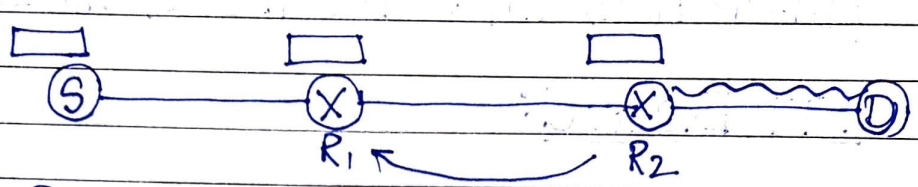
### Parameter problem:

(i) Strict SR
Source Routing




(ii) Corrupt header
CRC

## Destination Unreachable :

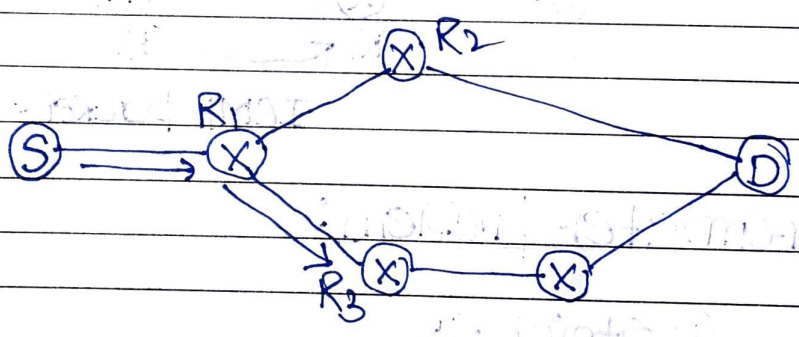



① Destination Host Unreachable

② Destination Port Unreachable

③ DF = 1
But MTU requires the packet to be fragmented

## Source Redirect




If $R_3$ knows there is a better path
warning message for
redirection

Date 21/7/1 Page

# Networking

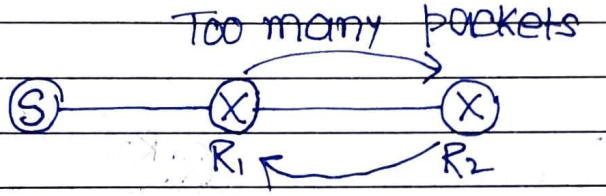
## ICMP

### Error Handling / Feedback messaging:

### TTL Exceed:


ICMP type = TTL exceed

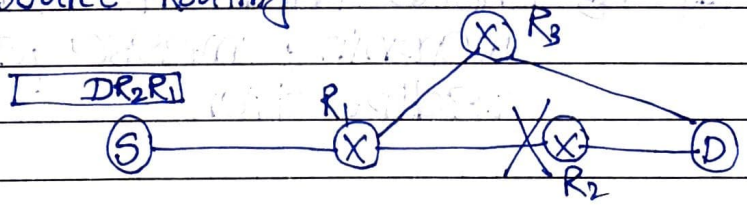
### Source quench:




ICMP packet = source quench

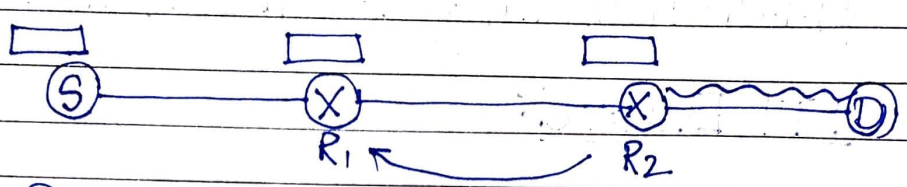
### Parameter problem:

(i) Strict SR (Source Routing)




(ii) Corrupt header
CRC

## Destination Unreachable :

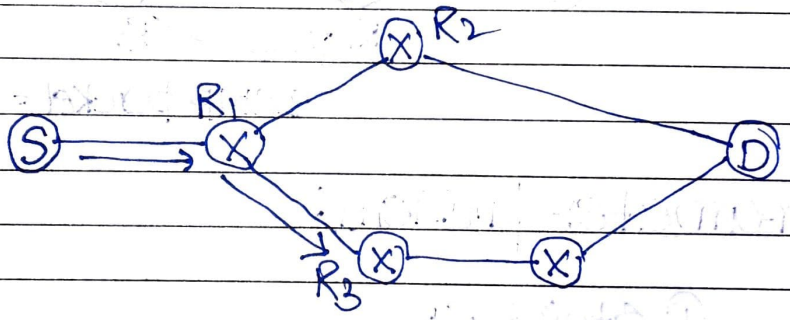



① Destination Host Unreachable

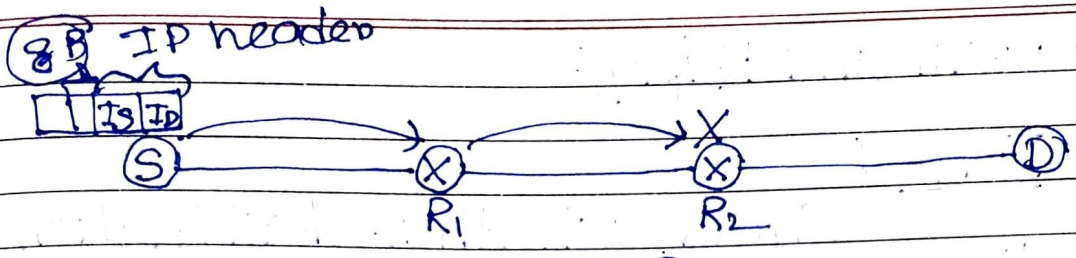
② Destination Port Unreachable

③ DF = 1
But MTU requires the packet to be fragmented

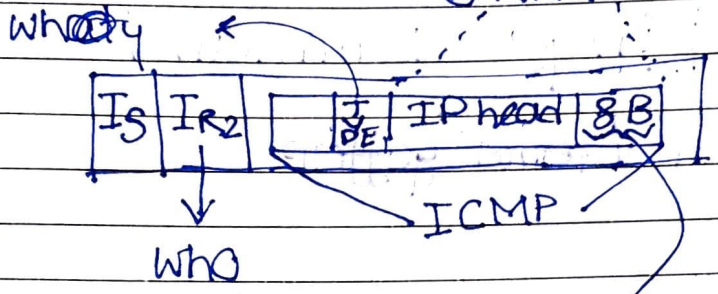
## Source Redirect




If $R_3$ knows there is a better path warning message for redirection




① Who
② Why
③ What




ICP (mostly used)
① Source Port
② Destination Port
③ Seq no

## Fragmentation

→ 1st fragment discarded → ICMP generate
Other " " → X

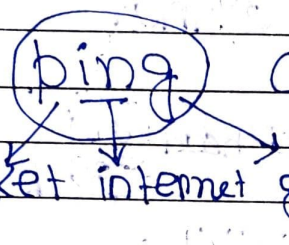
Date: / / Page

ICMP

## Request and Reply:

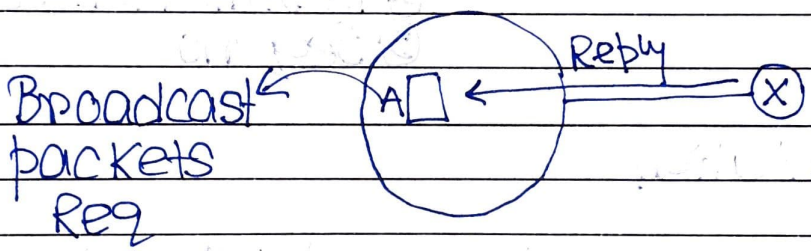
### ① Echo:

Check whether destination & the routers on the path are available or not




### ② Router solicitation:

Know the default router




### ③ Router Advertizement:

Router itself advertizes its availability

### ④ Subnet mask:

### ⑤ Time stamp:

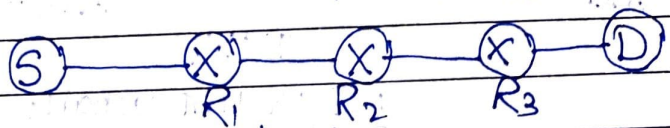
Sync time b/w two routers in different zones.

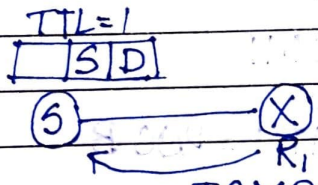
# Networking

## ICMP application

### Traceroute

Tracert www.google.com




TTL=2 $R_2$

TTL=3 $R_3$

TTL=4 Accept at Dest

no ICMP

↓ To get ICMP

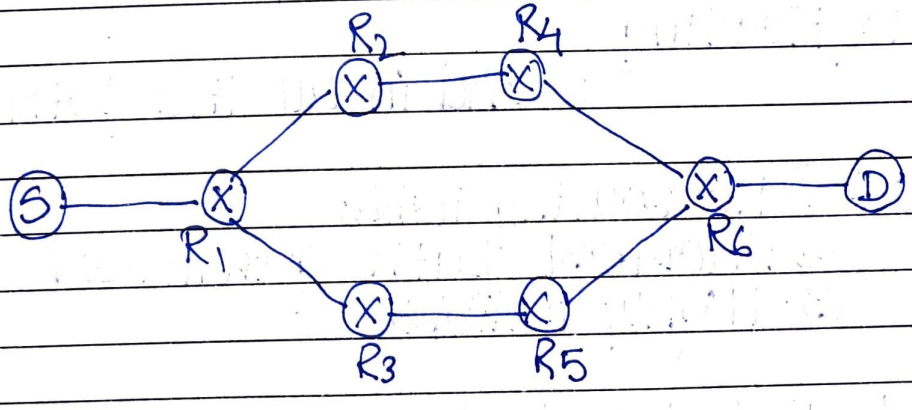
↓ from dest

**Record route option in IP**

→ All routers write their IP on the packet

→ Destination finds out the path taken

Dummy port no. → ICMP dest port unre

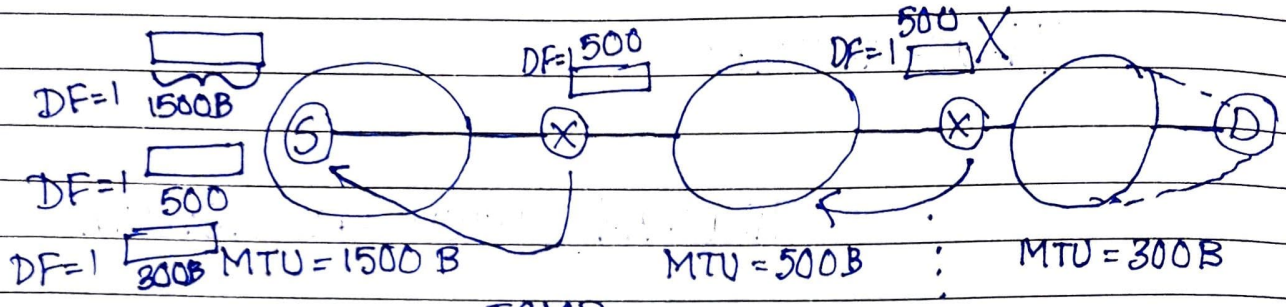



$R_1$ $R_2$ $R_5$ $R_6$ D

May not be a valid path.

Date: / / Page

# Path MTU Discovery: (PMTUD)
(ICMP application)

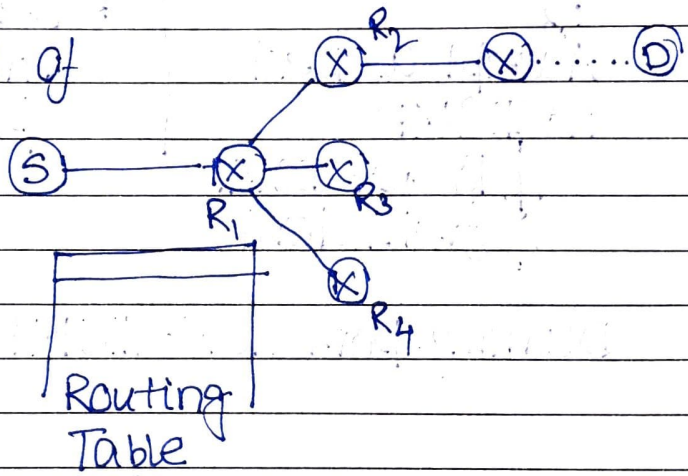



ICMP
1. Dest unreachable
2. MTU = 500

ICMP
1. DU
2. MTU = 300 B

—X—

Routing:
The method of preparing Routing Table




~~Ex~~ Flooding
Send in all the paths present

Adv: ① No routing table
② Shortest path guaranteed
③ Highly reliable

Dis: ① Duplicate packet
② High traffic

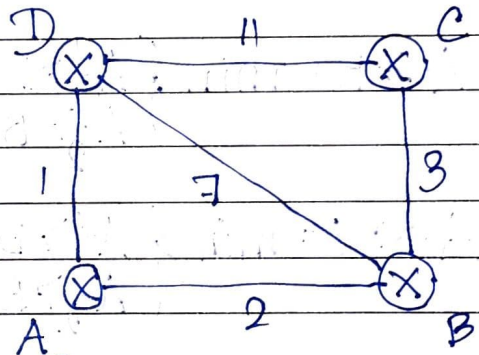
Date: / / Page

# Routing Algorithm

- Static
  - → Manual, offline (Not feasible)
- Dynamic
  - Distance Vector Routing (DVR)
  - Link State Routing (LSR)

## DVR:

Edge cost




Local Routing Table

| Dest | Dis | Next |
|---|---|---|
| A | 0 | A |
| B | 2 | B |
| C | ∝ | — |
| D | 1 | D |

| Dest | Dist | Next |
|---|---|---|
| A | 2 | A |
| B | 0 | B |
| C | 3 | C |
| D | 7 | D |

(Distance vector)

Exchange (Distance vectors) with neighbors

(At most 1 edge)

At A

DV from B & D

From B

| |
|---|
| 2 |
| 0 |
| 3 |
| 7 |

From D

| |
|---|
| 1 |
| 7 |
| 11 |
| 0 |

New Routing Table based on the received vectors

| | | |
|---|---|---|
| A | 0 | A |
| B | 2 | B |
| C | 5 | B |
| D | 1 | D |

$A \rightsquigarrow B = \min \begin{cases} A \xrightarrow{1} D + D \overset{7}{\rightsquigarrow} B \\ A \xrightarrow{2} B + B \overset{0}{\rightsquigarrow} B \end{cases}$

$A \rightsquigarrow C = \min \begin{cases} A \xrightarrow{1} D + D \overset{11}{\rightsquigarrow} C \\ A \xrightarrow{2} B + B \overset{3}{\rightsquigarrow} C \end{cases}$

$A \rightsquigarrow D = \min \begin{cases} A \xrightarrow{1} D + D \overset{0}{\rightsquigarrow} D \\ A \xrightarrow{2} B + B \overset{0}{\rightsquigarrow} B \end{cases}$

At most two edges

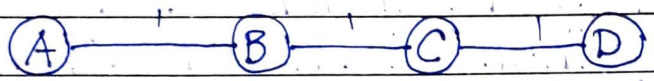
# Networking

## DVR (contd.)

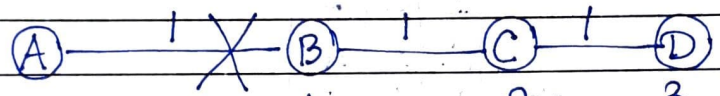
### Disadvantage:

### Count to infinity:

Bad news spreads slow
Good " " fast



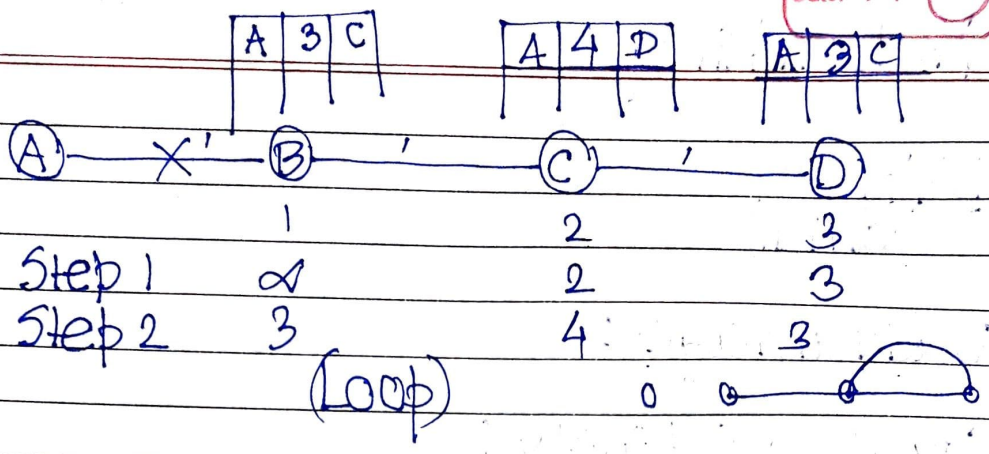

| | B | C | D |
|---|---|---|---|
| | ∞ | ∞ | ∞ |
| step1 | 1 | ∞ | ∞ |
| step2 | 1 | 2 | ∞ |
| step3 | 1 | 2 | 3 |




| | B | C | D |
|---|---|---|---|
| | 1 | 2 | 3 |
| step1 | ∞ | 2 | 3 |
| step2 | 3 | 2 | 3 |
| step3 | 5 | 4 | 5 |
| step4 | 5 | 6 | 5 |
| ⋮ | | | |
| | ∞ | ∞ | ∞ |

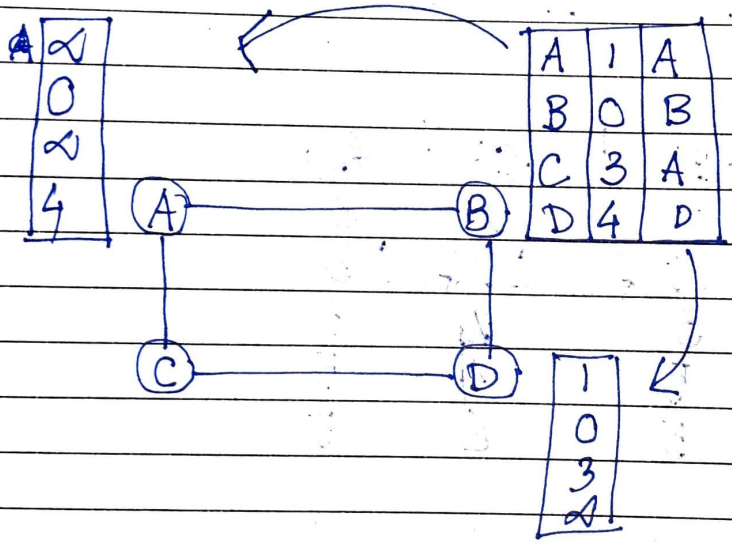
Solution: Split Horizon

Date: / / Page




## Split Horizon:

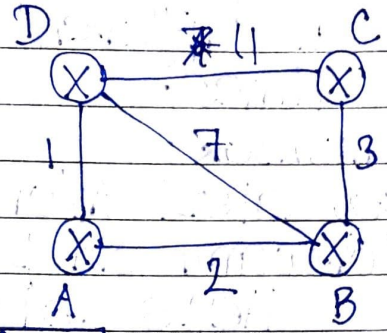
If C is dependent on B,
↳ then C should not help B.
↳ send the corresponding distance as ∞



Date: / / Page

# Link State Routing: (LSR)

(Higher b/w requirement)




| A | |
|---|---|
| | |
| | |
| B | 2 |
| D | 1 |

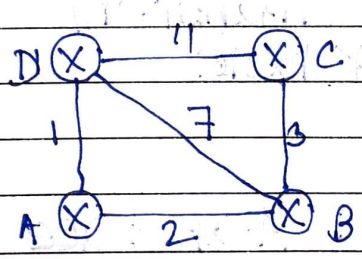
| B | |
|---|---|
| | |
| | |
| A | 2 |
| D | 7 |
| C | 3 |

(Link State Packet)

Flood the packets

Every node has info about each node (Global Knowledge)

At A




Routing table is generated by Dijkstra Algorithm

Date: / / Page

**Problem:**

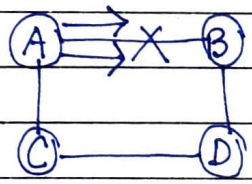
Heavy ~~tra~~ traffic due to flooding → Duplicate packet may reach at different times
- Sequence number
- Helps keep latest information
- Do not forward old packets
- Helps reduce traffic

TTL: Solves infinite loop

**Problem (Again!)**

Error in seq no.
May be quite high
no more update

**Looping problem: (not persistent though)** Transient




~~Black hole~~

**Black hole problem:** A→B pace packets are wasted

(Transient)

| DVR | LSR |
|---|---|
| (I) Low b/w req | (I) High b/w req |
| (II) Local knowledge | (II) Global knowledge |
| (III) Less traffic | (III) High traffic |
| (IV) Slow convergence | (IV) Fast conv |
| (V) Loop problem | (V) Not permanent |

# (Networking)

## TCP:

### TCP Header:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Source port (16) | | | | | | | | (16) Destination port |
| sequence no (32) | | | | | | | | |
| acknowledgement no (32) | | | | | | | | |
| (4) Header length | 6 bit | U | A | P | R | S | F | Adv window (16) |
| checksum (16) | | | | | | | | urgent pointer (16) |
| options (0-40B) | | | | | | | | |
| Data | | | | | | | | |

Header (20-60) B

length:

→ port no is needed to identify the process.

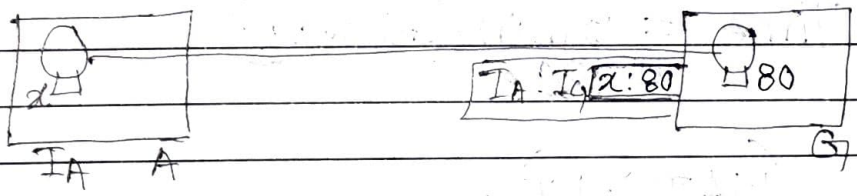
End-to-End protocol

Multiplexing & demultiplexing

$(0, 2^{16}-1)$ 0-1023 well known, 1024-49151 Reserved, Rest → General

IANA → Assigned Authority → Internet Numbers

TCP connection oriented: resources reserved




(IP + Port no) = Socket → uniquely identify a connection

(32 + 16) = (48) bit

## Sequence number:

TCP is a byte stream protocol. Every byte that comes from application layer to transport layer is counted.

IP is packet stream : identification number

segment

**Random initial number:** (Least significant 32 bits of the clock)

**Wrap around time:**

$2^{32} = 4$ GB

After $2^{32}$ ~~num~~ seq no is used up start from the same seq no.

WAT depends on the ~~bq~~ bandwidth

BW = 1 mbps

↳ 1 sec → 1 mB goes out

$= 10^6$ B

$10^6$ — 1 s

$2^{32}$ — $\left(\frac{2^{32}}{10^6}\right)$ sec (1B – 1 seq no)

Life Time = 3 min (average)

= 180 s

(WAT > LT) ✓

BW = 1 gbps (WAT < LT) Problem

Date: 11/09/2020

Increase the seq no by using [Options] → [Timestamp]

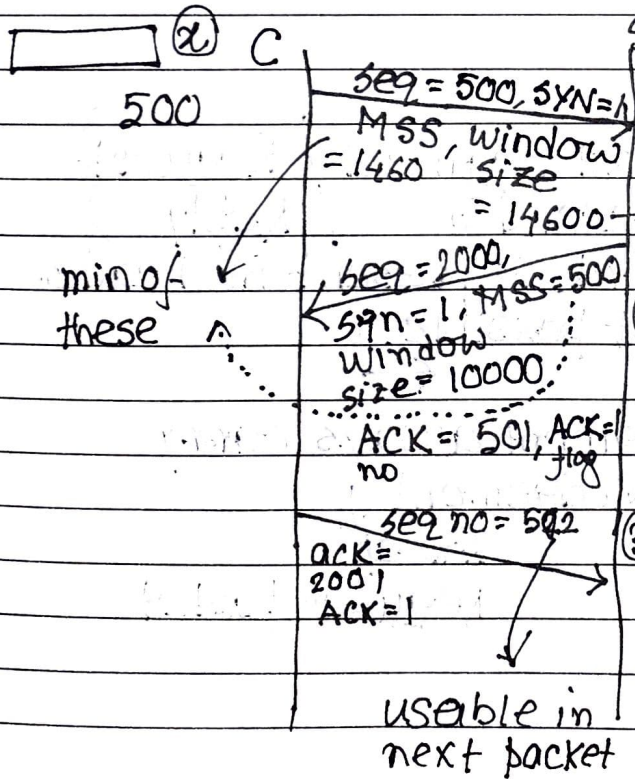
**Header Length:**
scale factor 4.

**Acknowledgement no:** Seq no expected next (byte)

SYN Flag : Synchronization
ACK Flag : Acknowledgement

**Connection oriented**
1. Connection establishment
2. Data transfer
3. Connection termination

**Establishment:**




Maximum Segment size (MSS) → (Options)

(Available buffer) sender can send 10 segments max

| 1460 | 20 | TL |
|---|---|---|

| 1480 | 20 | NL |
|---|---|---|

| H | 1500 | T | DLL |
|---|---|---|---|

Ethernet

(SYN) consumes one seq no

Date: 11/09/2020

TCP uses piggybacking and pure acknowledgement

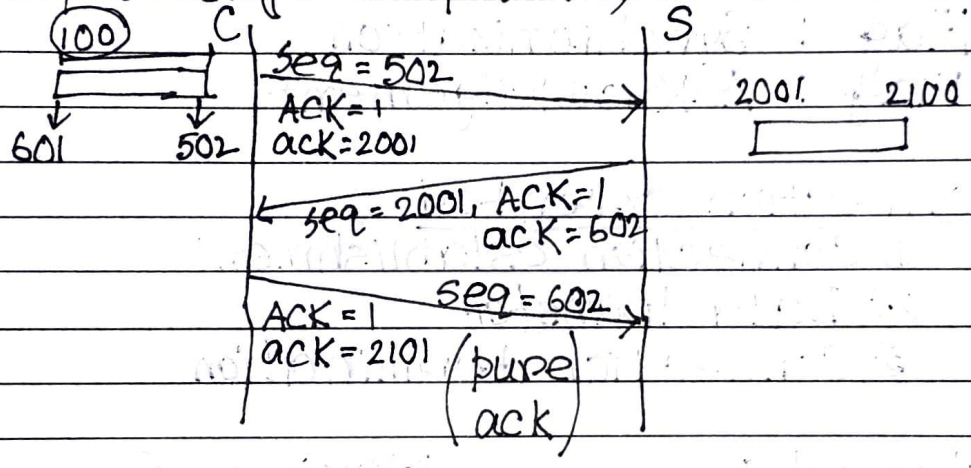
SYN=1 1 seq no
ACK=1, pure acknowledgement, 0 seq no
FIN=1, 1 seq no
1 data, 1 seq no

available for use in next packet

3 way handshake
connection established

Data Transfer:

500 (agreed upon MSS)




| SYN | ACK | |
|---|---|---|
| 1 | 0 | First segment / Request |
| 1 | 1 | Second " / Reply |
| 0 | 1 | Ack is present in header |
| 0 | 0 | Not possible |

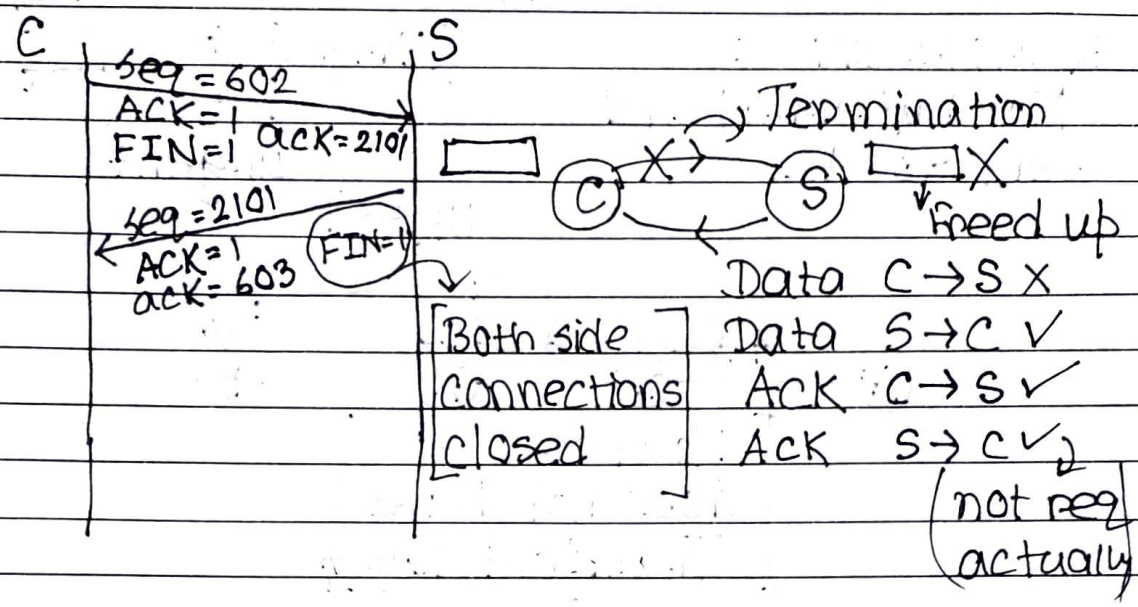
TCP segment always carry acknowledgement
Except.
Request packet

Date: 11/09/2020

# (Networking)

## (TCP)

## Connection Termination :




SYN : synchronizing seq no & MSS

ACK : Ack no field is valid or not

FIN : Request for connection termination

PSH : Push Flag

AL

↓ Byte

TL [ MSS ]

NL

DLL

PL

↗ waits for some time otherwise high overhead

↳ ~~Real time a~~

[ Interactive application : Problem

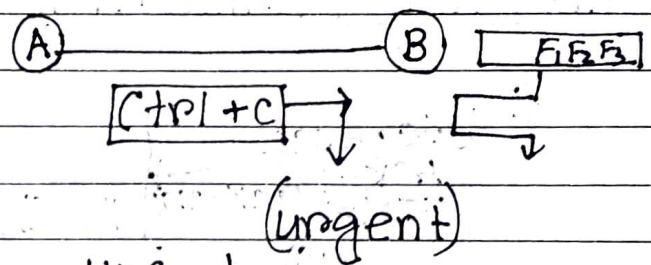
↓ PSH = 1

↓

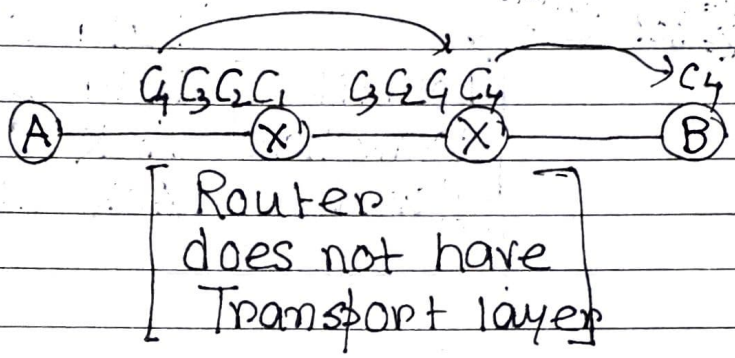
No buffering; just send how it is.

↳ (Same in receiver side)

## URGent pointer : flag




Receiver side :
Not buffered and sent to AL.

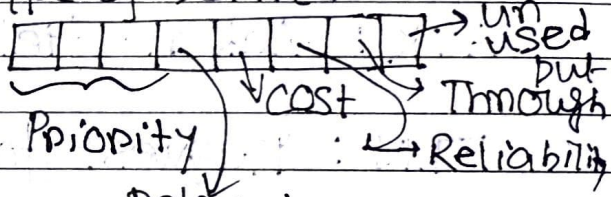



TL URG=1
NL priority = 7

Urgent pointer:
till which part is urgent.

Out of order packet.

IP datagram:
Type of service




send via the path with least delay.

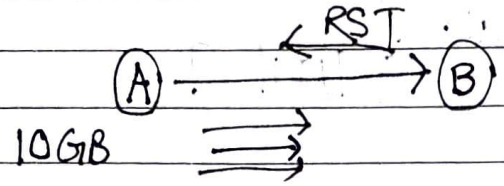
Reliability = 1
Send via most reliable path

Throughput = 1
Send via high throughput path

This is used by administrators normally. Not for normal users

## RST :

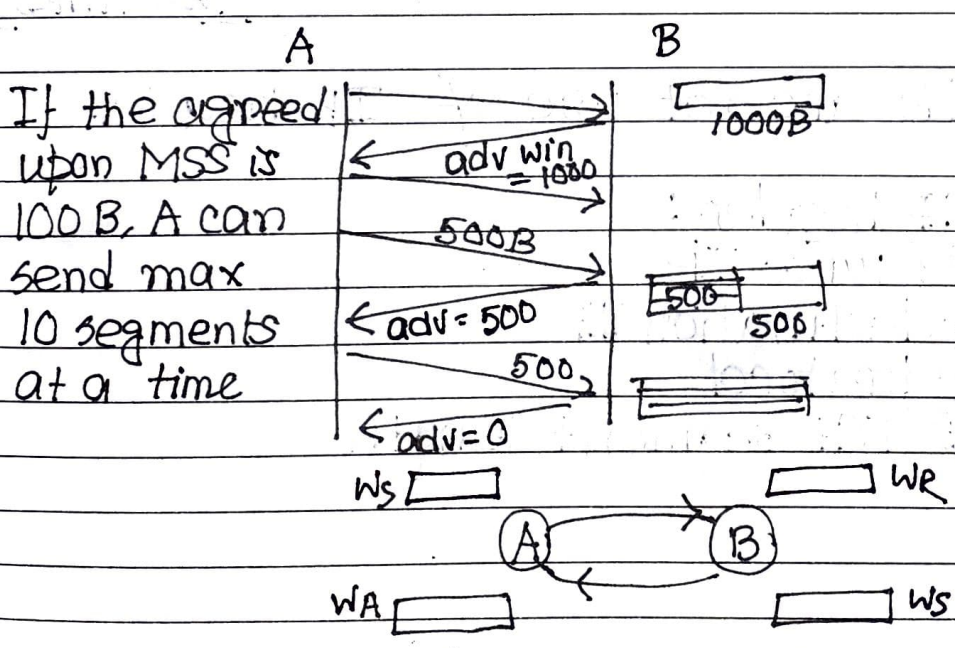
Reset flag




If something goes wrong
→ intruder
→ server down
→ any other strange thing happen

## Window size / Advertizement window :

used for flow control
sender should not send more than what receiver can receive

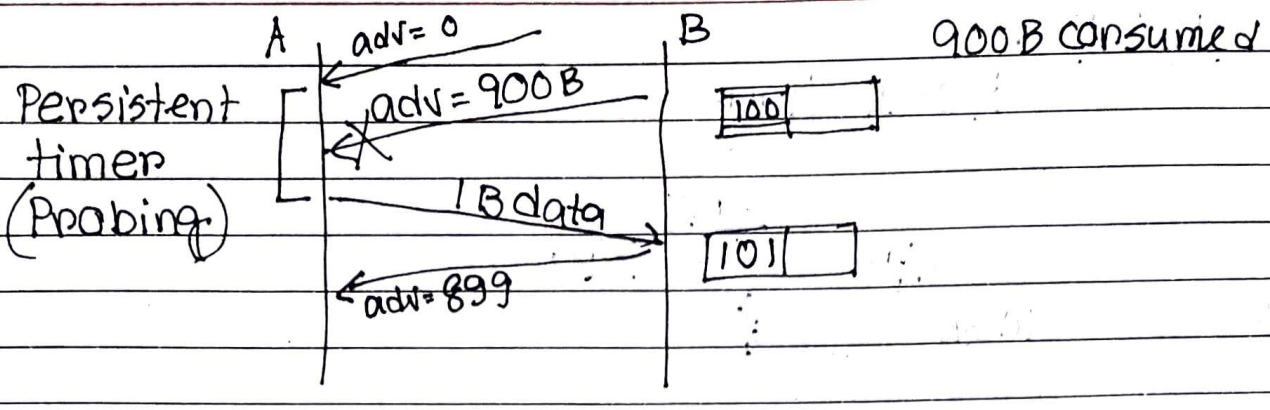






(TCP Full duplex)

Date: / / Page




adv window : 16 bit
$= 2^{16}$ B = 64 KB
To increase this limit
14 bit from options
(16 + 14) = 30 bits total

Checksum : 16 bit
(TCP header + TCP data) + Pseudo IP header

IP: Header checksum → IP header → Checked two times

All fields are not use of IP header

Pseudo IP header :
① source IP
② destination IP
③ protocol
④ TCP segment length

# (Networking)

## TCP Header:

### Options

① Timestamp: (WAT < LT)

② Window size extension

③ Parameter negotiation:
Ex: MSS

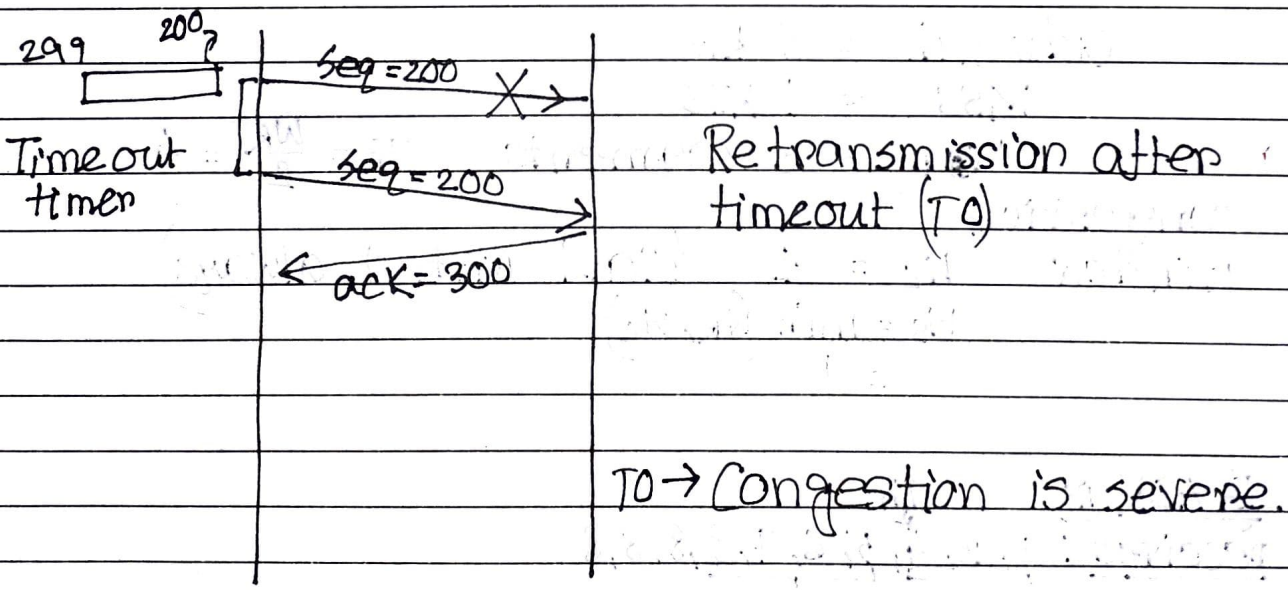
④ Padding:
Header size: multiple of ④

## Retransmission:

TCP uses a combination of
Selective + Go back N
Repeat

→ Ws = Wr
→ out of order

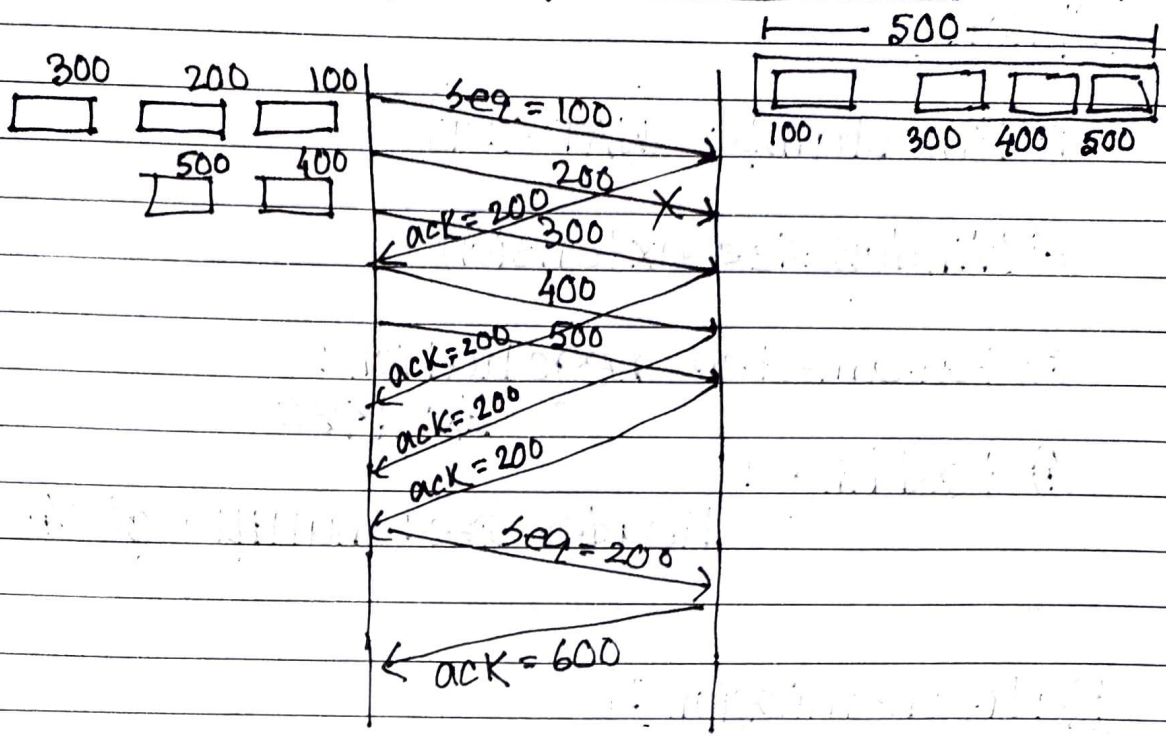
→ cumulative ack




Packet gets lost normally due to congestion.

Date: / / Page

## 3 duplicate ack / Early retransmission:




This indicates that congestion is not that severe

## TCP congestion control:

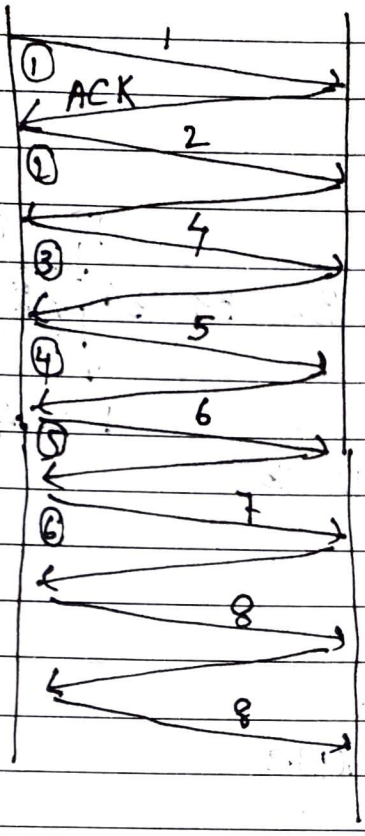
congestion meltdown

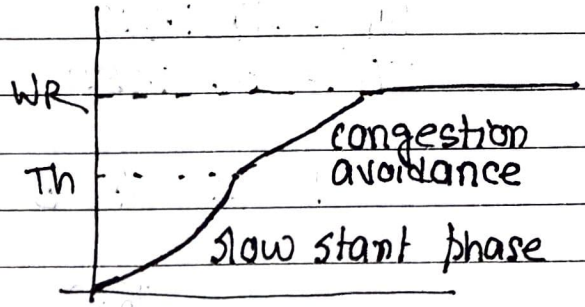
adv win = 8 KB
MSS = 1 KB
$W_R$ = 8 segments $Th = \frac{W_R}{2} = 4$

congestion window $W_C = 1$ (start with 1, always)

$W_S = \min(W_R, W_C)$
$= 1$

If ACK is received: 1, 2, 4, 5, 6, 7, 8, 8, 8, 8
Th
Exponential
Linear



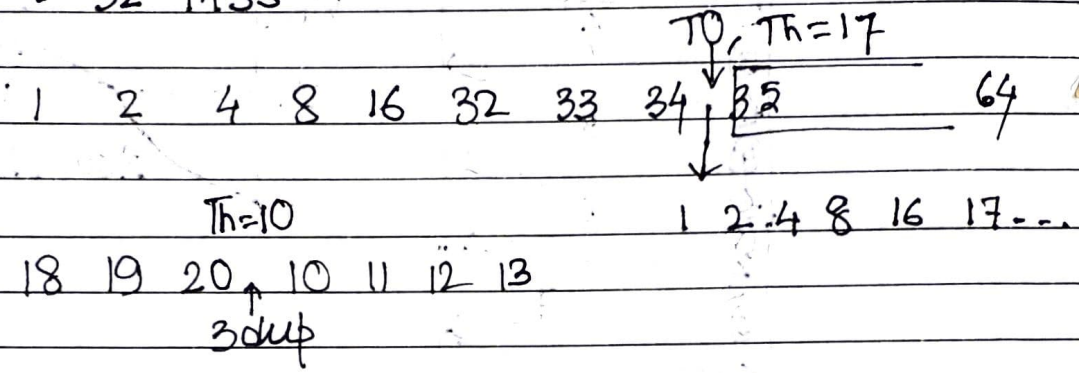




6 Round Trip Time : then full window capacity (8)

## Congestion Control Algorithm (CCA) :

(i) Slow start
(ii) congestion avoidance
(iii) congestion detection
→ Timeout
→ 3 duplicate packet
→ icmp source quench
one host only
not the entire network

$W_R$ = 64 KB
MSS = 1 KB
$W_R$ = 64 MSS
Th = 32 MSS




① Timeout :
New Th = $\frac{1}{2}$ Wc
Slow start phase

② Three duplicate ACK:
New Th = $\frac{1}{2}$ Wc
Congestion avoidance

(Networking)

## TCP:

### Timer management:

① Time wait timer
② Keep alive timer
③ Persistent timer
④ Acknowledgement timer
⑤ Time out timer

### ① Time wait timer:

When request to close a timer is received, ~~de~~ wait for (2 x Life Time)

If immediately closed the port no becomes available → delayed packet problem.

### ② Keep alive timer:

idle connection

~~When Client sends a packet~~
When Server receives a packet from Client it restarts the ~~keep~~ Keep Alive Timer → this keeps track of last time packet received from the client

KAT expires → probing (10 probes)

↓

connection termination (free resources) ← no reply from client

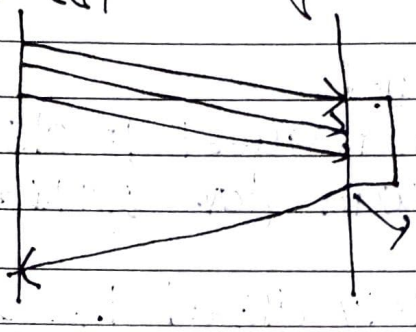
Date: / / Page

③ Persistent timer:

Adv window = 0

④ Acknowledgement timer:

cumulative acknowledgement : + piggy backing




⑤ Time out timer:

Dynamic nature

① Basic
② Jacobson

Network layer
TO = 2 x RTT
since its hop to hop.
Things are complex in Transport layer since nothing known about the network

Date: / / Page

1) Basic algorithm:

Initial RTT = 10 ms (guess)

TO = 2 × RTT

= 20 ms

Actual RTT = 15 ms

NRTT = $\alpha$ IRTT + $(1-\alpha)$ ARTT

$\alpha$ → smoothing factor (0,1) (0.5)

= 12.5

TO = 25

ARTT = 20

NRTT = 0.5 × 12.5 + 0.5 × 20

= 16.25

2) Jacobson's algorithm:

Deviation

IRTT = 10 ms guess

ID = 5 ms

TO = 4 × D + RTT

= 30

ARTT = 20 ms

AD = |ARTT − IRTT| = 10 ms

NRTT = $\alpha$. IRTT + $(1-\alpha)$ ARTT

= 15 $\alpha$ = 0.5

ND = $\alpha$ ID + $(1-\alpha)$ AD

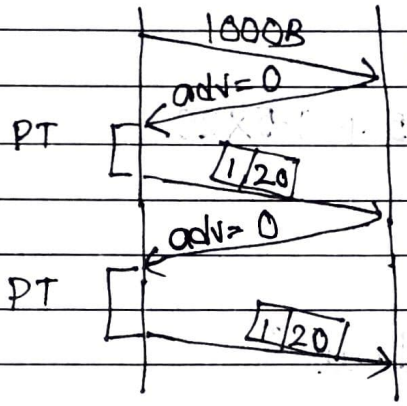
= 7.5

If not received within TO:

3) Karn's modification: double the TO and retransmit.
Keep retransmitting until received within TO.

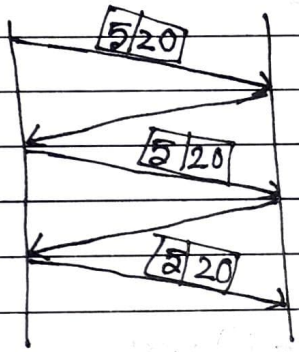
Silly Window syndrome: Efficiency ↓

case 1




(Transient problem)

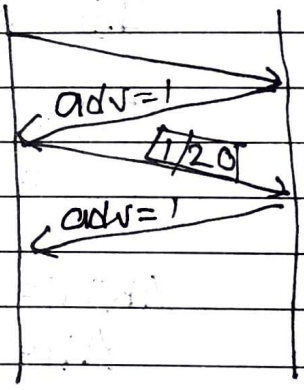
case 2:




Sender is producing data slowly

Nagle's algo: Transport layer must wait 1 RTT. Send that data then. wait till
min(1 MSS generated, 1 RTT)

case 3




Receiver is consuming data slowly

Clark's solution:
wait ~~until~~ till
min(1 MSS free, ½ buffer free)
then advertise

Date: 12/09/2020 (249)

# Networking:

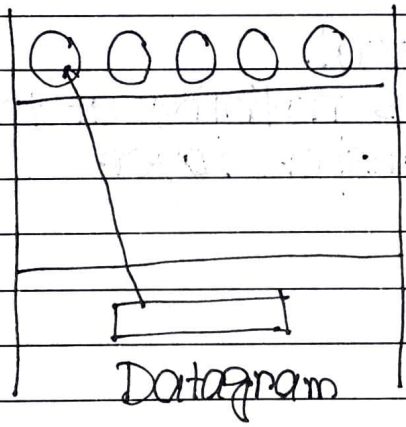
## UDP:

TCP is not efficient for:

① Application needs 1 req/1 reply
TCP overhead too high
→ DNS
→ Bootp, DHCP
→ network time protocol

② Broadcasting/Multicasting
TCP will require reserving buffer for each receiver

③ Speed needed rather than accuracy:
→ multimedia
HD: TCP
not HD: UDP
→ online games

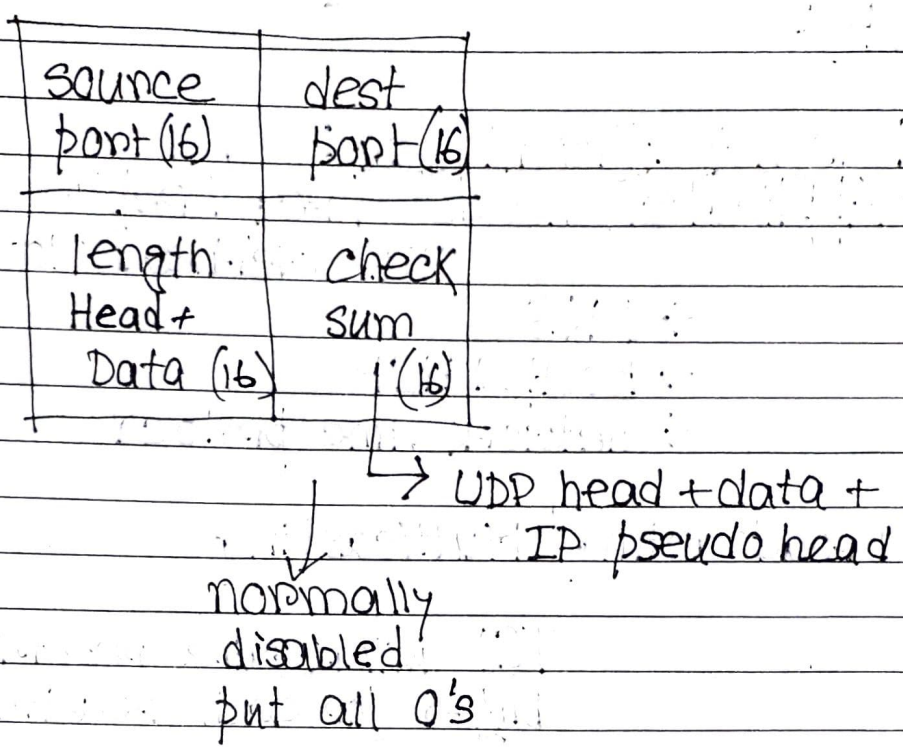



just identify the user process for that datagram and ~~pass~~ send to it

(user datagram protocol)

No ack

AL NL




① Traceroute: UDP should be able to activate traceroute in the IP datagram.
Record Route!

UDP is NULL protocol.
does nothing at the Transport Layer

# Hardwares: (Devices)

AL
TL
NL
DLL
PL

[A]

## Ethernet Cable:

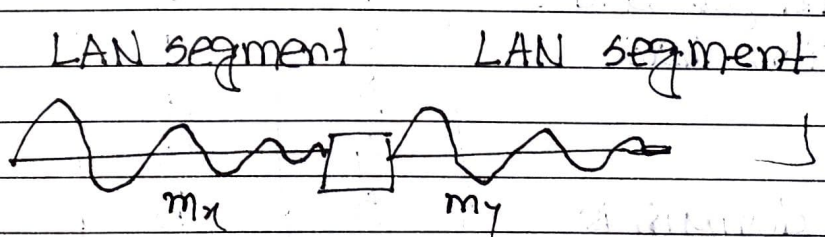
10 Base (2) → 200m
10 Base (5) → 500 m
(10) Base (T) → max length 100 m

10 Base 10 mbps bandwidth

no multiplexing
- Baseband: only one signal at a time via cable
- Broadband: multiple signals

1. operate at physical layer
2. attenuation
3. collision is possible. collision domain (m) ← # hosts connected

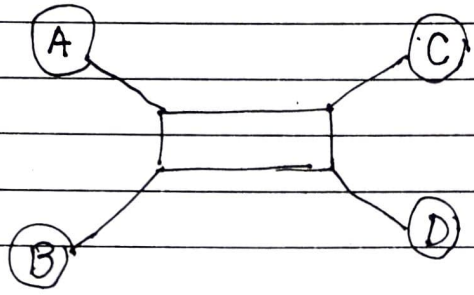
## Repeater: generate the same signal




→ physical layer
→ collision possible inside Repeater
→ collision domain : (m)

$m = m_x + m_y$

## Hub: multiport repeater



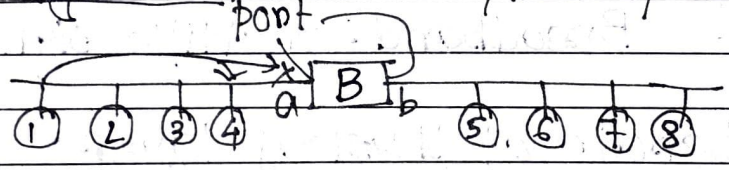

① Physical layer
② High traffic
③ Collision inside hub possible
④ collision domain (m) = $m_A + m_B + m_C + m_D$

Adv: Cheap

**Bridge**: It can connect (two) different type of LANs → multiple

Spanning Tree Algorithm → to avoid loops

Dynamic / Transparent / Learning




① Physical and Data Link layer
— can look at MAC addresses

→ can filter / forward Forwarding table

| MAC | PORT |
|---|---|
| 1 | a |
| 2 | a |
| 3 | a |
| 4 | a |
| 5 | b |
| 6 | b |
| 7 | b |
| 8 | b |

→ store and forward: no collision inside bridge

→ collision domain is reduced.

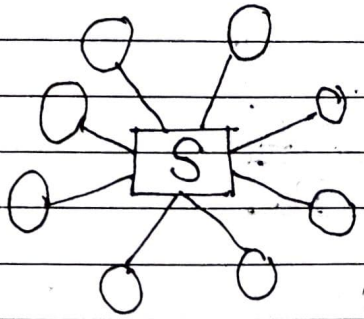
If ababababffff

# GATE

Date: 12/09/2020 Page 253

## (Networking)

### Switch:

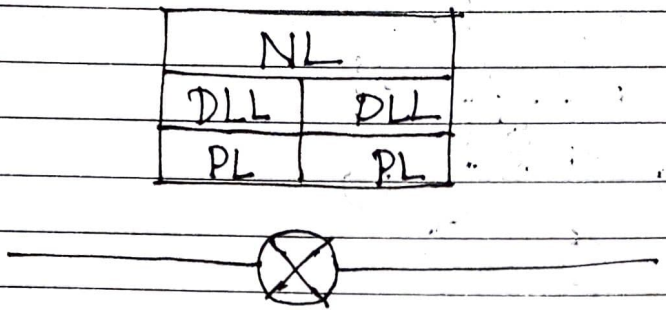



① Physical & Data link layer
② Links are full duplex
③ No collision. Collision domain = 0
④ Less traffic

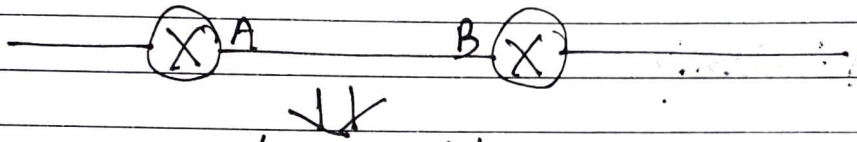
Disadv: costly

| | Broadcast domain | Collision domain |
|---|---|---|
| Repeater | same | same |
| Hub | same | same |
| Bridge | same | reduced |
| Switch | same | 0 |
| Routers | reduced | reduced |
| Gateway | reduced | reduced |

## Routers:

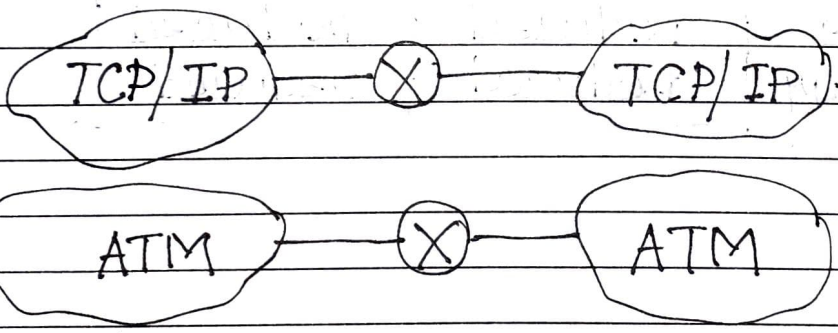



(i) MAC
(ii) IP addresses



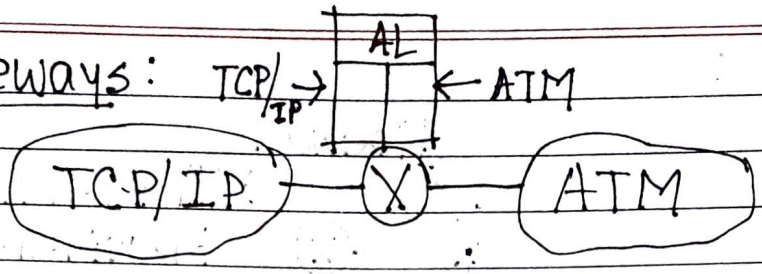

4 IP addresses

| | |
|---|---|
| 110.1.2.0/30 | NID |
| 110.1.2.1/30 | A |
| 110.1.2.2/30 | B |
| 110.1.2.3/30 | DBA |

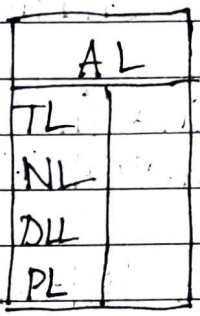


Date: / / Page

## Gateways:

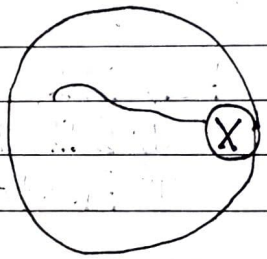



① protocol converter




② proxy :

Every outgoing packet goes through the proxy




proxy also works as cache/buffering

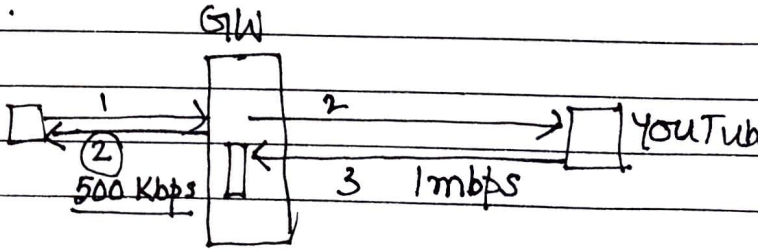
③ NAT :

④ Firewall

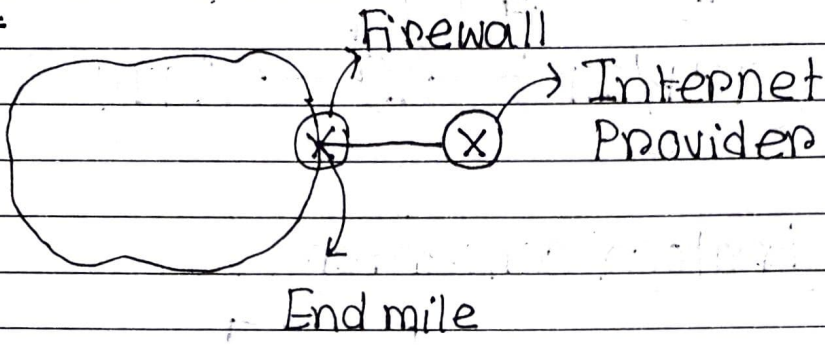
⑤ Deep Packet Inspection : (DPI)

→ video is present in web page.
→ malacious

⑥ Cache/buffer :



## Firewall :




Anything (coming in / going out) passes through firewall

Firewall type
① Layer 3 / packet filtering
② Layer 4
③ Layer 5 / proxy

Layer 3: PL, DLL, NL
(i) SIP, DIP : block the IP of hosts
(ii) Protocol : any protocol can be blocked
ICMP attacks can be blocked
(iii) A protocol from a host

Layer 4: PL, DLL, NL, TL
(i) SIP, DIP
(ii) Protocol
(iii) Port no : block a service
TELNET → block remote login
SMTP → block sending email outside
FTP → " " file transfer

# GATE

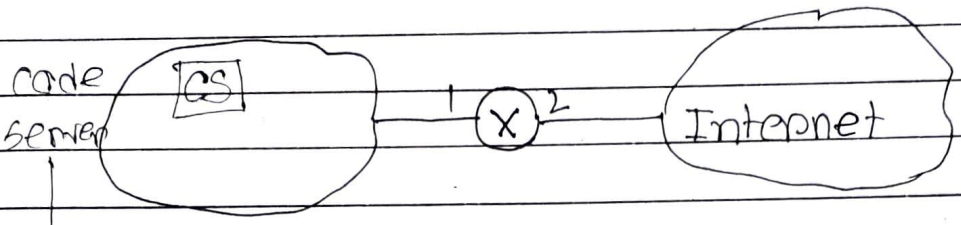
Date: 12/09/2020

## (Networking)

### Proxy/:

### Layer 5 Firewall:

PL, DLL, NL, TL, AL




no file should go out

① SIP, DIP: block a host
② Port no : block a service
③ service on a host
④ Application Layer data
- username & password authentication

### Rule table:

(21) FTP

| | SIP | DIP | SPort | Dport | Interface | user |
|---|---|---|---|---|---|---|
| 1. | Ics | – | 21 | – | 1 | – |
| 2. | – | Ics | – | 23 | 2 | NE |
| 3. | – | – | – | 80 | 1 | – |
| | | | | | | |

If a packet matches Rule Table, it is discarded.

① Stops any file going outside from code server
② (23) Telnet
no login from outside for non employees
username
password
③ No web page is coming → block internet.

## Application Layer & Protocols:

DNS
HTTP
(email) → SMTP
FTP
POP

## Domain Name Service (DNS):

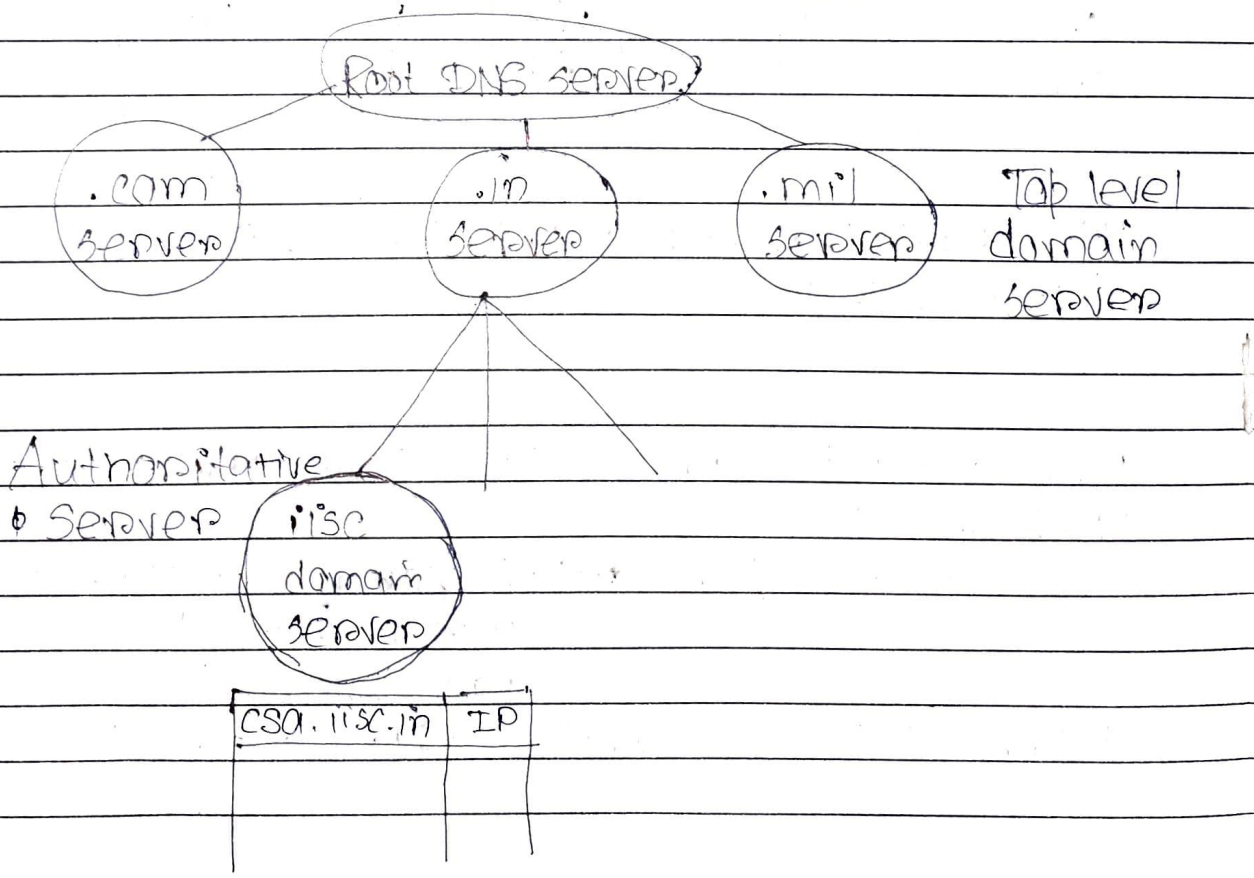
Translation: Name → IP → may change

**Domains**

(i) Generic domain
.com .edu .mil .org .net

(ii) Country domain
.in .uk .us

(iii) Inverse domain
IP → domain name



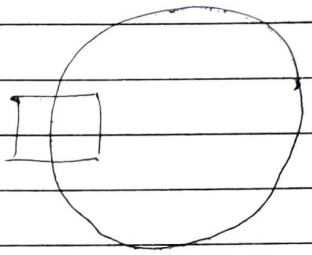
Date: / / Page

DNS record

google.com , IP , time validity

IETF has maintained (13) DNS root servers.

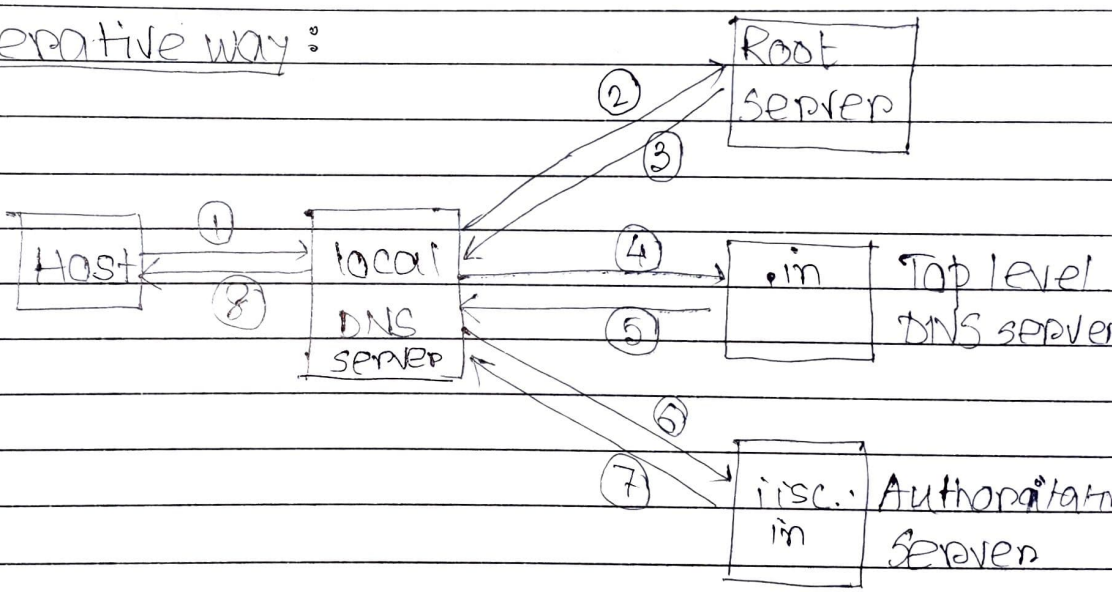
If one server fails, rest will work.




DNS overhead

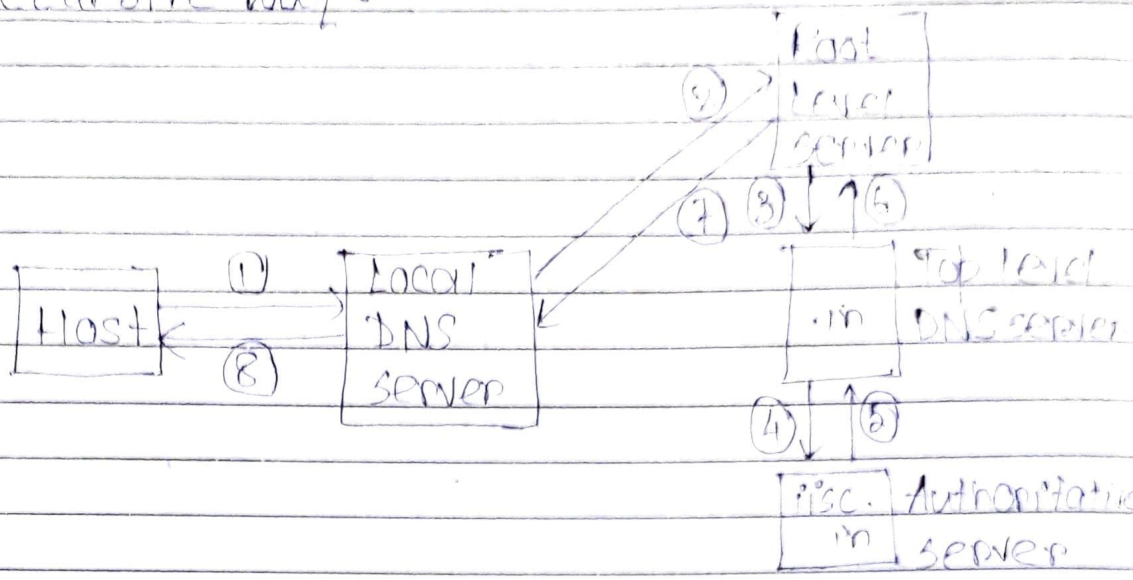
(domain name → IP translation time)

① Iterative way:



Date: / / Page

(2) Recursive way :




DNS uses UDP at Transport layer since all the messages are (Request - Reply) type.

# (Networking)

## Application Layer Protocols:

**HTTP:** runs at port no 80.

HTTP requires reliability
Ex: TCP

→ HTTP is inband protocol.
Data and commands goes in same stream.

→ HTTP is stateless → no user info is maintained.

cookies → client side

HTTP 1.0: non persistent connection
For every object one connection will open and closed after done.

HTTP 1.1: persistent connection
One connection for all

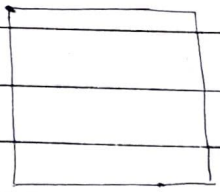
(1) Head: meta data of web page
web cache / HTTP cache

(2) Get:
(3) Post
(4) Put
(5) Delete
(6) Trace
(7) Options
(8) Connect: HTTPS

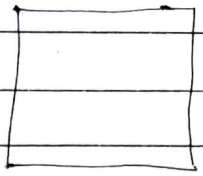
Date: / / Page

# FTP (File Transfer Protocol)

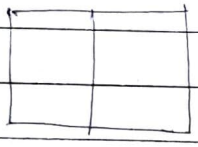
File Client



File Server



① IP of server
② Port (21)



→ control connection : persistent
→ data connection : non persistent
  · Opens at a file transfer and closes after it

→ out of band : command & data different stream

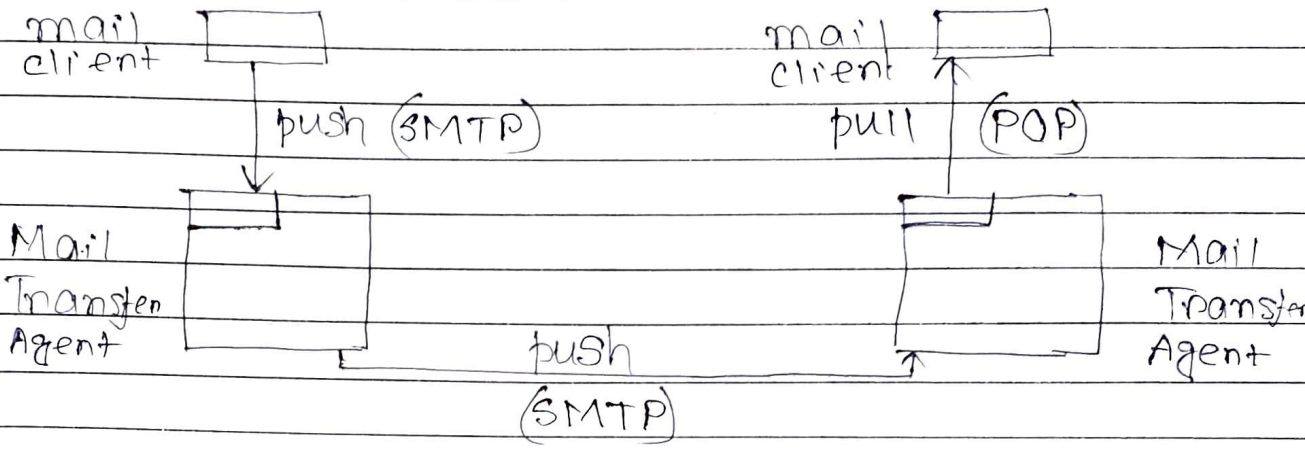
→ FTP requires reliability (TCP)
→ FTP is stateful.

→ client & server must be online together

Date: / / Page

SMTP (Simple Mail Transfer Protocol)
POP (Post Office Protocol)

Client & server need not be online together.




SMTP, POP: requires reliability (Ex TCP)
~~Multimedia~~ → in band, Text only
Non text → Text ⟶ Text → Non text
MIME
Multipurpose Internent Mail Extension

Gmail: (web page based email)
(inbox ⇒ web page)

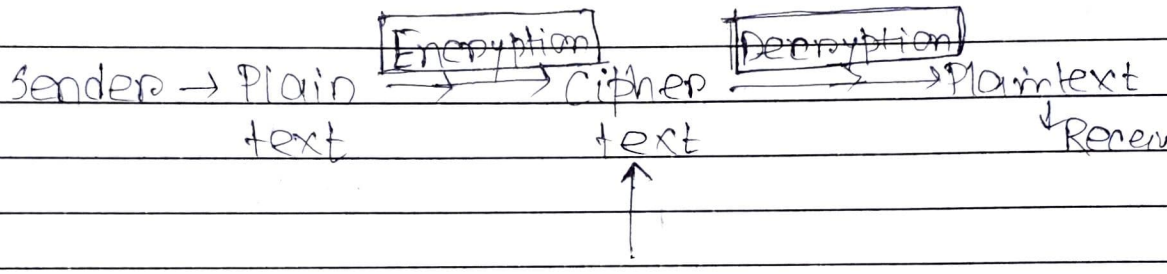
# (Networking)

## Network Security:

→ Plain text — Readable
→ Cipher text — Unreadable
→ Encryption
→ Decryption

→ Cryptography

### Traditional method:

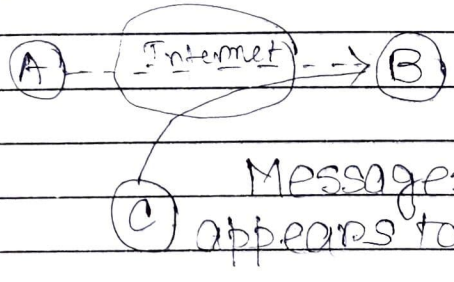



**Passive attack:** only see the data, cannot change the contents

Ex (i) Reading the content
(ii) Traffic analysis

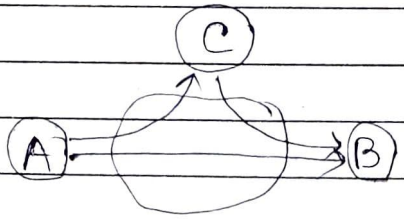
**Active attack:** can see and modify the data

(i) **Masquerade:**

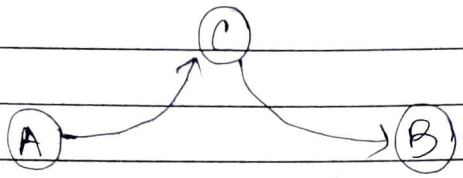


Date: / / Page

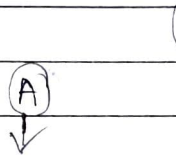
Replay.




Modification:




Denial of Service:




Disrupts service provided by ~~user~~

Does not get any chance to communicate with B.

Security services required to ensure the security of ~~comm~~ networking:

1. Authentication: signature
2. Data integrity:
3. Access control: password
4. Data confidentiality: encryption
5. Non repidiation: protection against denial by one of the entities involved in a communication of having participated

Date: 11/09/2024 Page

## Modular Arithmetic:

→ $a \equiv b \bmod n$

$(a \% n) = (b \% n)$

$13 \equiv 1 \bmod 12$

$24 \equiv 12 \bmod 12$

→ $n$ divides $(a-b)$

→ $a \equiv b \bmod n$

$c \equiv d \bmod n$

$(a+c) \equiv (b+d) \bmod n$

$(a-c) \equiv (b-d) \bmod n$

$(ac) \equiv (bd) \bmod n$

## Multiplicative Inverse:

$a \not\equiv 0 \bmod p$, $p$ prime

$\exists b$ such that $(a, b)$ coprime

$ab \equiv 1 \bmod p$

$b = a^{-1} \bmod p$

$a = b^{-1} \bmod p$

Ex: $2 \not\equiv 0 \bmod 7$

$2.4 \equiv 1 \bmod 7$

$2 = 4^{-1} \bmod 7$

$4 = 2^{-1} \bmod 7$

* If $a, n$ are co prime, then $a$ has a multiplicative inverse mod $n$

$\gcd(a, n) = 1$

Date: 14/09/2016 Page

## Eular's Theorem:

n +ve integer
gcd (a, n) = 1 (coprime)

$$a^{\phi(n)} \equiv 1 \bmod n$$

$\phi(n)$ : Eular totient function
# integers < n, which are relatively prime to n.

$$a^{\phi(n) \cdot t} \equiv 1 \bmod n$$
t integer

## Fermat's Theorem:

Special case of Eular's theorem

$$a^{n-1} \equiv 1 \bmod n$$ n prime
gcd (a, n) = 1

→ $a^{\phi(n)+1} \equiv a \bmod n$
→ $a^{\phi(n)t+1} \equiv a \bmod n$

# (Networking)

## Eulor's Totient function:

$\phi(n)$

#positive integers $< n$, ~~gcd~~ coprime to $n$.

$\phi(3) = 2$ (1, 2)

$\phi(14) = \phi(2)\,\phi(7)$

$= 1 \times 6$

$= 6$

$\phi(mn) = \phi(m)\,\phi(n)$ $\boxed{\gcd(m,n) = 1}$

(Prime factorization)

$n = p_1^{e_1} p_2^{e_2} \cdots p_x^{e_x}$

$\phi(n) = n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_x}\right)$

$\phi(36) = 3^2 2^2$

$= 36 \cdot \left(1-\frac{1}{3}\right)\left(1-\frac{1}{2}\right)$

$= 36\ \frac{2}{3}\ \frac{1}{2} = (12)$

## Primitive root:

$a \equiv (b) \bmod n$

↳ residue

Residue class: $[f(a)] \bmod n$