COMPUTER NETWORKS

Network → Interconnected devices

Computer Network → Interconnected computers

What? – Internet / Interconnected computers

Why? – Communication

How? – Protocols.

Internet – "network of networks".

Backbone of Internet ← Router → gives route to every particular packet

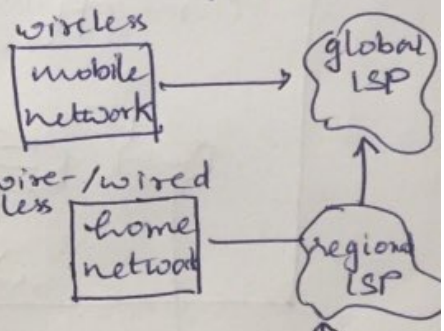
ISP – Internet Service Provider

Internet Standards → control sending & receiving of msgs. [rules]

↓

RFC – Request for Comments

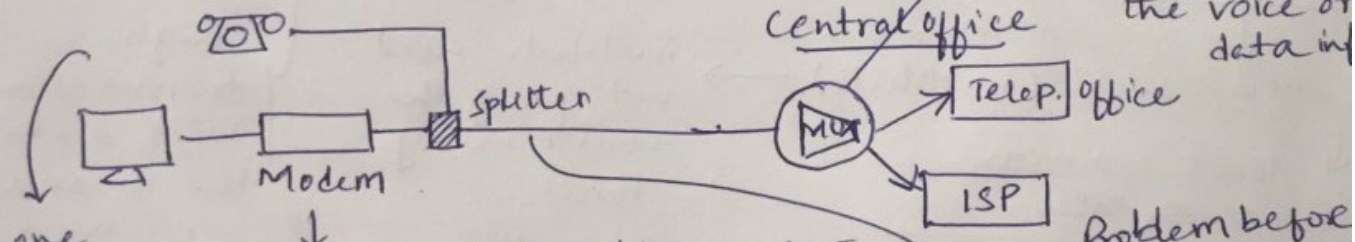
Ex: TCP, IP, HTTP, 802.11, Skype



Network structure:

→ Network Edge → hosts – clients & servers

servers in data centers



* one connection at a time
* 1 Kbps (loads slowly)

↓ converts the information packets to analog signals.

↓

DSL – Digital Subscriber Line

"Broadband connection"

Problem before

during reception → the analog signal waves are converted into suitable bits & packets

So splitter is introduced to split the signals

Hence, we can access 2 connections at a pint

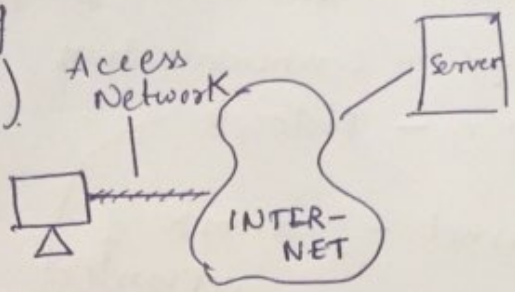
similar to telephone network

⟹ Cable network → TV signal (cable is tapped for usage)

Not secure [as there can be many tappings]

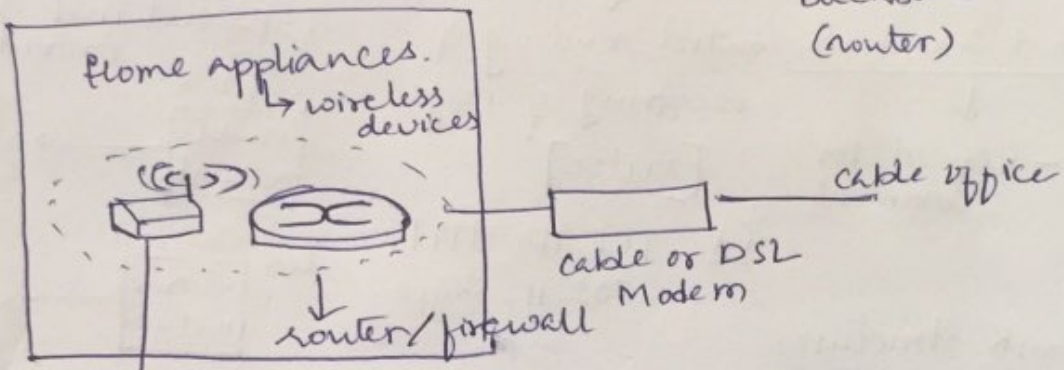
For security and more information passing

frequency division multiplexing (different channels on diff. frequencies)



Backbone (router)

Home Network



wireless access point

Ethernet — Enterprise access networks → Connections are made through ethernet switches.

web servers (institute) → stores common access sites

institute need not ping the router many times

high transmission rates due to the co-axial wires used.

Wireless Access networks →

3G, 4G (wide-area wireless).

Wifi (wireless LAN).

Medium → physical / wireless

LAN – twisted pair (two copper wires twisted)

Physical Media

→ guided media → signals are guided like in wires or cables [Ex: copper, fibre, coax]

→ unguided media – freely propagation

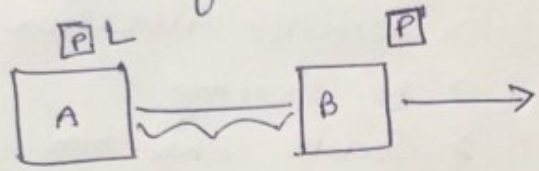
Ex: Radio.

coaxial cable → to avoid noise or disturbance (like in telephone wires)
↓
concentric copper conductors.

fiber optic cable → glass fiber (prevents noise)
↑ signal strength – cannot be tapped, secure connections

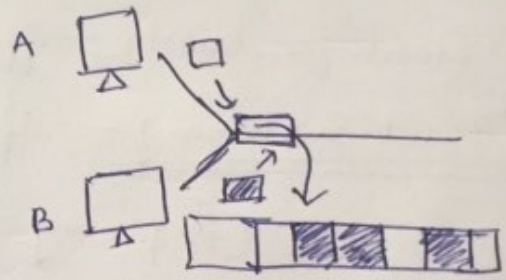
Host → sending data packets. → packets of data → break into chunks known as packets
↓
transmitted at rate 'R'

Packet Switching
↓
"Store and forward"

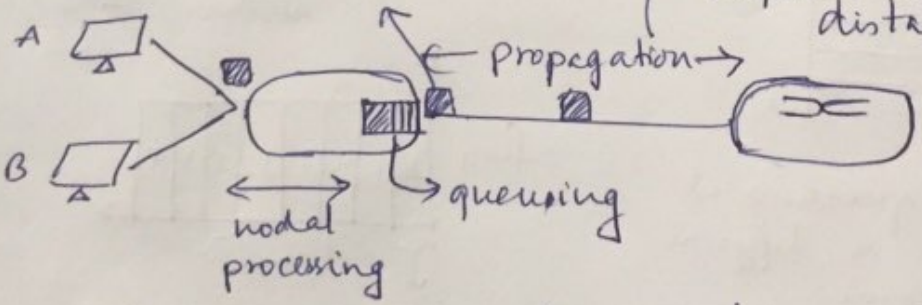


Transmission Delay $= \frac{L}{R}$

queuing → if arrival rate > transmission rate of the link to link.
↓
limited queue size
if it exceeds → then the packets are dropped



Packet Delay =



length of the physical link.
(distance)

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop} = d/s$$
(processing) (speed)

d_{prop} ↓ changes with medium [propagation of the * radio signal]
not (bit)

Packet Loss → you can retransmit the packet
↓
reaching its time to live even before it reaches.

Throughput – rate at which bits transfered between sender/receiver (bits/unit time)

depends on the capacity of the medium

Alternative for Packet Switching – "Circuit Switching" ∵ secure the connection

Advantages
- No queuing
- Secure connection

dedicated resources = no sharing

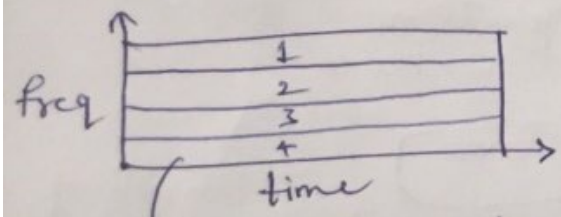
disadvantages
→ delay in the circuit causes havoc in other networks

Packet Switching (Vs) Circuit Switching

Packet Switching	Circuit Switching
> No Resource Allocation	> Resource Allocation
> Sharing of Resources	> No sharing
> Less Secure	> Secure connection
> Queuing delay	> No Queuing delay
> No QoS guarantee.	> Quality of Service [the packet reaches the destination]
> Good for bursty traffic	

Circuit Switching – if all the users want to use the same path

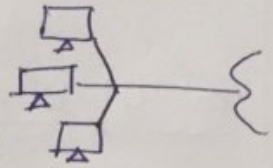
FDM (frequency division)

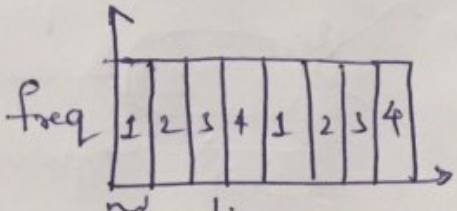


4 users are operating at the same time in different frequency regions.

TDM

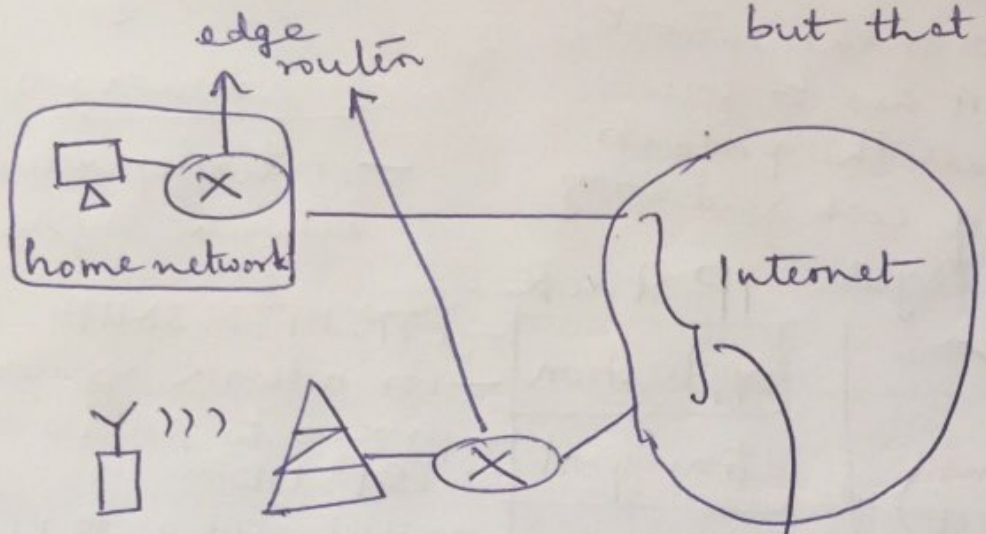
Time slots are given





channel is given entirely for 1 user then control transfered to 2nd user.

In case of heavy traffic, for circuit switching you need to allocate resources to all the users but that is difficult

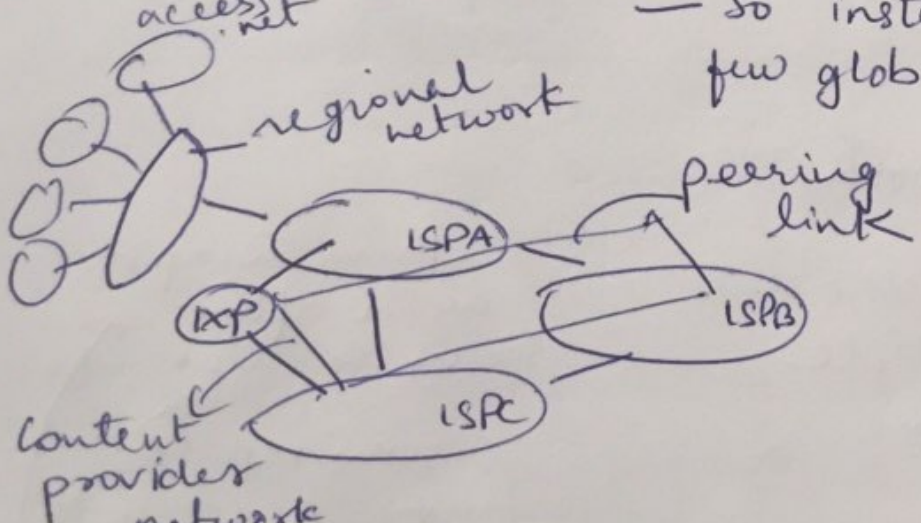


how are these connected?

Using global ISPs (a common point) to all the access nets

IXP → internet exchange points.

- but having a single global ISP is not secure as if that gets hacked, then everything comes out
- so instead of one, we have few global (main) ISPs.



Layers – protocols to be transmitted over a medium

application
transport
network
link
physical

→ for a packet to be transmitted over the network, it has to follow all the protocols of each and every layer

Advantages
– Modularity

Disadvantages
– Interdependency between layers

– (Internet Protocol) (IP stack) ✓

IP stack

application	– FTP, HTTP, SMTP – For network applications
transport	– process to process communication – TCP, UDP
network	– packet shld go to ? (next destination) – IP, routing protocols
link	– error correction methods [correct packet or not]
physical	– digital to analog and transmission over medium

bits on the wire

– OSI (open systems interconnect) reference model

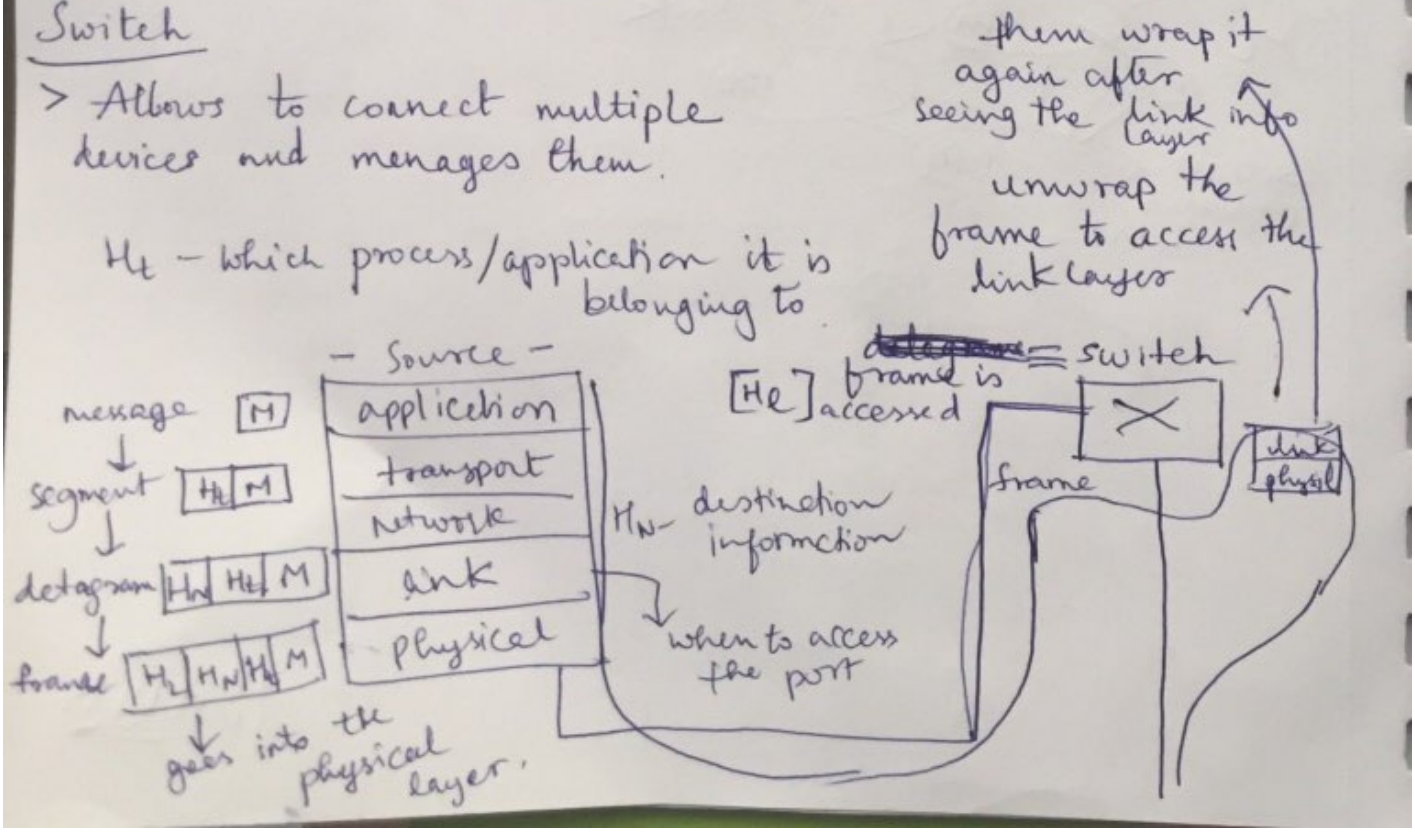
till here we know all the superficial information – where to go, process data of the packet

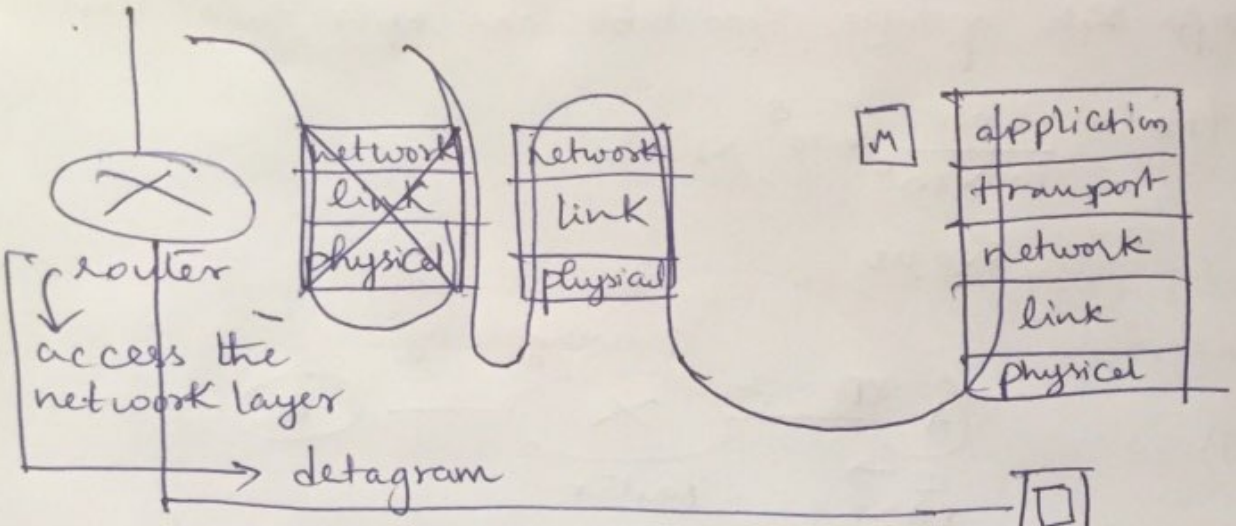
2 extra "layers" = presentation, session → "checkpoint"

Router
> Directs data in the network.

Switch
> Allows to connect multiple devices and manages them.

H_t – which process/application it is belonging to

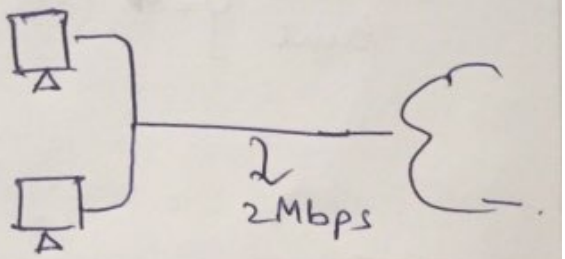




[Intermediate routers can be changing, so the frame [HL] can be modified or changed]

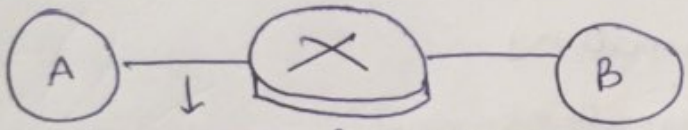
A 2Mbps link is shared in some users. Suppose it transmi at 1 Mbps. If Packet Switching is used, will there be any delay if there are

- 2 users → No delay
- < 2 users → No delay
- > 2 users → Delay



propogation delay = 20μs on each link.

processing delay for a packet = 35μs



1000μs + 20μs + R 35μs + 1000μs + 20μs

packet switching "store and forward" device.

processing ✓
Queuing ×
transmission ✓
propogation ✓

> a packet of size $10,000 = 10^4$ from A to B

$(20\mu s)2 + (35\mu s) + (10^3 \mu s)2$

$= 2075\mu s$

$t = \frac{L}{R} = \frac{10 \times 10^6}{10^7} = 10^{-3}$

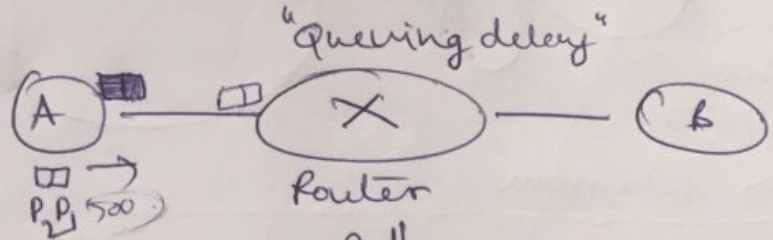
$= \frac{10^4}{10^7} = 10^{-3} s$

$= 10^3 \mu s$

> 2 packets of size 5000 bits, one after the other.

$$d_{tran} = \frac{5000}{10 \times 10^6} \times 10^6 \ \mu s$$

$$= 500 \ \mu s$$



R

~~2(500μs) + (20μs)2 + (35μs)2 + 2(500μs) + (20μs)2~~

1000 20 70 1000 40

~~[35 + 500] +~~

~~queuing + 500 μs + 35μs~~

queuing delay

P_2 P_1 processing P_1 transmitting P_1 done P_1

Total time ⟶ 500μs + 20μs + 35μs + 500μs + 20μs + 500μs

1075

= 1575μs

t=0 : P_1 starts transmitting

t=500 P_2 starts transmitting

t=520 P_1 reaches R

t=555 P_1 is processed at R
& P_1 starts at R

t=1000μs P_2 finishes transmission

t=1020 P_2 reaches R

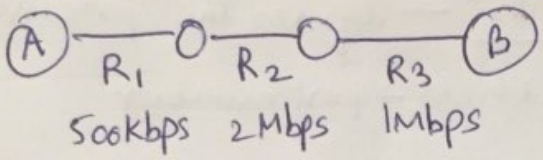
t=1055 P_2 is processed at R
P_1 is transmitted
P_1 starts propogating towards B
P_2 starts transmission

t=1075μs P_1 reaches B

$t = 1555\mu s$: P_2 is transmitted
P_2 starts at R

$t = 1575\mu s$: P_2 reaches R

Total Time: $1575\mu s \equiv 3 \times tx + 2 \times propagation + 1 \times processing$



— Throughput = 500kbps (R_1)

File Size = 4 million bytes $\equiv 4 \times 10^6 \times 8$

Transmission delay = $\frac{4 \times 10^6 \times 8}{500 \times 10^3} = 64sec$

APPLICATION LAYER —

* Creating a network app. —
  * write programs
    — run on end systems → have the five-layered architecture.
  * no need to write software for network core devices (backbone) → require the datagram "Network layer" for transmission

↘ Protocols —

open protocols
- HTTP
- SMTP
} interoperability

Architecture ↗ client-server
↘ peer to peer (P2P)

server —
- always-on
- non-mobile
- (fixed IP) location should not change

clients —
- communicate with server (need not be on always)
- can be mobile (dynamic IP)

[two end systems do not communicate with each other]

[Communications between 2 End-systems] ≡ direct communication

~~no need to~~ be "on" always
may not be
(so connection may or may not establish)
→ Erase dynamic IP addresses.
(difficult to track down the location)

(peers request service and also provide service)

*: bluetooth, self-reliable network, device-device communication (no connect with the base station)

= Complex Management

Communication among processes → Identifying the process

↙ client process – initiates a connection.

↓ server process

Identifier ≡ (32 bit IP address)

↓

? but IP address of host is not enough to identify the process

↙

so "ports" – defines the processes

Identifier ≡ IP address + port number

- Services required by an application?
  - → Security (encryption)
  - → data integrity (no data loss but should reach intact)
  - → timing (ex: internet telephony, interactive games)
  - → Throughput

Transport Protocol Services – (donot care about any of the TCP features)

TCP service | UDP service → UDP is faster than TCP (like telephoning)

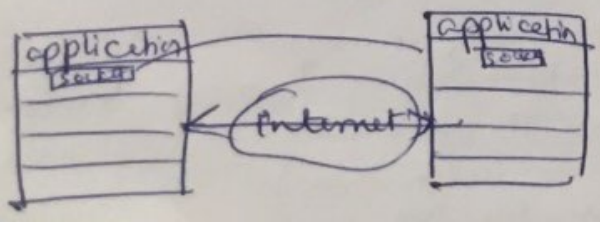
– ~~secure~~, (not that secure) reliable – unreliable

– retransmission
– flow control
– doesnot provide "timing, min. throughput"
– setup a connection before the transmission

– doesnt provide any of these things.

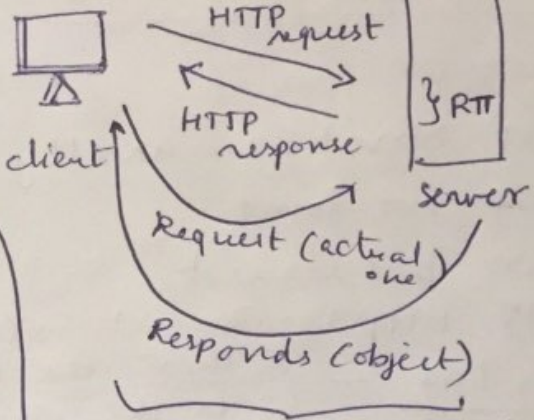
↔ connection – orientation protocol

Sockets: processes send/receives messages to/fro from its sockets.

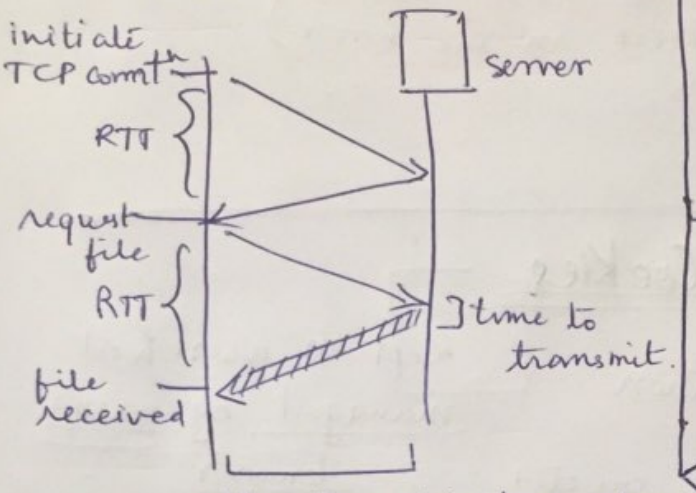


How does HTTP work?

hypertext transfer protocol
— client/server model



Round trip time (request → response ←) = 2



after the object reaches the client, connection is closed.

connection-oriented protocol ≡ TCP transport protocol will be used.

"Stateless" protocol
(last used is not stored ≡ everything is a new connection)

Non Persistent HTTP ≡
— Multiple objects require multiple connections.

Persistent HTTP ≡
— Multiple objects can be sent over.
→ server remains open for further requests.

Response time ≡
2 RTT + file transmission time

HTTP server closes the connection after the transfer is done and a new connection has to be established

HTTP Messages

Request — Response

- ASCII

Message —

GET / POST
- Methods
- URL
- Version

request line

Header lines

body — the objects you get back.

- Connection Type / Time ≡

HEAD — asks server to leave the requested object out of response

POST method — uploaded to server in entity body
in form filling (passwords, secure content)

DELETE — delete objects from server

GET method (input uploaded in the URL of request line)

PUT method — upload something

Response message

status line — HTTP/1.1 200 OK \r\n
→ protocol + status code + status phrase

HTTP response status codes:

→ 200 OK ⇒ Status phrase.
→ 301 Moved Permanently (later in the message, it gives the new address)
→ 404 Not found
→ 400 bad request
→ 505 HTTP Version not supported.

[the data ie is returned (current)]

Entity Tag – (there can be modifications to the object in the server)
↓
will be same
"if not modified"

User-Server State : Cookies

stores some information on the server

Kept on user's host managed by user's browser

for the first time site assigns → unique ID created to the user.

↓

in the backend database

in response the server sends the cookie file

Next time → host → access → server

usual http request
cookie: 1678

with the cookie.

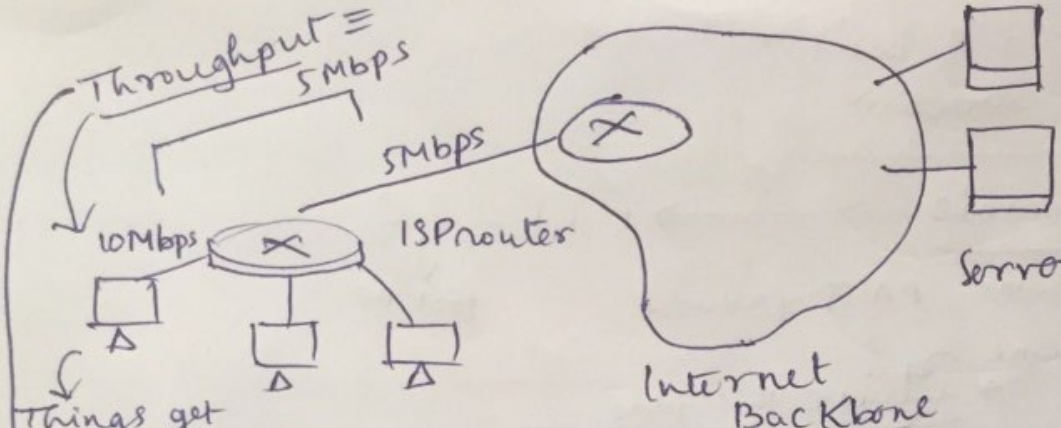
cookies
↓
in the browser folder.

HTTP: Stateless
converts the request → "state"

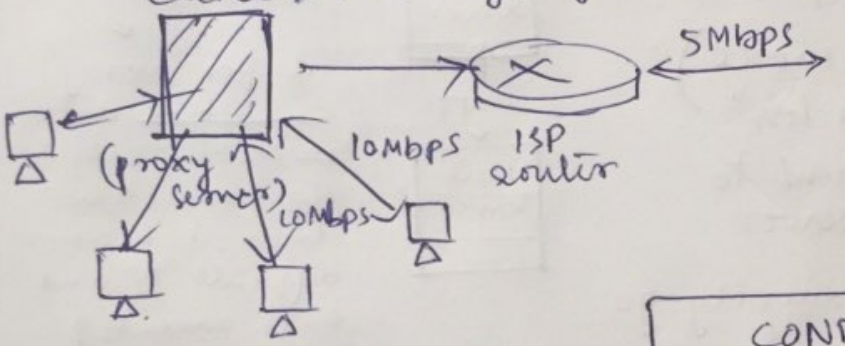
helps keeping state.

Web Caches – proxy server

aim: Carry out client requests without involving the main / origin server



↓ Solution

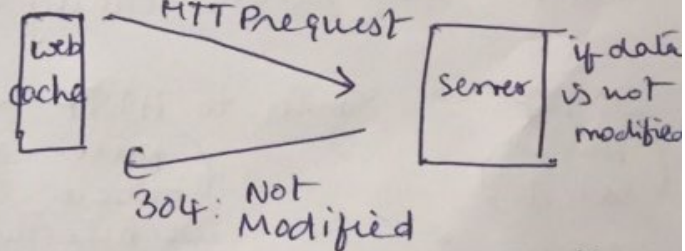


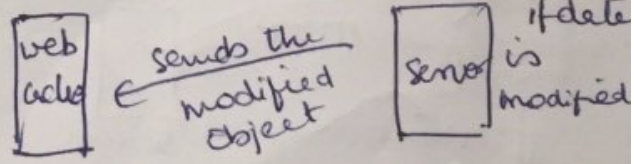
faster than the direct connection
[as distance access is decreasing]

if request → is object in cache ≡ cache returns object
– else cache – requests from origin server and returns it to the host

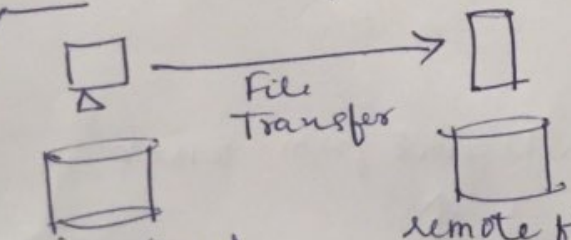
CONDITIONAL GET

cache is up-to-date or not

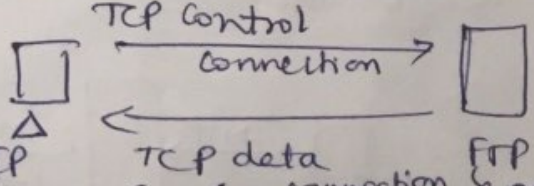




→ has "STATE"
FTP: file Transfer Protocol.



if connection is set up then we can browse the remote directories and files
Maintains 'state'



after the transfer is done, it gets closed but → Control doesn't gets closed

control connection } "out of band" connection

FTP Commands → FTP = 20 (Port no.)

- USER username, PASS password
- RETR filename → retrieve the files

Protocols for email — (SMTP, POP3, IMAP)

Simple mail transfer protocol

features

→ Persistent connection — SMTP: 25 (port no.)

→ State

→ TCP ? (reliable service "no data loss" from client to server)

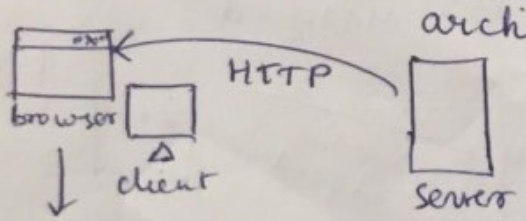
messages passed between client & server → 'ASCII coded'

data should directly go the destination without hopping onto any other mail server: security issues

Persistent Connection → if for every connection data packet transfer, the connection closes, it becomes difficult to send huge amounts of data.

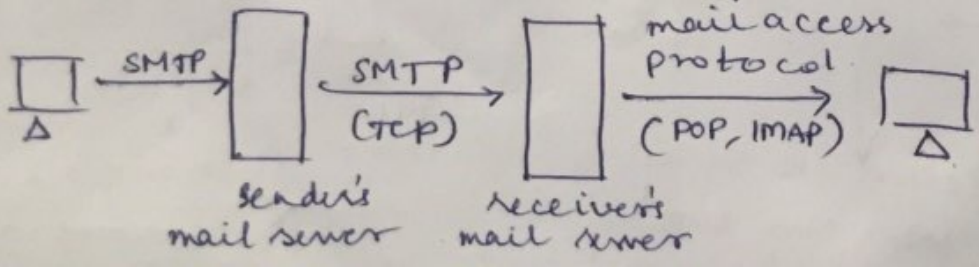
SMTP ≈ Similar to HTTP — both need TCP Connection & Persistent connection. (Only in few cases)

(push based) (pull based) architecture



pull the elements of the browser

retrieval from server



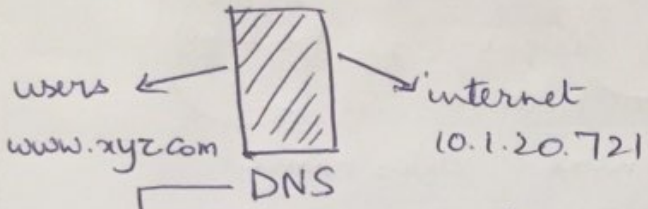
POP3 → downloaded into the system [data from server]

↓ Post office Protocol

→ cannot be accessed from other devices as server is emptied after data is ~~copied~~ dumped into the "host machine"

↓ host machine

DNS — Domain Name System



(directory service)

↓ distributed database

IP address (32 bit)

↓ used for addressing datagrams

↕ name of the site.

DNS services —

* host aliasing ↔ or mail server aliasing

alias names are given to few complex canonical names.

alias name	canonical name
www.ibm.com	www.ibm.---.com.---

Centralised DNS ✗

↓

— as if it fails, everything else fails

— database is placed somewhere so the distance varies.

— traffic volume is a problem.

Hierarchical DNS

↓

Root DNS

↓

.com DNS → yahoo.com DNS servers, amazon.com DNS server

.org DNS → pbs.org DNS servers

.edu DNS → pdy.edu DNS servers

Every local ISP has its own local DNS server

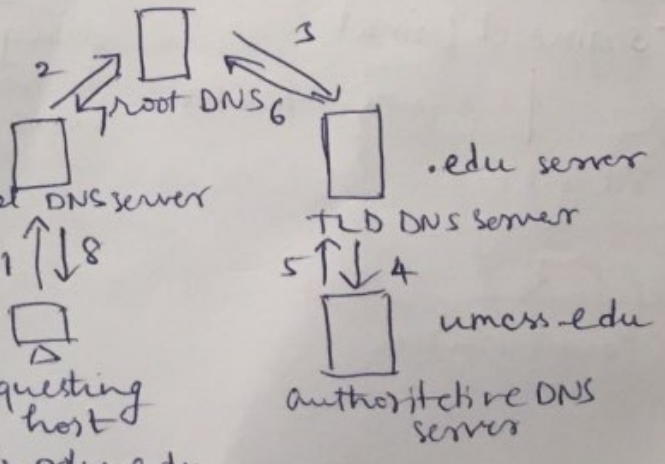
↓ which keeps cache

→ iterated query →

↓ ↑ no. of steps

one server does it's job and throws the request on some other server's table

[delegates to other lower hierarchical DNS servers]



→ recursive query —

— in this case, local DNS server is not being pinged unlike in the iterated query case.

DNS: Caching, Updating Records

stores resource records (RR)

RR (name, value, type, ttl)

resource record

time to live

type => A { name: hostname of authoritative server
value: IP address

=> NS { name: domain (eg. foo.com)
value: hostname of

=> CNAME { name: alias name
value: canonical name

=> MX => name of mailserver

TRANSPORT LAYER —

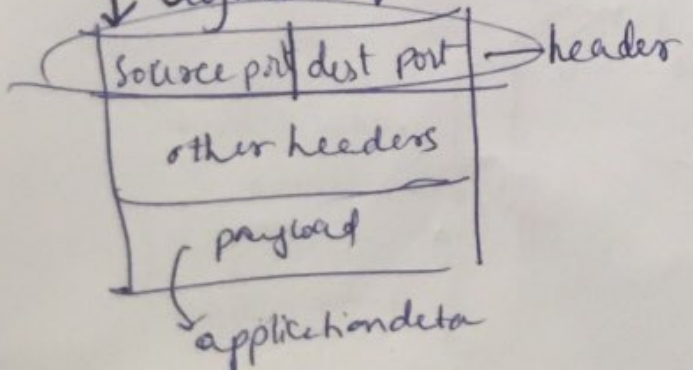
transport protocol

"send" side
breaks down messages → segments

"rcv" side
reassembling of segments → messages

Multiplexing vs DeMultiplexing
data from diff sockets and add ~~tran~~ network header
Socket → host

segment format



Not exactly reuse

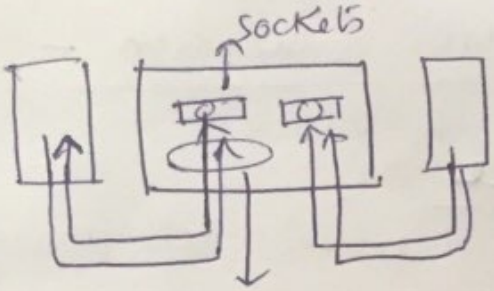
TCP
- Reliable (in case of data loss, retransmission)
- Congestion Control
- Connection-oriented
- slow
- Throughput guarantee

vs

UDP
- Fast
- Connection-less
- Simple

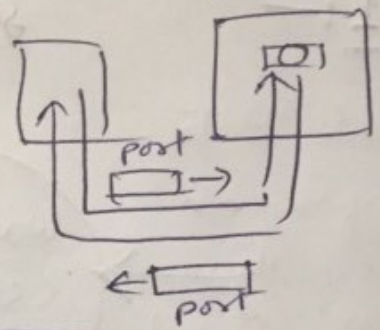
extension of "best effort" IP

doing its' best to transmit a packet.



TCP
MUX / DEMUX

UDP → Connectionless demux



UDP → used in DNS

→ User Datagram Protocol (UDP)

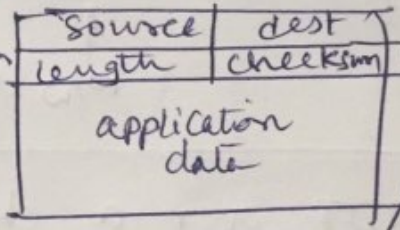
↓ because — fast

— No need for reliability
[as already reliable on local DNS]

— Routing Tables — updated periodically
↳ if there is loss, also, the info

— Reliability transfer
(— can be add to the application layer)

→ UDP SEGMENT FORMAT

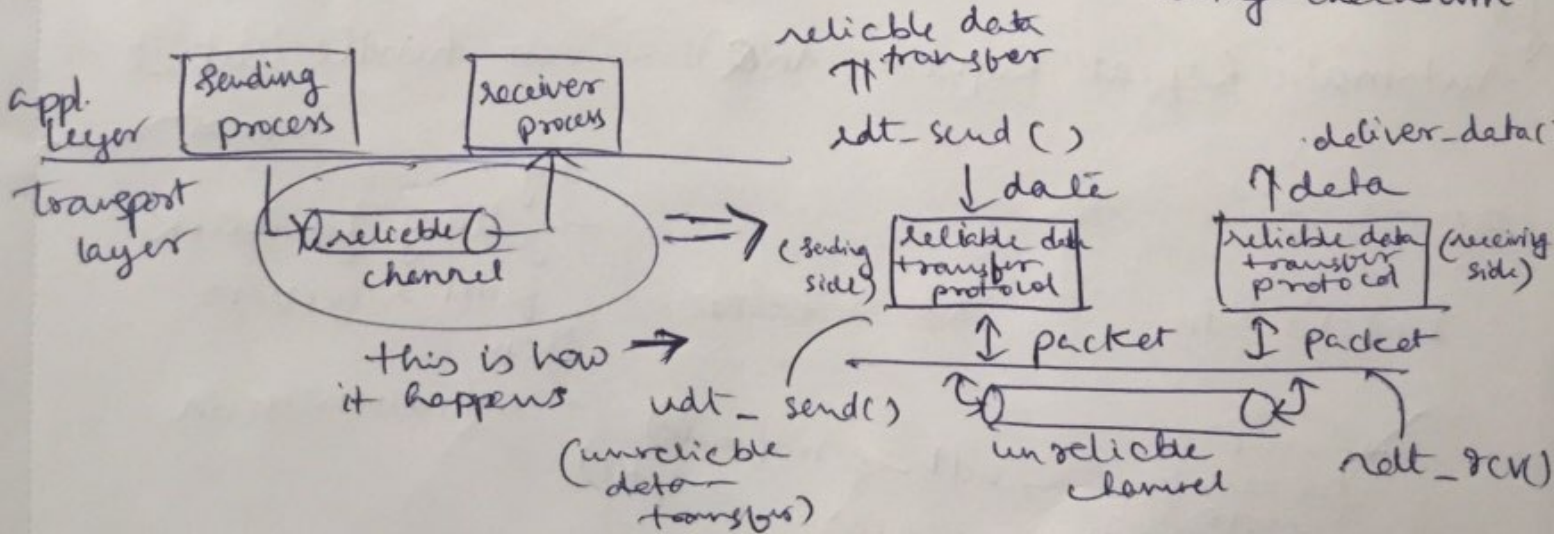


length of UDP segment

error corrections [while data transfer]

⇒ reliability checking can also be done using checksum

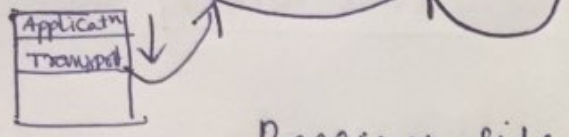
Principles of data transfer —



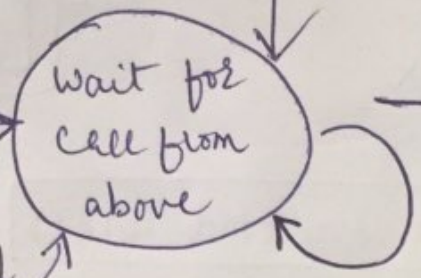
Finite State Machine: for Reliability Connection –

Case 1:–

Reliable Data Transfer over a perfectly reliable channel

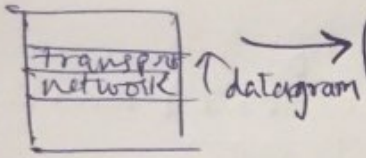


Sending side



reliable data transfer
↑
rdt_send (data) [appl^n]
pkt = make_pkt (data)
udt_send (pkt) [transm]
↓
unidirectional data transfer

Receiver side



receiving the packet from below level
rdt_rcv (packet)
extract (packet, data)
↑deliver_data (pkt)
transport layer

and then they are extracted

Case 2:– RDT over a channel with bit errors

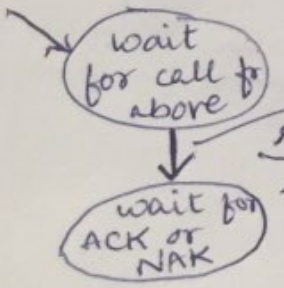
- "ACK" – without errors – acknowledgment packet
- "NAK" – with errors – negative acknowledgement

↓
Automatic Repeat Request (ARQ) – can handle/detect errors by
– getting feedback from receivers
then
– retransmission

RDT 2.0 –

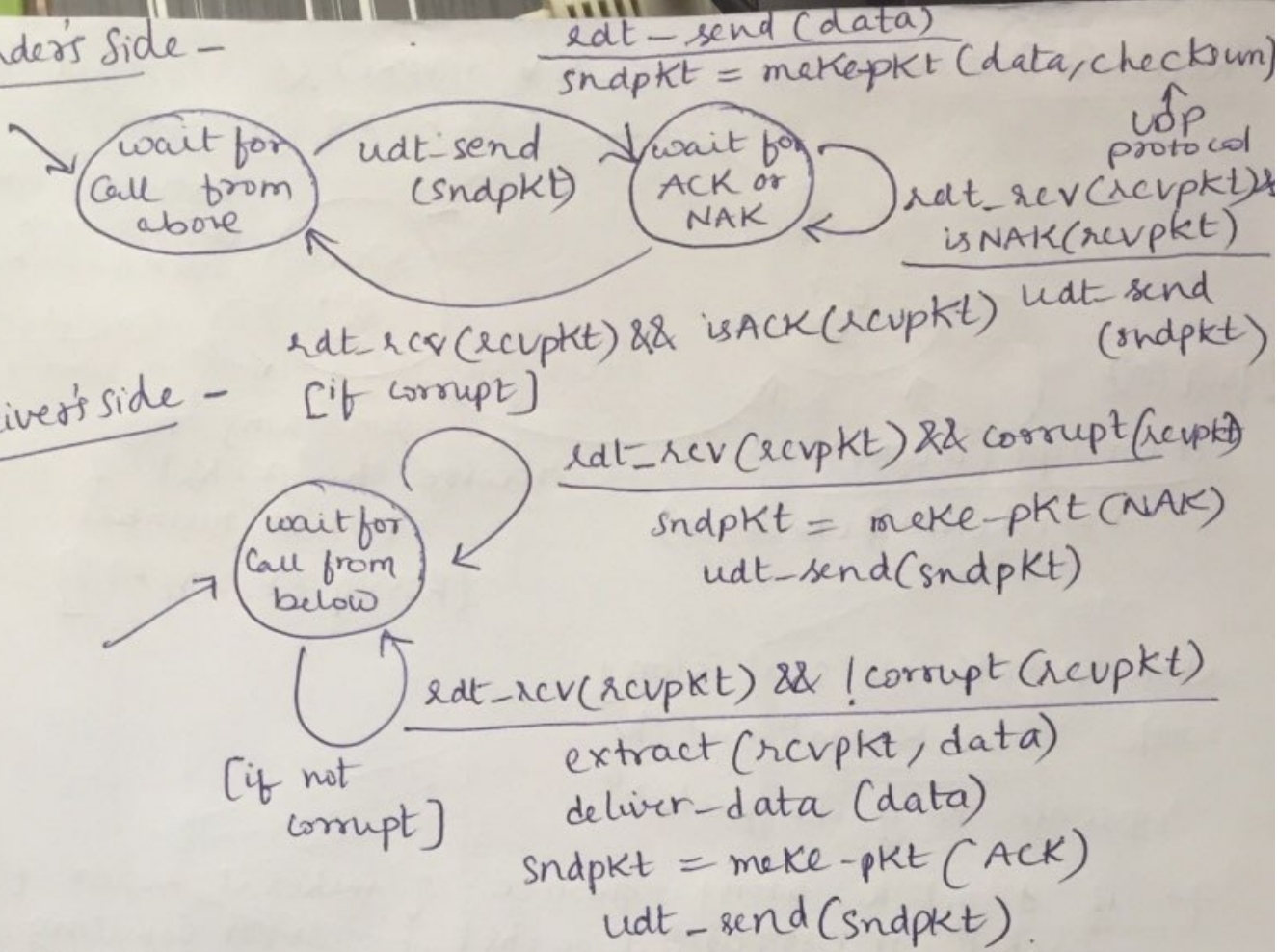
Sender's Side

~~Receiver's Side~~



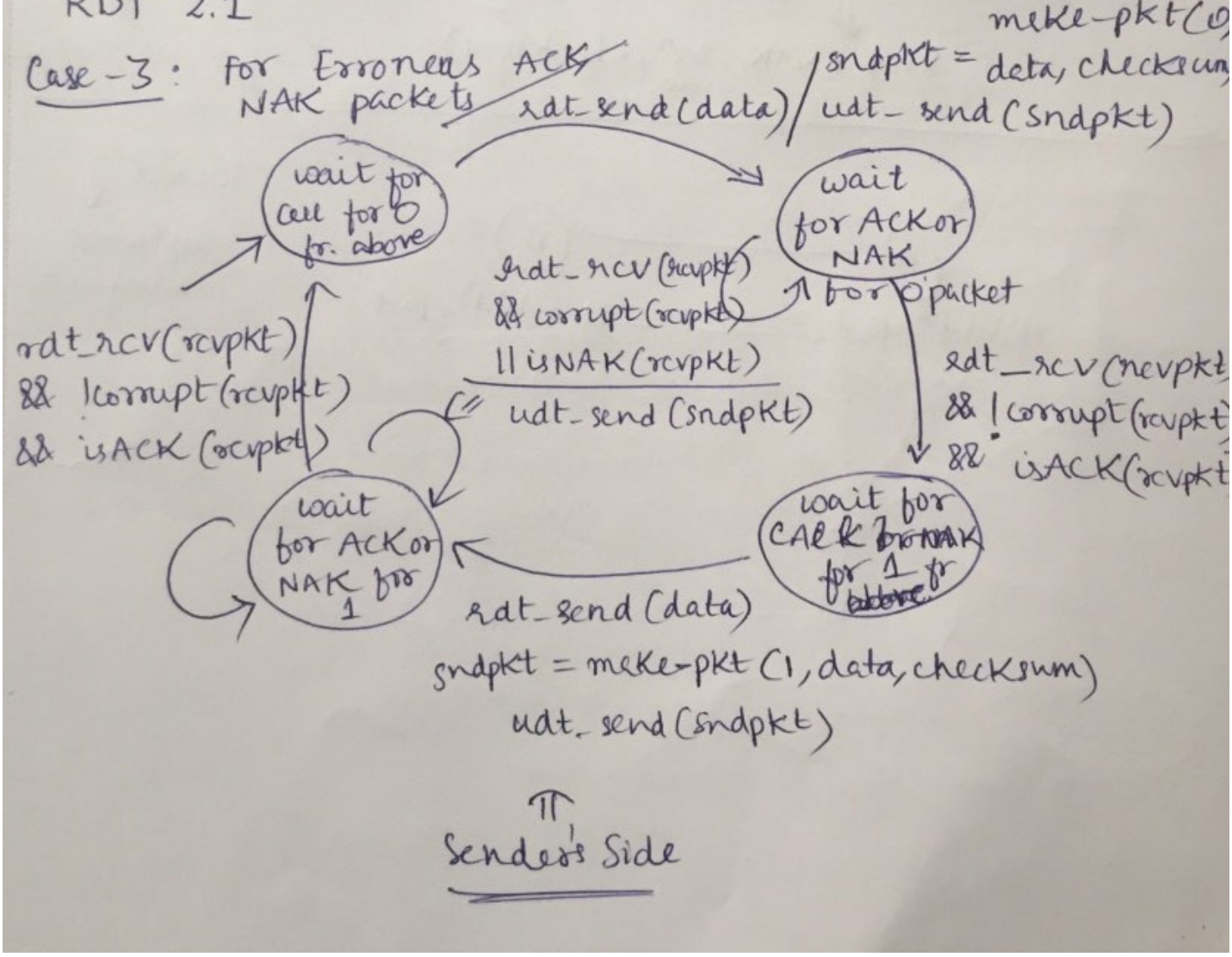
udt_send (sndpkt)
rdt_send (data)
sndpkt = make_pkt (data, checksum)
due ↑ to UDP protocol's error handling

Sender's Side –



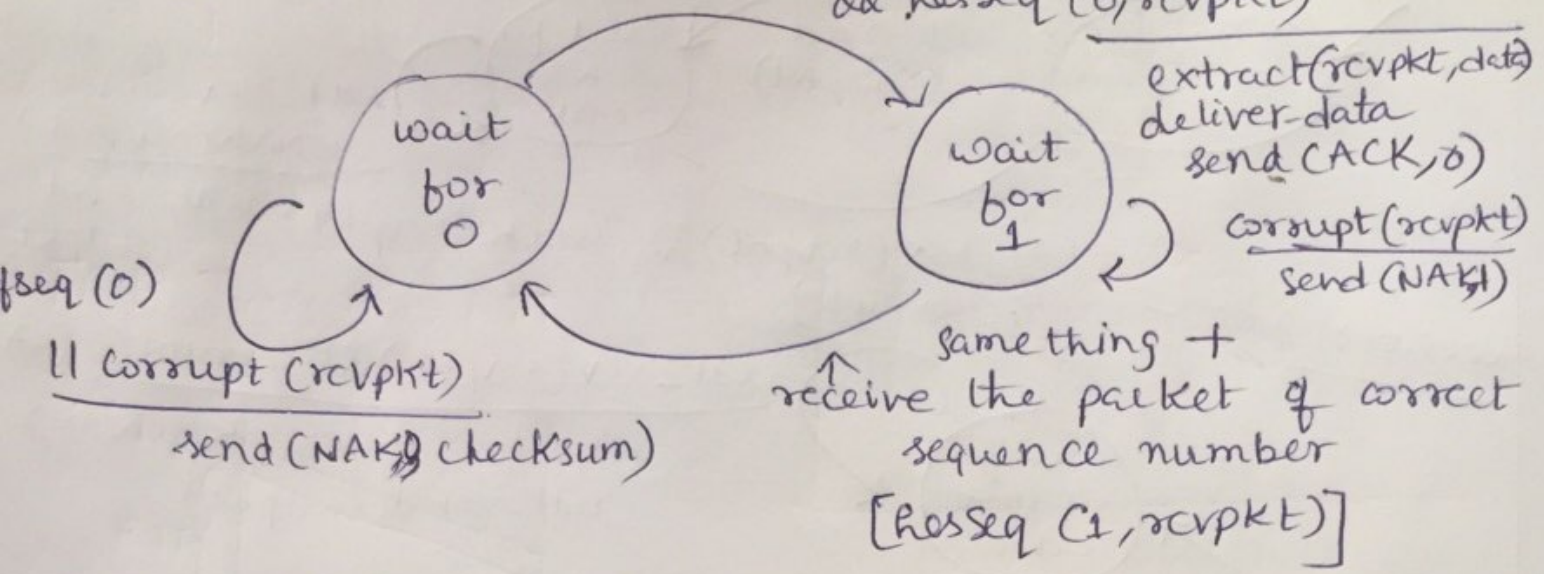
RDT 2.1

Case-3: For Erroneous ACK/NAK packets



Sender's Side

Receiver's Side ⇒

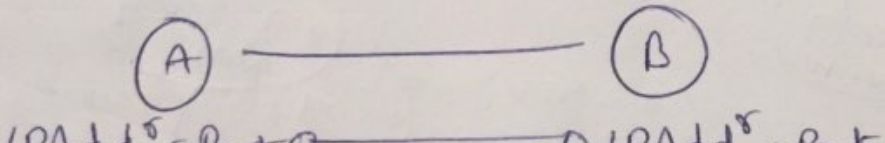


NAK & ACK are sent along with "checksum" and the "sequence no. of the packet".

† if I get a wrong sequence packet or corrupted packet } makes it easier for error handling

⇓

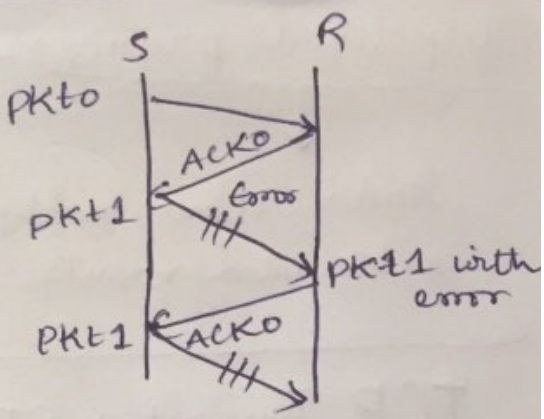
send (NAK, seq^n no., checksum)



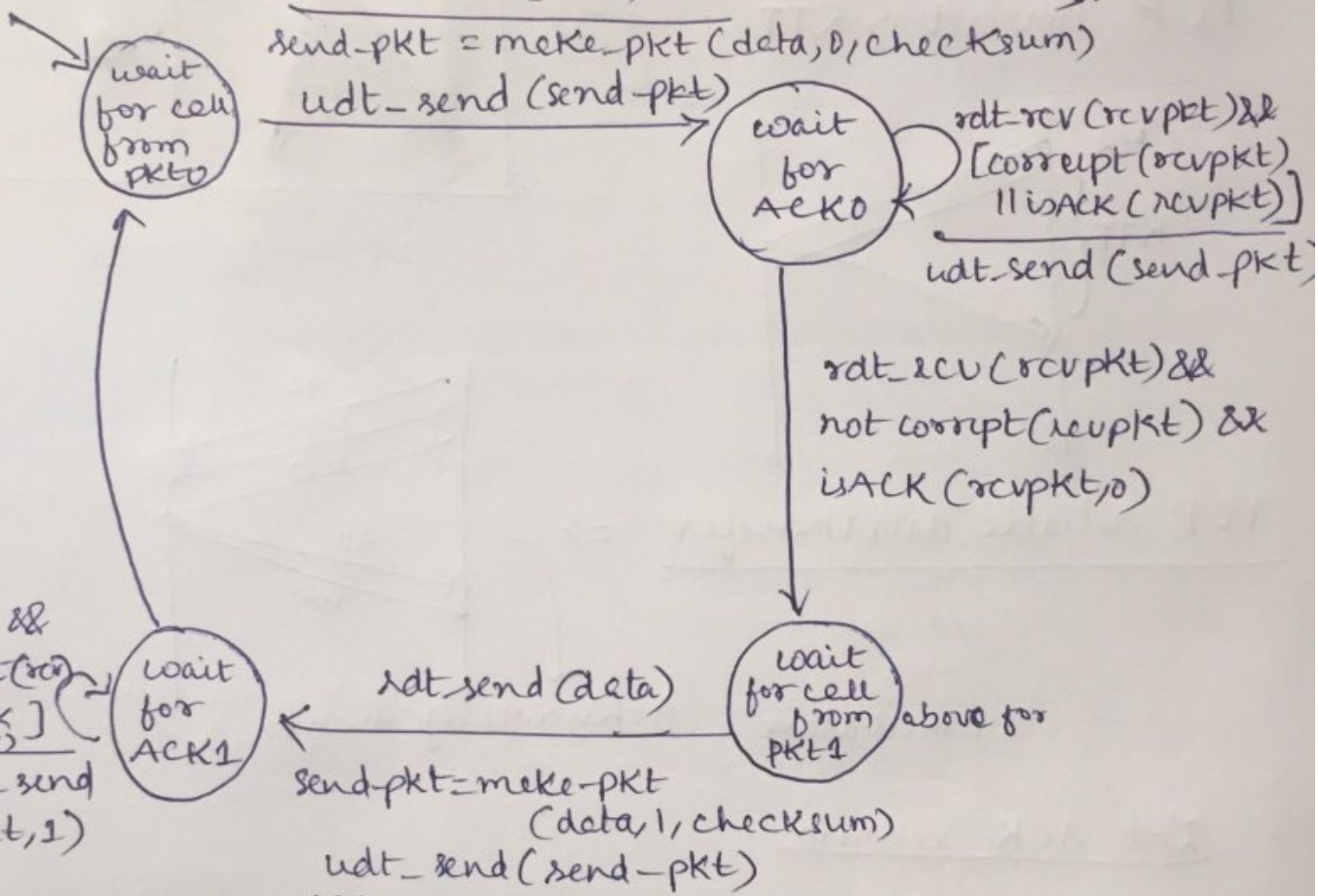
Socket ↓ Software interface

rdt 2.2 – Discarding NAKs

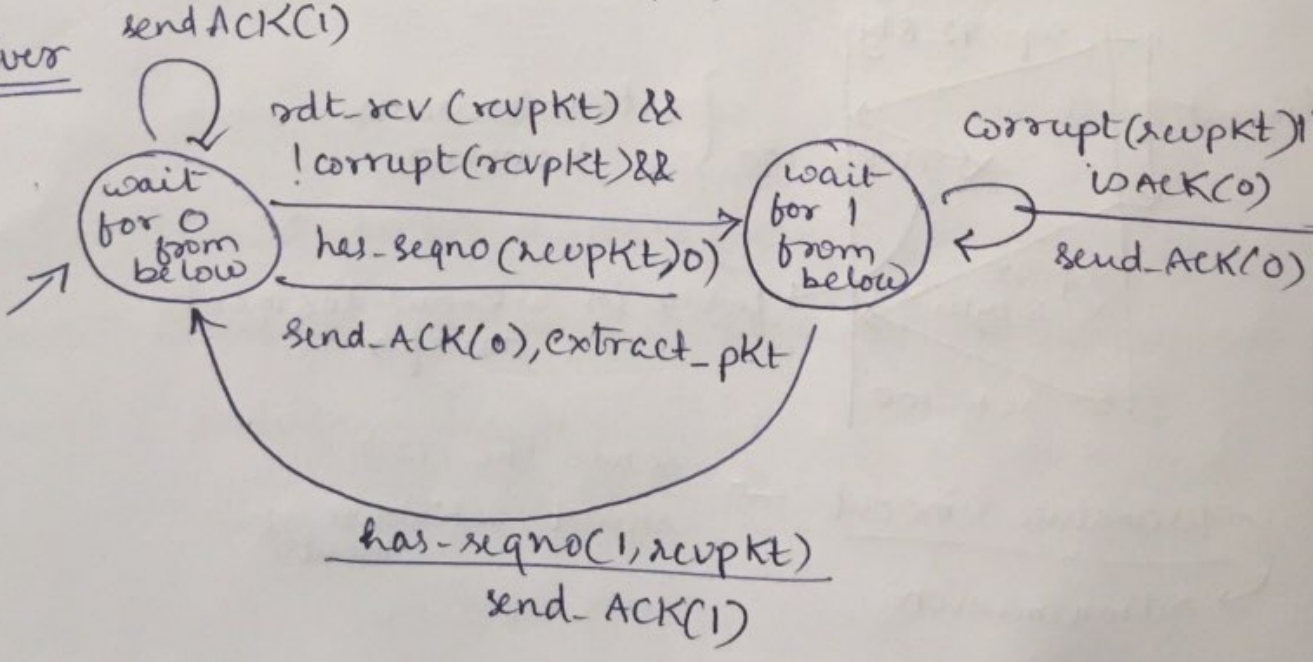
ACK of last pkt received OK



Sender



Receiver

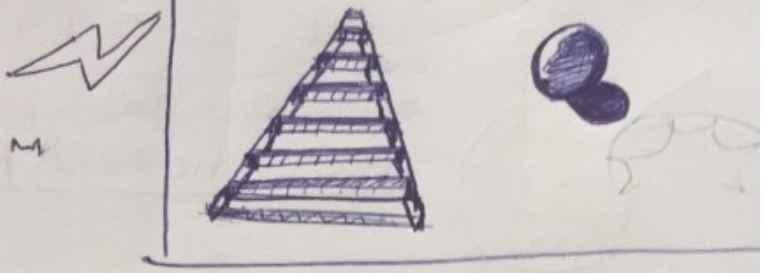


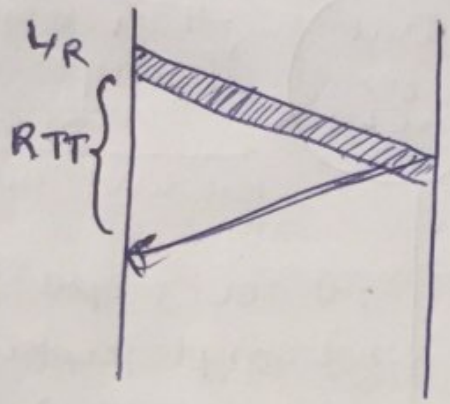
pipelined protocols → sliding window protocols

↓

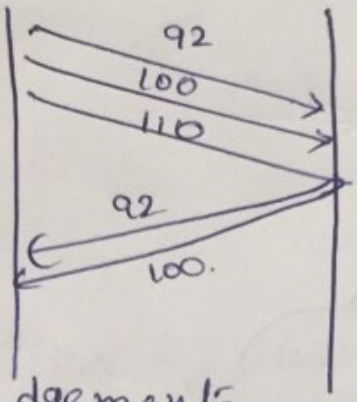
Sender allows multiple acknowledgements

TCP Timeout > RTT



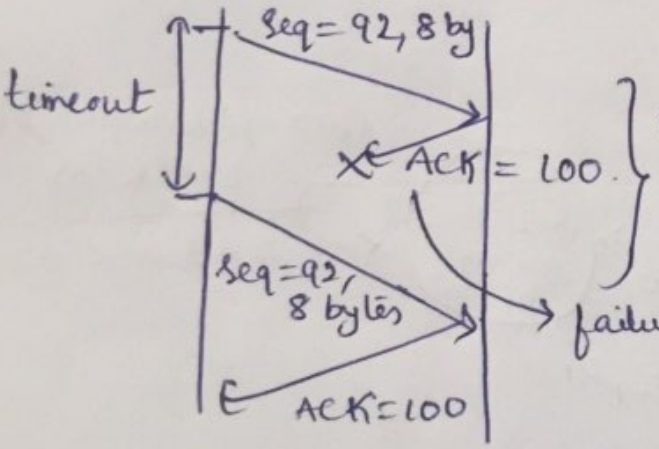


TCP reliable data transfer ⇒



Cumulative Acknowledgements

Lost ACK Scenario



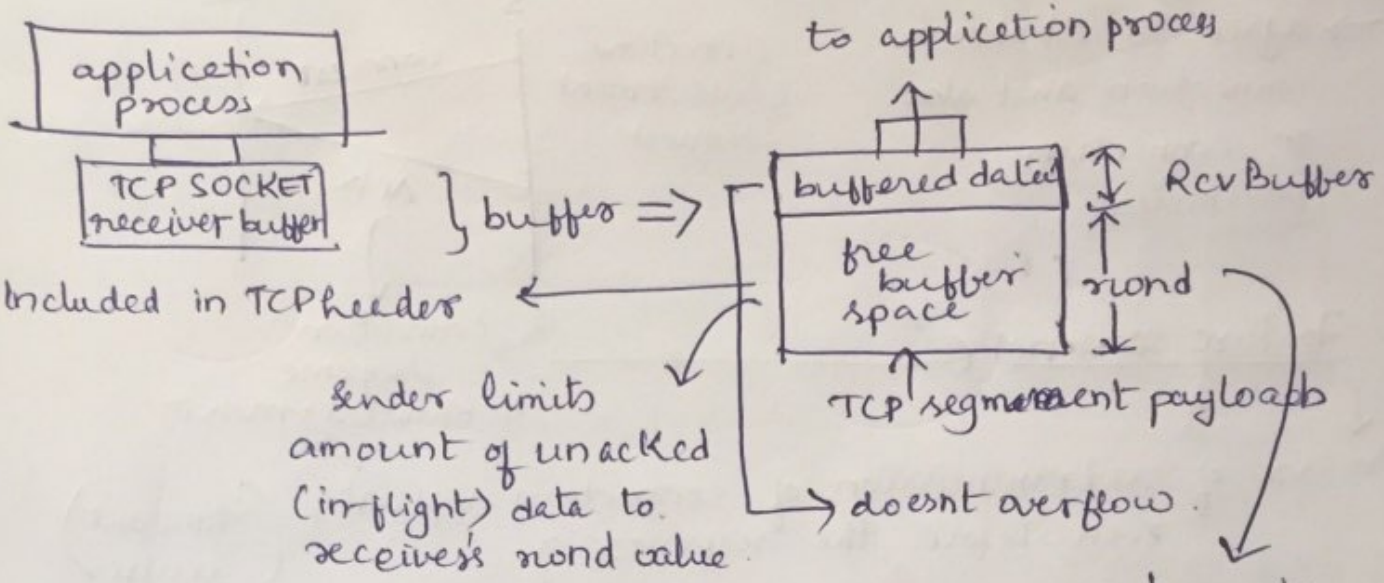
failure in acknowledgement transfer.

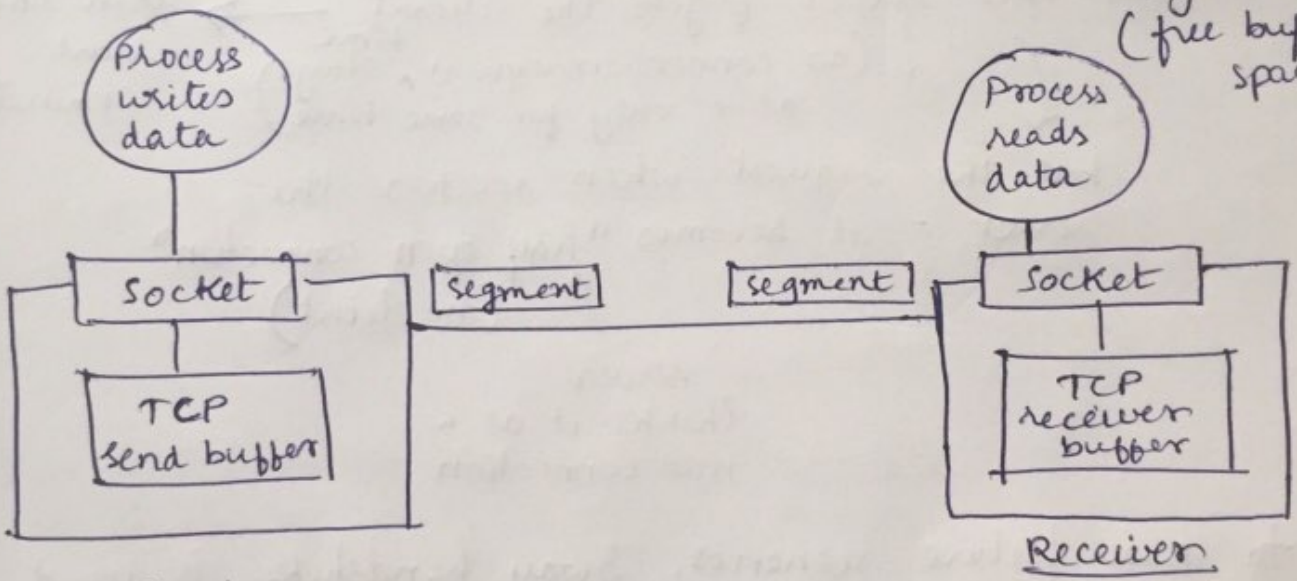
premature timeout → sends the last request acknowledgement

↓

retransmission

TCP Flow Control — Sender sends a lot of packets at a time, receiver has to control the flow.





Variables

→ LastByteRcvd
→ LastByteRead
→ Rcvbuffer
→ rwnd

At receiver, condition for no overflow?

→ LastByteRcvd − LastByteRead ≤ Rcvbuffer

rwnd = Rcvbuffer − [LastByteRcvd − LastByteRead]

At sender,

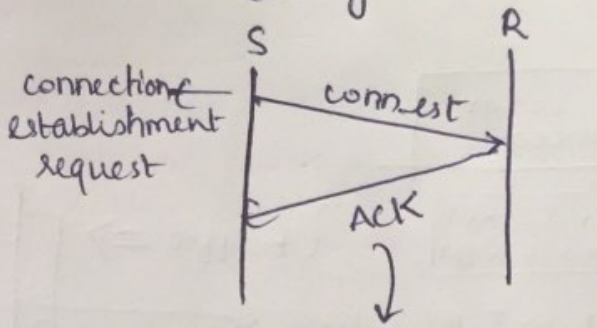
→ (LastByteSent − Last byte ACK) ≤ rwnd

Connection Management –

↓

before the handshake → agree to establish connection and also on connection parameters.

2way handshake



↓

connection response acknowledgement

Failure scenarios

↓

In case of retransmission of connection request, even before the response ↓ (timeout occurs)

in that time server forgets the client → then the client terminates

(as connection request stays alive only for some time)

↓

but the request when reaches the server ≡ it becomes "half open connection" (no client)

↑

server thinks it as a new connection

due to these failure scenarios, 3way handshake is used.

3wayhandshake

client sends a segment with a SYNbit = 1

↓

server returns an acknowledgement of SYNbit = 1

↓

client sends an ACK to "I that I'm still alive, you can send me response"

⇓

then it can send data.

TCP → close a connection (flag: FIN bit (finish)

↓

client's approval to close the connection

client can receive data but cannot send data

TCP Congestion Control →

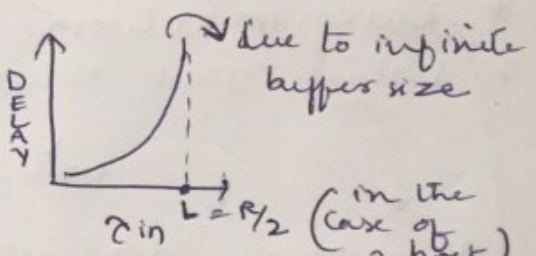
too many sources sending too much data too fast for the network to handle (like Facebook)

flow control → individual sender ↔ individual receiver

Causes
* long delay & (queuing in buffers)
* lost packets (buffer overflow at routers)

Congestion Control in TCP →

case 1: congestion scenario
→ more packets – more delay (since no retransmission and infinite buffer size) →



→ End to End Congestion Control (TCP)

→ Network assisted congestion control (feedback from the routers)

in TCP, sender should limit the rate at which it sends traffic → ? – cwnd – [contention window]

? to know whether a congestion happens or not

⇒ (lastbytesent – lastbyteACK) ≤ min(cwnd, rwnd)

"No congestion" constraint.

↓

"Loss event" →

Timeout or receipt of 3 duplicate ACKs from sender

> Change the "sending rate"

– last segments ≡ congestion
– ACKs ≡ N/W is delivering
– B/W probing

increased by a factor until the above constraint is valid.

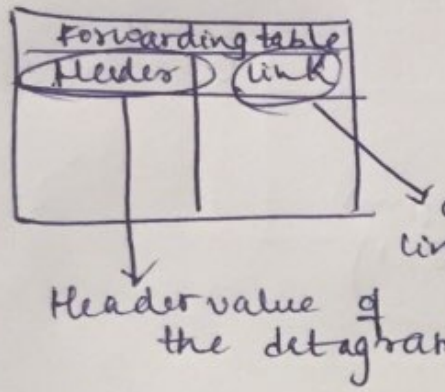
Network Layer →

segments → datagrams

present in every host and router

Functions —

* forwarding — next link (immediate one) (where to forward the data the next instance)
* routing — entire path



output link to which it needs to pass onto.

Header value of the datagram

* Datagram Network = connectionless → datagram is just pushed forward
* Virtual-Circuit Network ≡ connection-oriented
  - was used in telephone networks
  - VC number — identifier
    - packet has diff VCs on each link

routers: no "end to end" connection info

Forwarding Table ≡ of Datagram Network only the destination address (IP address in the header) & output link

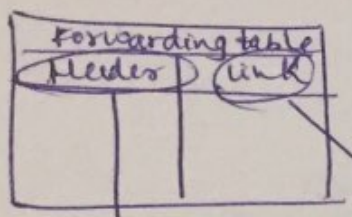
Network Layer

segments → datagrams

present in every host and router

Functions —

* forwarding — next link (immediate one) (where to forward the data the next instance)
* routing — entire path



output link to which it needs to pass onto.

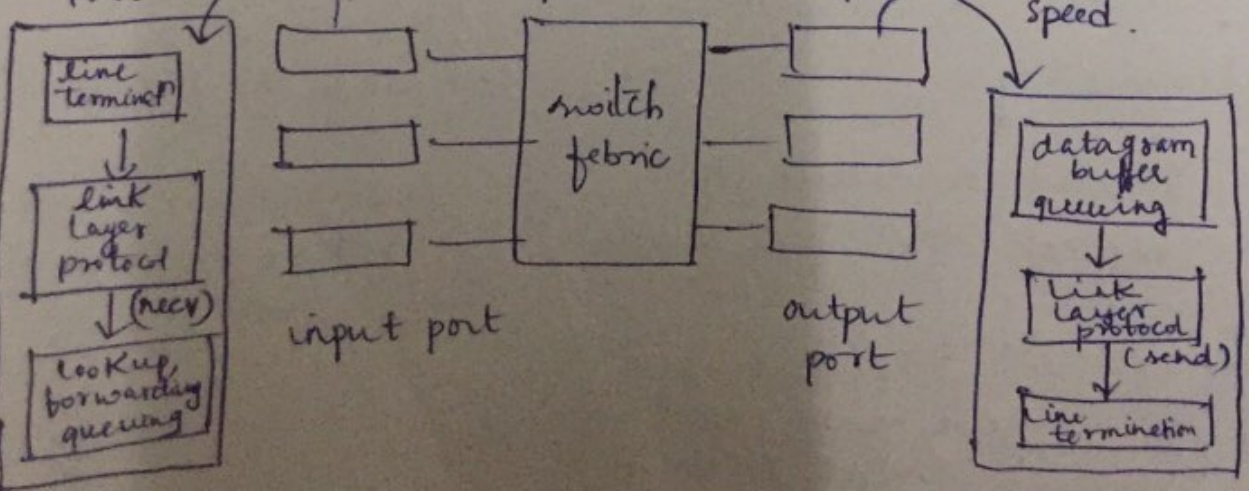
Header value of the datagram

* Datagram Network = connectionless → datagram is just pushed forward
* Virtual-Circuit Network ≡ connection-oriented
  - was used in telephone networks
  - VC number — identifier (packet has diff VCs on each link)

routers: no "end to end" connection info

Forwarding Table ≡ of Datagram Network — only the destination address (IP address in the header) & output link

Router Architecture
- routing algorithms / protocols
- forwarding packets from one input port to the other
  - input port: speed at which it performs operations on packets = line speed.



Switch Fabric → switching rate ≡ rate at which packets can be transferred from input to output ports

- → memory (copy of packet will be made in memory)
- → bus (common channel) → Shared bus ≡ Speed is limited due to bandwidth of bus
- → Crossbar switch (Banyan circuits)

OUTPUT PORT → Buffering *

→ Important.

No enough space in buffer → data loss due to congestion → packets get dropped in the buffers at both the ends.

Head of the line blocking (HOL)
→ datagram in front blocks the datagrams in the queue

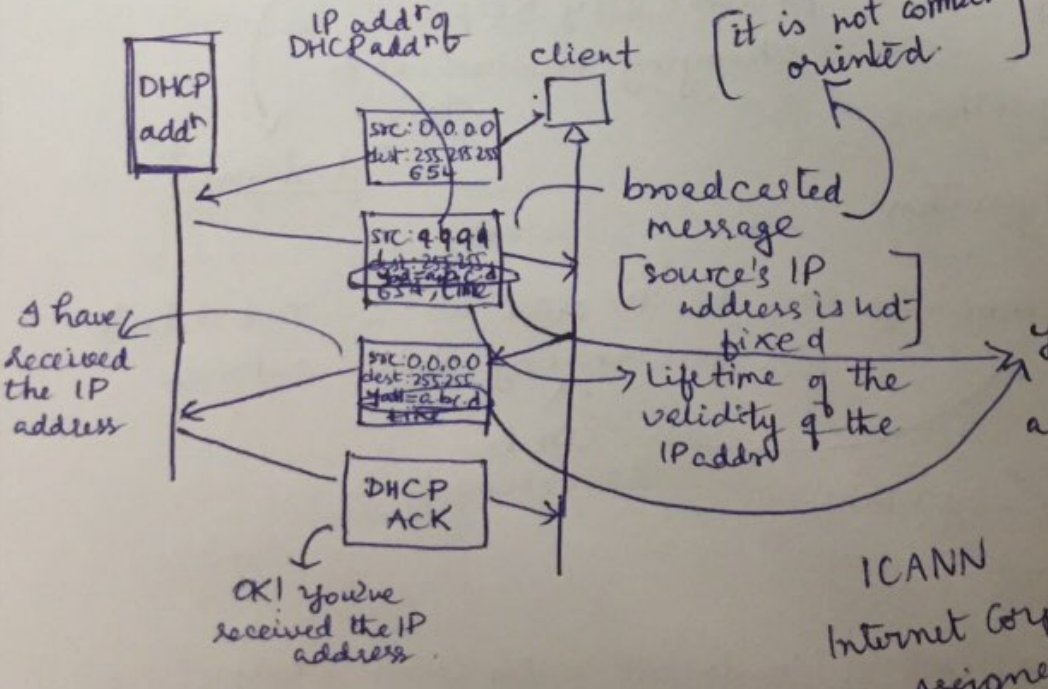
fragmentation - breaking down the data [flags & offsets]

subnet - link between routers (n/w islands (having similar IP addresses)

DHCP (Application layer)
↳ Dynamic Host Configuration Protocol.
→ provides IP address. DHCP ≡ UDP [it is not connection oriented]

0.0.0.0 → not given to anyone.

255.255.255.255 → No fixed dest.



ICANN
Internet Corporation for Assigned Names and Numbers

Interplay between routing, forwarding →

Routing algorithm (determines the end-end path through network) ⟶ forwarding table ↓ [local forwarding @ this router]

↑ more the bandwidth → ↑ high transmission

Network - complete graph (path from one node to another should exist) ↓ may or maynot be direct connection.

$c(x, x')$ ≡ Cost of the link (x, x') ↓ inversely related to bandwidth

Routing Algorithms ≡ Least cost path is found out. ☹

Network — static vs dynamic

static ↓ no need to change the network state regularly ↓ "link state" algorithms

dynamic (- traffic pattern keeps changing frequently) Periodically keep changing the network state. ↓ "distance"

- Dijkstra's Algorithm

Routing algorithm → can be treated as a substitute for forwarding table entries (can be determined)

"distance-vector" algorithm → Bellman-Ford

$d_x(y)$ ≡ Cost of least-cost path from x to y

$$d_x(y) = \min_v \{ c(x, y) + d_v(y) \}$$

$\{$ all the neighbors v of x

link gets broken = ∞] threshold time

Count-to-infinity problem
↳ ping pong effect of the wrong updation on all the other nodes in the network

datagram
↓
DATA LINK LAYER → "frames" are sent over → to the physical layer (error control)

datagram + H_e → frame

Ethernet
Wifi
Bluetooth

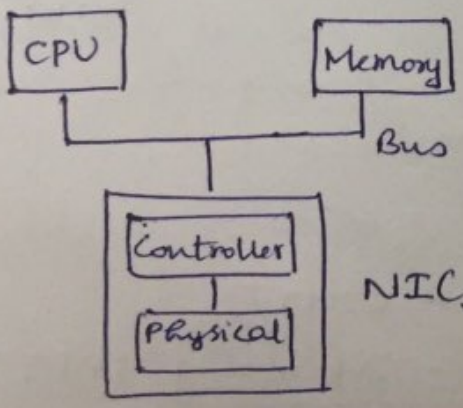
Channel access protocol +
error (detection & control)
Reliable data delivery
Flow control
} link layer

work of LINK LAYER

Channel access protocols change over different links

Half duplex – one at a time connection
Full duplex – both at a time connection
} link layer allows these kind of connections.

- Ethernet
- Wifi
- 4G
- Bluetooth.



NIC Adapter ≡ Id (MAC address)

sender

* encapsulate datagram in frame
* add error checking bits rdt, flow control etc

Receiver

* look for errors (detect)
* correct the error
* rdt, flow control (if poss)
* pass it to N/w layer.

Multiple Access Protocols →

> Point to Point Link (2 nodes are directly connected to each other)
> Broadcasting link
  ↳ Ethernet
  wifi — 802.11
  adhoc networks

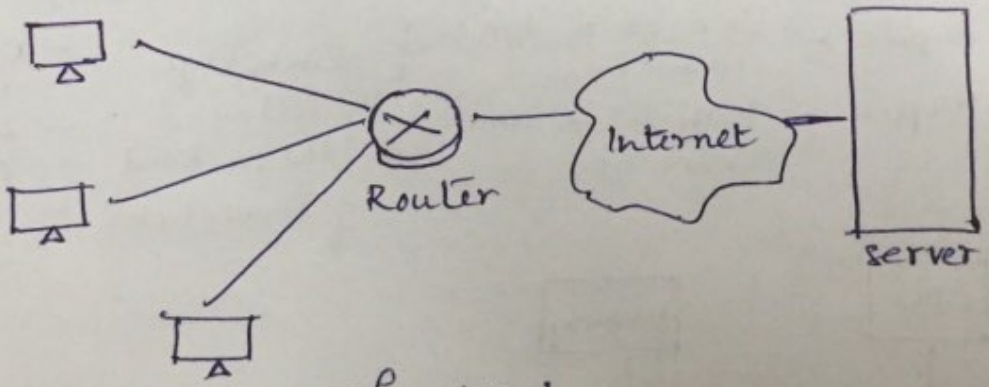
MAC Protocols →

> Random access
> Taking turns.

Channel Partitioning protocols →

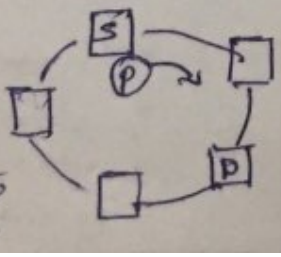
> FDMA (frequency) → divide the channel into various frequencies without interfering
> TDMA (time).

Local Area Network: (LAN)



Channel Access mechanisms
→ 2 classes

1.) Random access board — CSMA/CD
2.) Token-passing board — Tokenring, FDDI
  ↳ systems are connected as a ring topology.



- Keeps a copy and passes it along back to the source

Token Ring (disadvantage) – unnecessarily moving it across various systems as intermediary irrespective of destination. (passes onto other systems)

↑

better choice ≡ FDDI

Link Layer Addressing / MAC – address →

IP address ⟶ host

MAC address ⟶ adapter (physical device)

↓

every host

Features ≡

– 6 bytes address

– Hexadecimal notation, 1 byte ≡ pair of hexadecimal notations.

8E – FF – FF – FF – FF – FF

↓ 6 bytes

are not dynamic in nature

– MAC addresses are fixed.

for more address space, we moved from IPv4 ⟶ IPv6.

So what happens to MAC addresses?

IEEE manages the address spaces.

fixes the 24 bits (first) chunk of addresses that can be changed ~~by the~~ given to a company ≡ 224 addresses.

8E-FF-FF-FF-FF-FF

24 bits (IEEE)

↓

can be changed by the company

Translation from IP address to MAC address — ARP

↓

Address Resolution Protocol

Node A ARP Table.

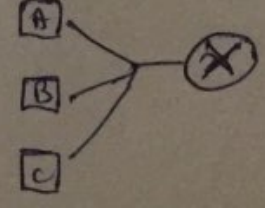
IP Address	MAC Address	TTL
10.0.1.2	8E-FF-....	200

why is there a TTL ?

↓

Dynamic IP addresses.

stores the same subnet addresses.



A → C & if C's information is not in the A's ARP table

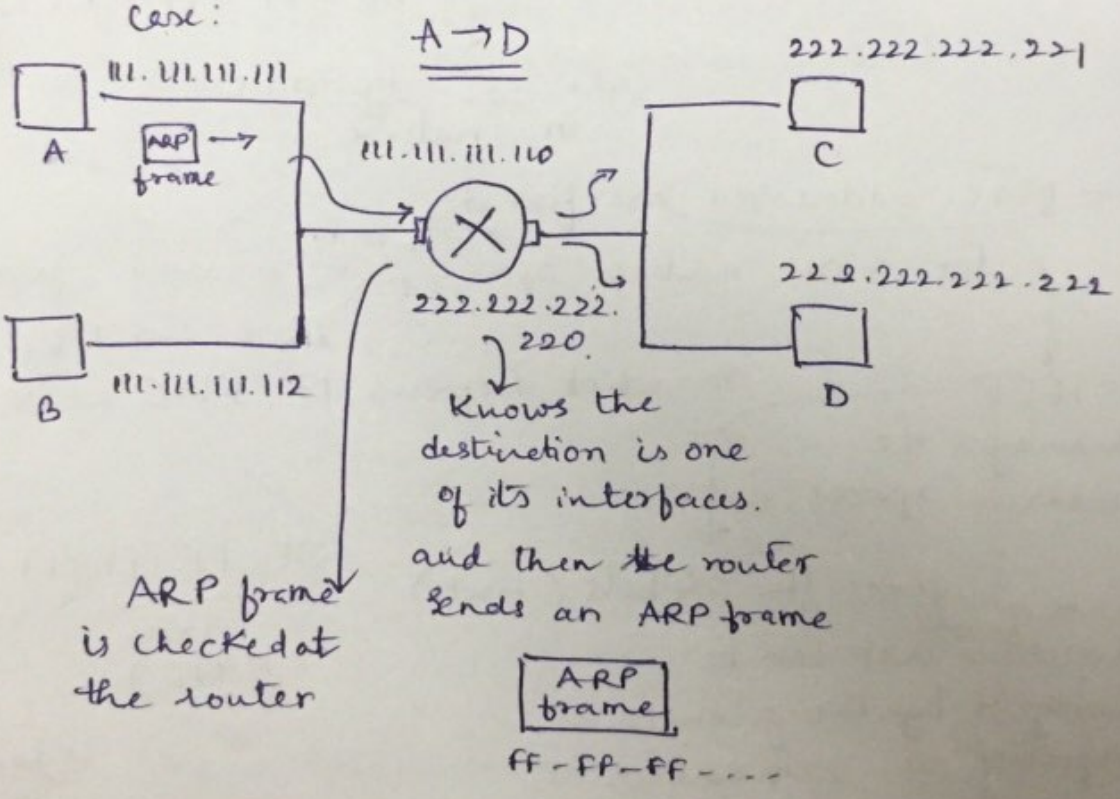
it has to an address resolution
has to make a frame.

ARP packet → to know the MAC address of the destination
frame → simply broadcast to every system of ARP packet

MAC address standard ≡ FF-FF-FF ....
(just like 255.255... in the case of IP addresses)

[if IP address is not the one required, simply discard else the destination responds with ARP response (MAC address) of the destination]

— Not in the subnet case:



Knows the destination is one of its interfaces. and then the router sends an ARP frame

ARP frame is checked at the router

ARP frame
FF-FF-FF-....

Ethernet:

46 - 1500 bytes

Preamble	Dest Addr.	Source Addr	Type	Data	CRC
8 bytes	6 bytes	6 bytes	2 bytes (layer protocol)		Circular Redundancy check

channel is free → send (transmit) and keeps sensing

else → just "backoff" (stops transmission & sends a "jamming" signal)

on sensing a collision

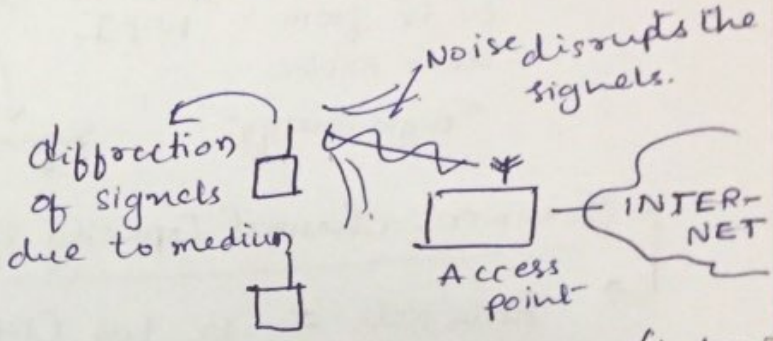
CSMA/CD ≡ IEEE 8023. (Protocol used)

Ethernet Cables → 10BASE-T → 10Mbps. → twisted pair

100BASE-T → Backend Ethernet.

WIRELESS NETWORKS →

Signals face-challenges
- Reflection
- Refraction
- Diffraction
- Scattering



≡ classification:

→ Infrastructure-based networks

→ Infrastructure-less networks

⇓

ad-hoc networks → NFT
- mobile Adhoc networks
- vehicle Adhoc networks.

there is a central entity (infra st.) that provides the base for the network.

Ex: cellular networks

communicate with the help of Base stations

provides a channel for communication

→ WiFi, Access Points.

every info goes through the Wireless Access Points

Wireless sensor network → Network of Sensors (wireless)

Main node responsible to carry out all the operations.

Challenges ⇒

* Multipath Propagation
* Decrease in signal strength
* Interference (noise due to traffic)

Antennas — capture the energy that these are receiving.

data should be received at a certain threshold value.

- Signal to Noise Ratio (SNR) = $\frac{\text{Signal}}{\text{Noise}}$.

↑SNR, receiving with higher energy at the receiver's end = decipher easily = better reception

- signal to (Interference plus noise) ratio (SINR) =

"low energy" → noise → can be from the medium.

Interference → can be from other devices "high energy"

$$\frac{P}{N+I} \approx \frac{S}{N+I}$$

S → signal strength

- Shannon's Channel Capacity Theorem =

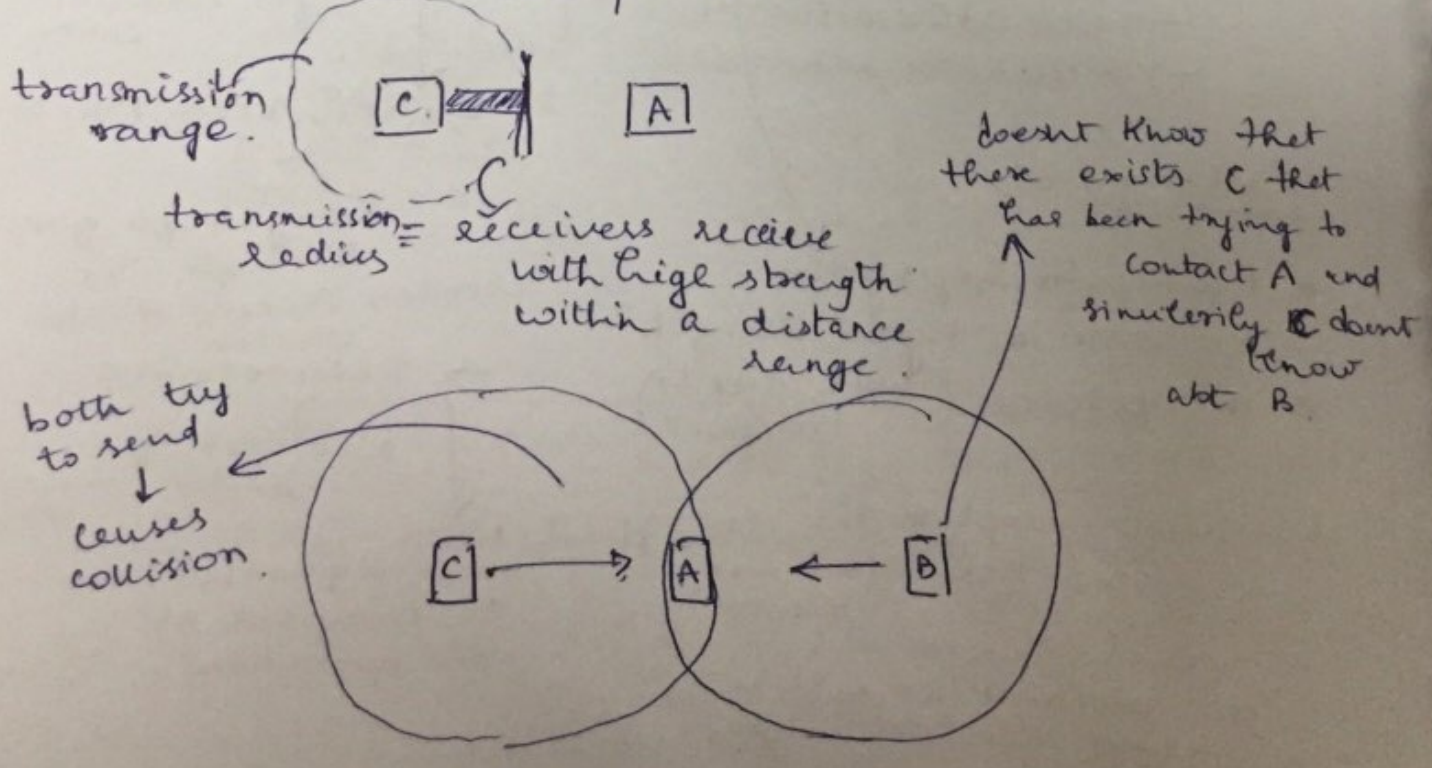
Data Rate = $B. \log_x (1 + SINR)$

x → changes with the bits being transmitted on the channel.

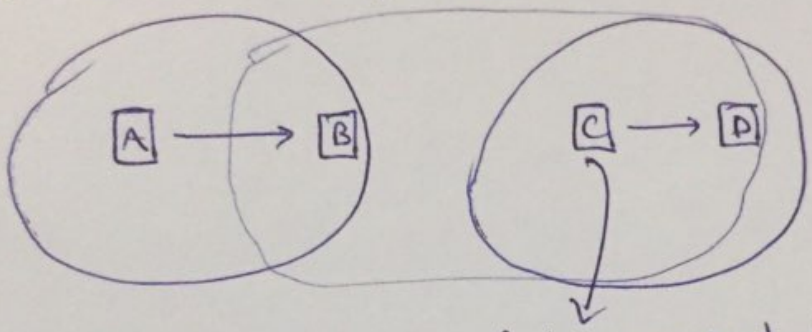
> Interference Issues

- Problems

→ Hidden-terminal problem.



→ Exposed terminal problem:



C is exposed to a terminal that is already busy in a channel.