Text Books:

1) Computer Networking — Top Down Approach.
(Edition 5)
(Kurose and Ross)
(Also p.p.t's)

---

*Communication Networks:

- Broadcast communication network
- Switched communication networks
  (Intermediatary switches, routers etc pass data from source to destination)
  - * Circuit switched communication networks
    (End to end resources are reserved for each call)
    (Time sharing or resource sharing can take place if same switch is used in link from source to dest.
  - Packet Switched communication networks
    (Packets share resources)
    - Datagram network
    - Virtual circuit network.

---

*Packets: Structure
- Header
- Data
- Trailer (optional)

Statistical Multiplexing: Sequence of packets entering & exiting is not fixed.

- Demultiplexing → tells which data is for the destination by reading the header.

⇒ Datagram packet Network:

- No setup & no route is reserved / dedicated. (Described no route in packet header)

⇒ Virtual circuit switching:

- A route is reserved but can be shared (Described route in packet header)
- This route is created on setup.

Protocol:

- Agreement on how to communicate.

ex: HTTP, TCP, SMTP, IP

** The Internet (An example of a network)

- transmission rate = bandwidth

⇒ RFC → Request for comments

⇒ IETF

1) • Client/Server model, host requests and receive services.

2) • ~~Peer~~ Peer to peer model, minimal or no use of dedicated servers (P2P networking) (Client/Client communication).

Communication:

Application Layer → Transfer Layer → Datalink Layer → Hardware

---

*Types of Internet Connections:

1) DSL: (Frequency modulated)

- A splitter splits data (frequency modulated) so telephone & DSL Modem can use the network.
- DSLAM central office receives this data and then sends it to ISP's or Telecom network.

2) Cable Network:

- Similar to DSL, but the TV's service provider provides the internet instead of DSLAM Telecom related offices.
- CMTS central office sends to ISP.

3) Home Network:

- Direct wired connection provided by the ISP. We can use a router to use multiple devices with internet.

---

*Transmission Media: (Across a network)

1) Guided → Twisted Pair (TP), Coax, Copper, Fiber Optic
2) Unguided → Radio, (Satellite) etc

---

* Internet Structure;

- Backbone of ISP's to which end users connect to.
- There are links b/w these backbones, "Peering links".
- IXP's are present in the middle to which these ISP's are also connected (not peering links). IXP → Internet exchange point
- ISP's are classified into Tier 1, 2, ... etc.
- Tier 1 ISP's are well connected and located worldwide.

—— x ——

Network Security Issues:

1) Distributed Denial of Service attacks (DDoS)
2) Packet Sniffing (Wireshark)
3) IP Spoofing (False source address)

—— x ——

Packet Delay & Loss;

- When router buffer is full, packets are dropped.
- If other packets are in queue, then delays occur.

—— x ——

Definitions:

1) Link bandwidth : maximum rate (bps) for sender to send data along link.

2) Propogation delay: delay for signal (light) to travel from source to destination

3) Packet transmission time: time for the sender to send all the packets

4) Queuing delay: time packet needs to wait since queue was not empty when it arrived.

5) Processing time: Time taken to process packet header etc.

Performance Metrics:

** Total Delay, $d_{nodal} = d_{proct} + d_{queue} + d_{trans} + d_{prop}$

d_{trans}: Time to start/select packet and sending it, completely.

If link bandwidth = B
packet length = L
$d_{trans} = L/B$

Note: If propogation medium is not defined, consider propogation speed = 2×10^8 m/s

---

Note: Queuing delay:

- If queue has X bits already present, then it takes X/B seconds to wait in the queue before our packet gets addressed.

Queuing Delay:

- Input data arrival rate = La
  Transmission rate = R

⇒ If La/R ~ 0 } Avg. queuing delay is very small.

⇒ If La/R ~ 1 } Avg. queuing delay is large.

⇒ If La/R > 1 } Avg. queuing delay tends to ∞.

**

——x——

1) Jitter:

- Variability in delay.

2) Round Trip Time: (RTT)

- Two-way delay from sender to receiver & back

3) Bandwidth Delay product:

- Product of bandwidth & delay "storage" capacity of network.

——x——

* Little's Theorem:

- Packet arrival rate $= \lambda$
- Mean delay of packet $= d$
- Mean no. of packets in system $= \lambda d$

—— x ——

* Throughput: (Rate of transfer)

- Bottleneck link determines the throughput of the system.
- No. of bits successfully sent / time = Throughput
- Link utilization = (throughput of connection) / link rate.

—— x ——

Note: We can test/evaluate these metrics by either pinging actual websites or using network simulators

—— x ——

* File Transfer: (Sending Packets)

1) Point to Point:
   - At each step, check for correctness
   - Timeouts, then retry.
   - Hard / Time taking / Inefficient
   - Failure, redo only last step.
2) End-to-End:
   - Only sender/receiver do the checks.
   - Checksums.
   - On failure, redo from beginning
3) Both: • Perform partial checks in point-to-point & verify checksum at end.

* Applications:

1) Client-Server:

- Servers → permanent IP addresses
- Clients → connect to servers, can have dynamic IP's.

2) P2P:

- Peers → talk to each other, dynamic IP's and ↳ management gets com

3) Hybrid:

- Uses client-server for some activities & P2P for others.
- ex: Skype, Instant Messaging.

—x—

* Sockets: (Doors b/w Applications & OS trans mecha

- Processes send/receive messages to/from its socket.
- Before data is sent through a socket transfer protocol, parameters etc can be

—x—

Note: Sockets ~~get assigned to physical~~ is the combination of port on the system & IP address of system.

—x—

Note: RFC's describe/define protocols.

—x—

*Note: Apps need transport services
→ data integrity } Audio etc can afford some loss.
→ timing
→ throughput.
→ security.

———x———

* Internet Transport Protocol Services:

1) TCP: → Slow!

- reliable transport → files etc use this
- flow control
- congestion control
- connection oriented → setup b/w client & server

⇒ No timing guarantee!

2) UDP: → Fast! } Used in real time applications.

- None of the above features are provided.
- Sometimes gives timing guarantee?

———x———

* Web Application Layer Protocol:

I) HTTP: (Port 80 used)

- client/server protocol.
- uses TCP for transport.
- * It is stateless → past client requests are not stored.

a) Persistent:

- Each TCP connection can be used by any number of objects.

b) Non-Persistent:

- Each connection only used by one object and then closed.

⇒ Non-persistent HTTP,

Response time = 2RTT + file transmission time
per object request
↓ Initiate connection + Request

(RTT = Round trip time for packet)

⇒ Persistent HTTP,

Response time = 2 RTT + file transmission initially
↳ RTT + file transmission later on.

—— x ——

i) HTTP Request:

- First line is the request line followed by header lines.
- Each header line followed by "\r\n" (carriage return & line feed characters)
- If "\r\n" is at beginning of the line, then it denotes end of header.
- After the header, we have the body.
  - GET: url should contain the parameters needed.
  - POST methods: form input etc can be used.

(GET, POST, PUT, DELETE, HEAD → Request types in HTTP)

ii) HTTP Response:

- First line is status line.
- Header lines of similar format as request (May contain different contents).
- Body followed.

* User-Server state (Cookies)

- A cookie is generated & stored on websites backend, and we also store the cookie ID on our browser
- When we visit the website again, then while sending the request, we also send the cookie ID, and the website gives us personalised content.

—— x ——

* Web Caches (Proxy Server)

- Similar to normal caches → we add a proxy server in between
- When a HTTP request is sent, if object is in cache, it returns it else it requests object from origin.

⇒ Multiple clients can also use the same proxy server.

⇒ The ~~client~~ cache (proxy server) acts both as a client & a server.

* • Conditional GET:

⇒ This would send a request to the origin, and if our proxied version of the webpage is updated, we get a 304 'Not Modified' response. → Negligible transmission for header response.

⇒ Otherwise, we get back a response with all the data.

—— x ——

* II) FTP: (Port 21) & (Port 20)

- client/server protocol
- Port 21 used for control connections.
- Port 20 used for data connections.
- FTP maintains state unlike HTTP.

—— x ——

*Email: (Uses SMTP protocol to send emails)

Note: Text

- SMTP uses TCP as transfer protocol. → Port 25

*1) Mail Server:

- Mailbox: Contains incoming messages for user
- Message Queue: Outgoing (to be sent) messages

⇒ A mail server contacts another mail server as a "client" or as a "server".

—x—

Note: SMTP uses 7-bit command messages & gets responses. (ASCII)

Note: ⇒ Connection: Handshake, Data transfer, Closure

—x—

2) User Agent:

- Uses many other protocols to open/read mail of a given user that is sent from the mail server
- We send mail via SMTP itself & server

⇒ SMTP uses persistent connections. (server is also SMTP)

⇒ SMTP can send multiple objects in a single multipart message unlike HTTP.

* Mail Access Protocols:

- POP, IMAP etc are used on the user agents to view the messages.

⇒ POP → Authorization phase & Login
→ Transaction phase & Retrieval of mails list & downloading individual each mail.

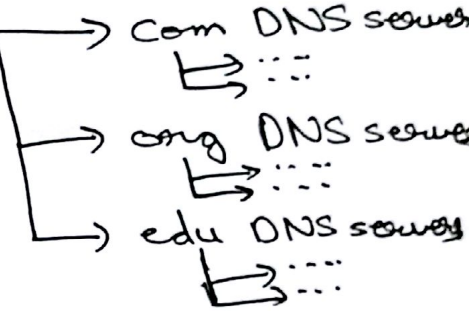
DNS (Domain Name System)

- Name servers store a distributed database of mapping b/w IP to website name (host name) & vice versa.

⇒ DNS is an application level protocol.

- DNS also does host aliasing. (URL shrinking etc)
- Load distribution, where a single host name could be using lots of servers on different IP's.

⇒ DNS Servers hierarchy (Root DNS servers)



⇒ Every ISP, Institute has a Local DNS server which is at the lowest level in the hierarchy.

- DNS requests from Local DNS to other DNS servers can be iterative or recursive (Iterative is when local DNS does all the work talking to all other DNS servers) (Recursive, the local DNS root does the hard work and gives the exact IP)

? DNS Records: (Name, Value, type, TTL) } (RR) Resource record format stored in DB.

1) type=A
2) type=NS
3) type=CNAME
4) type= MX

- DNS also uses request/response methods to communicate (slides for format, 2–76)

*P2P Architecture:

- No always-on server
- Peers connect with each other & have dynamic IP's which can change.

⇒ ex: Bittorrent.

Note: i) In a Client Server architecture, N ** peers downloading from 1 server.

⇒ Server Upload Cap = u_s

∴ NF/u_s = Time for server to upload (F → Filesize)

⇒ Client download is F/d_{min} (d_{min} ↑ Download of peer.)

∴ $D_{c-s} \geq \max(NF/u_s, F/d_{min})$

* ii) In a peer to peer architecture, F * is sent to N peers. (Server needs to send the first copy)

$D_{P2P} \geq \max(F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i))$

↑ Total upload is NF, since N peers download the file, and this is shared between all the clients/peers.

∴ We can see, as $N \uparrow, \Sigma u_i \uparrow$, so its good.

⇒ P2P architecture is a lot faster compared to Client server architecture.

- In Bittorrent,
  ⇒ Files divided into chunks of 256 Kb

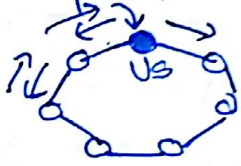
* ⇒ Tracker keeps track of peers currently connected
  ↓
  A Server

- Torrent is a group of peers exchanging chunks of a file.

* Requesting & Sending Chunks:

- To know whom to request, we use the distributed hash table
- We request chunks which are rarest & which we are missing first.
- We send chunks to top 4 peers sending us chuks. Every 30 seconds randomly pick another peer & send chunks. (Every 10 sec, repick top 4).

** Distributed Hash Table:



To get hash value

- Every peer stores info about another small number of peers (Distributed)
- Key-Value hashes used to figure out which clients have which parts of file.

* ⇒ Key is the file hash, value → peers who have this. (their IP's)

(Circular checking is used to get value of key if we don't have it, (predecessor, successor)

- Each peer keeps IP of , so we can ask them for hash value, they recursively ask till get.

⇒ But this can be slow, so we use shortcuts (we remember some other IP's and ask them etc)

* Also to handle peer churn (when peers leave), we store 4 IP's of 2 adjacent neighbours & their neighbours, so we can connect to the next neighbour if our adjacent neighbour leaves.

Note: Once we get the hash value, we know who all have the file chunk needed, so we start requesting them.

** Transport Layer: *

- Connects Application & Network Layers.

1) TCP → reliable, in-order (Connection oriented)
2) UDP → unreliable, un-ordered.
3) Pipeline → reliable (connection-oriented)

* Multiplexing / Demultiplexing:

- Multiple messages from sockets get transport headers ⟶ Demux headers to deliver to correct sockets.
- TCP/UDP segments have source/dest ports mentioned, and each ~~Transport header~~ datagram gives source/dest IP's, and a single datagram carries a single segment.

Note: UDP, just send data from 1 socket to another. Each socket is just IP + Port.

---

* Connection Oriented (TCP): (Reliable data transfer) (RDT)

- TCP socket → Source IP/Port
  → Dest IP/Port

⇒ So each socket links two ports only.

- A connection creates a new socket when getting established.

Note: User Datagram Protocol (UDP), RFC slides

- No congestion control } So its fast
- No connection etc
- Checksums are sent along to check if data sent is correct

- In TCP, any unreliability due to network channel, then TCP raises an error, asks sender to send again. If data is correct, then its passed on to the application.

⇒ Use finite state machines to describe TCP (slides)

Error Handling:

- ACK's → Acknowledgements
- NAK's → Negative Acknowledgements. } From dest to source.

⇒ If ACK's or NAK's are corrupted, sender sends packets with sequence numbers, so destination can discard duplicate packets.

↳ sequence numbers 0 & 1 are used. (slides)

Note: In RDT 2.2, we only use ACK's. But the ACK also carries latest successful packet.

---

* Data Loss: (RDT 3.0)

- Sender waits till timeout for acknowledgement and resends if no ACK is received.

——x——

* Pipelined Protocols:

- Sender sends multiple packets, and waits for an acknowledgement.
- Sender & Receiver both store 2 buffers. (With windows for each)
  ⇒ i.e. only one RTT delay for the k packets sent.

*1) Go Back N:

- Sender sends N packets, reciever sends acknowledgement after receiving some timeout t'. The timer starts again when next packet is received
- So if ack is sent for mth packet, then all packets upto m have been recieved.
- Keep timer for the set of N packets, on escpire of timer, send N packets again.
- Ack is sent for the highest inorder packet number → since all packets till that have been recieved correctly.

* 2) Selective Repeat: → To improve GoBackN

- Recieves gives individual ack for each packet.
- So on a timeout, sender only sends that one packet.
- Once reciever reorders packets once it gets all N, then the sender can start sending the next packets

(Note: Selective Repeat Dilemma) (Aldus)

↳ Keep sequence number size $\geq 2 \times$ Window size to solve this dilemma.

---

*Estimating Timeout:

- Timeout $>$ RTT , otherwise we will always get a timeout.

⇒ For successful acks, calculate ERTT

$$\text{EstimatedRTT} = (1-\alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

(Generally ~~$\alpha \approx 0.125$~~) $\alpha \approx 0.125$

SampleRTT ↑ Current RTT for a successful ack on a packet. (Ack. Time − Send. Time)

- We also use DeviationRTT

$$\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta(\text{SampleRTT} - \text{EstimatedRTT})$$

$(\beta \approx 0.25)$

⇒ Timeout = EstimatedRTT + DevRTT*4

---

*Note: In TCP, each byte corresponds to a sequence number.

---

*TCP Fast Retransmit:

- Even if we don't get a timeout, but get multiple duplicates of acks. Sender understands that the receiver missed a packet after number, so sender sends it again.

---

*TCP Flow Control:

- Sender limits amount of unacked data based on recievers buffer space.

⇒ rwnd value indicates buffer space which is sent in message headers from reciever to sender.

—— x ——

*Handshaking (Establishing a connection)

- There are problems with 2 way handshake

⇒ So we use 3 way handshake. (Initiate, OK, Ok) (TCP)

—— x ——

Note: Closing a connection in TCP is also a similar way to 3 way handshake.

—— x ——

*Congestion Control:

- Retransmissions, Timeouts can cause congestion.

1) End-End control: (TCP)

- Source & Destination using acks & timeouts do the retransmissions as needed.

2) Network Assisted control:

- A lot of help/assistance is provided by the routers etc along the network

—— x ——

: Fast Recovery in TCP ←

* TCP Congestion Control:

* Additive increase, multiplicative decrease.
* Sender increases window size by 1 (for every RTT) until loss occurs and then reduces size by half. (Sawtooth behaviour)

———x———

* **Note:** Slow start ensures increase is exponential initially but then the increase is linear. (window size).

———x———

* TCP Tahoe (old)

- Decrease of window size to 1 if loss is detected
- Uses slow start behaviour.

TCP Reno (Popular)

- Decrease of window size to half of its current if loss detected.
- Uses slow start behaviour, but this will go with a linear increase only ↙

———x———

Note ⇒ Slow start uses
* a threshold (ssthresh) to decide when to go to a linear increase
→ On loss, $ssthresh = \frac{1}{2} cwr\_window\_size$

———x———

** **Note:** TCP Throughput $= \boxed{\frac{3}{4} \times \frac{W}{RTT}}$ → Window size

(W_{cwr} varies from $\frac{W}{2}$ to W)

$\therefore avg = \frac{3}{4} W$

———x———

——END of MID 2——

*Network Layer:

- Transport Layer segments are packed into datagrams
- Routing is the entire trip from source to dest.

* ⇒ Forwarding is how the packet moves within a router (input to output of router)

*Forwarding Table: (Virtual Circuit)

- At each router, it determines which output link of the router the packet must take. (Using the headers of the ~~datagram~~ → packet (#VC.No etc)).

*Datagram vs Virtual Circuit:

- Datagram (Latest) ⟶ Connectionless service

  Virtual Circuit (Old) ⟶ Connection based service (Route predetermined)

*Datagram Forwarding Table:

- A range of addresses (IP's) are entries of the table which give us an output link corresponding to each range. (These IP's are destination IPs

—— x ——

Note: When we search a datagram forwarding table, we find the range which is the longest prefix match of our required dest IP and we use it.

———x———

* * Switching (Packets) (From input to output)
(Hardware of router)

1) BUS → Fast, rate limited by bus bandwidth → generally not a problem.

2) Pipeline → Rate limited by memory transfer speed

? 3) ~~[illegible]~~

———x———

* Buffering:

- If transmission is slower than arrival rate, then the buffer is filled.
- Scheduling policies are used to determine which packet to schedule first.

———x———

* Note: Recommended buffer capacity

$$= \frac{RTT \times C}{\sqrt{N}}$$

C → Capacity of link?

N → No. of parallel links

———x———

* Internet Protocol (IP)

- **IP Fragmentation:**

⇒ On links of low capacity packets are split into smaller packets & headers are added.

——x——

Note: IP Address → 32-bit identifier for host, router interface.

——x——

* Subnet:

*

- Higher ~~part~~ order bits of the IP address.

⇒ Lower order bits are the ones corresponding to the host.

- Devices within a subnet can reach each other without ~~inter~~ the routers help.

——x——

* Classless InterDomain Routing: (CIDR)

⇒ IP = a.b.c.d/x , x = No of bits in subnet part of the IP.

32 − x = No of bits in host part of the IP.

——x——

Note: **DHCP**: Gives a host a dynamically assigned IP, IP of DNS server, first hop IP for client etc.

** (Handshaking)

⇒ Commands → DHCP discover, offer, request, ack.

——x——

Note: In DHCP, broadcasting of IP requests/responses are done since the source has no specific IP yet, so broadcast just goes to all possible connected machines on the network and our client picks up the IP from the broadcast.

Note: If two clients are requesting for DHCP simultaniously, then they might get the same IP & we get an IP conflict. So the clients would request IP's again.

Note: ICANN gives IP ranges to ISP's
↳ Assigns domain names etc.

Network Address Translation (NAT)

(used to solve less no. of IP's problem)

- All IP's from within a network are sent out as a single IP.

⇒ Provides security for machines within a network

- NAT routers (LAN) stores a mapping of, (WAN)

(Source IP, Port) ⟶ (NAT IP, New Port)

& uses this mapping to send & recieve data from the internet. (Local IP's can be whatever we want & NAT can just store the mapping)

⇒ Provides problems for P2P etc, since this NAT always comes inbetween.

⇒ Also issues that its no longer an end-end connection etc.

Note: NAT & IPV6 are 2 methods which can help us by giving us a much longer range of IP's. NAT is less preffered, due to the problems it causes.

Note: On using P2P with NAT, both the end establish a connection with a relay so their connection is made.

—x—

Internet Control Message Protocol: (ICMP)

- Used to send error messages.
- Discover & fix errors at various levels. (Slides).

(Used along with traceroute at times)

—x—

*IPv6:

- 40 bit headers used instead of 32.
- No fragmentation of packets are allowed.

*⟹ Tunneling:

- Among IPv4 routers, IPv6 datagram is carried as the payload in IPv4 datagrams.
- This is useful if some routers do not support IPv6.

—x—

*Routing Algorithms:

→ Link State Algorithms → Centralized cost matrix. (We know all the paths of the network & their costs)
(Dijkstra's Algorithm used)

→ Distance Vector Algorithms → Each router only knows some info about adjacent paths etc.
(Bellman-Ford DP)

⇒ Static → Routes change slowly over time.
Dynamic → Routes costs etc can change quickly etc.

Note: In distance vector algorithms:
→ changes/updates of a nodes estimate will send a signal to all its neighbours who propogate it to their neighbours if their value changes.

---

* Link Cost Increase Problems: (Count to ∞ problem)

- When the link cost increases, then the paths from a neighbour to the goal which go through us again, then value changes are occuring to and fro, takes a long time to stabilize.

* 1) Poisoned Reverse: (A partial solution to the count to ∞ problem)

- When a link cost changes, signals to neighbours are sent saying the link b/w node & neighbour = ∞, so they dont come back to this node.

(This might ignore shortest paths at times)

---

* Hierarchial Routing:

- Above algorithms are not applicable in actual scenarios.

⇒ Aggregate routers into AS (Autonomous systems)
↳ They run "Intra AS" routing protocol.

- Gateway router → At edge of an AS & another AS.

Note: Inter-AS routing at the edge connecting routers of AS must have info on which routers can be reached by a certain AS.

⇒ For Intra AS routing Dijkstra or Bellmann ford is used. (similar)

—x—

Note: Hot Potato Routing:

- Select shortest route to destination using intra AS routing to get distance to edge nodes & then inter AS for dist to destination.

—x—

Intra AS Routing:

1) RIP: (Routing Information Protocol)

- Distance vector algorithm, but limits hops to dest ≤ 15, else marked as ∞.
- On updates of hops etc, same sort of advertisements are done.

⇒ These 15 hops are b/w AS's.

- Poison reverse is also used here so back & forth problems dont occur.

* 2) OSPF: (Open Shortest Path First)

- Uses Dijkstra's algorithm.

⇒ Similar to RIP, but messages are authenticated.

- Multiple same-cost paths can be stored.

⇒ For each link, depending on the service cost is given (pay for service).

* Hierarchial OSPF:

- In each AS, a backbone with backbone routers are present which are connected to a boundary router.
- These backbone routers have a lot of 'inter routing' info.

⇒ Within backbone OSPF is used to route packets.

---

* Inter AS Routing:

(Slides)

1) BGP: (

- Within AS, iBGP advertising happens
- Between AS, eBGP advertising happens.

⇒ Restrictions etc to stop some inter AS links can be done in BGP. (policy)

- Uses prefixes to determine which AS the packets with a certain prefix need to be sent to etc.

---

* Data Link Layer:

⇒ Can perform flow control b/w 2 adjacent nodes (routers).

- Error Detection & correction (if possible)

* ⇒ Half Duplex & Full Duplex
  - Half Duplex → Single way communication
  - Full Duplex → 2-Way communication

- This layer is implemented in our N.I.C. (Network Interface Card).

——x——

* Error Detection:

- Use some EDC bits so that they can be used to correct packets.
- Parity checks. (~~Even~~ odd or Even no of 1's)
  - Single bit parity → Can only detect errors
  - 2D data, row & column parity → Can detect & correct single bit errors.

⇒ Checksums can be used at transport layers.

* ⇒ Cyclic Redundancy Check: (Slides)

- Use some $G_r = r+1$ bits & we ensure our data is divisible by G_r by appending redundant bits to our data.

——x——

Note: An ideal multiple access protocol,
→ when M nodes are sharing each should get N/M bandwidth (N = Total bandwidth)

———— x ————

MAC Protocols: (Medium Access Control)

1) Channel Partitioning: (esc: TDMA) (FDMA) → Effective at high load
→ Time slots & frequency based

*2) Random Access Protocols: → Effective at low load.
* ⇒ They directly send packets
⇒ On collision, after a small amount of time, we retransmit.

esc: 1) Slotted ALOHA:
- Time divided into slots (frames). If 2 or more nodes transmit, then they all detect a collision.
- After collision, they will try send with a probability p in next frame.
- On a certain p_x & $n \to \infty$, max utilization = 37%

2) Pure ALOHA:
- No time division into slots, so this is worse, max utilization = 18%

*3) CSMA (carrier sense multiple access)
- If channel is idle then send, else wait.
- Collisions might occur if packets are sent at the same time (Due to propogation delay).
- CSMA/CD, on collision they stop transmission & try again later.

Ethernet (NIC) uses this.

$$\frac{1}{1 + 5t_{prop}/t_{trans}} = \text{Efficiency}$$

*3) Taking Turns Protocol:

a) Polling: (Very inefficient)

- A master & slaves, the slaves respond to master's poll if they want to send data.

*b) Token Passing:

- Control Token passed from one node to another in a circle.
- If a node has the token, it can send data.
- Single point of failure → If something happens to the token

---

* MAC Address & ARP: (Address Resolution Protocol)

- 48-bit MAC Address → In our NIC.
- IEEE stores available MAC addresses for manufacturers of NIC's.

** ⇒ ARP does MAC address to IP of destination (Similar to how DNS functions).

- ARP table stores mapping for IP to MAC address. It also stores a TTL, time to live (generally 20 mins) after which it gets invalidated.

- ARP broadcasts queries & corresponding IP single cast's its MAC address to the ARP source & it will update its ARP table.

⇒ When sending the packet, another wrapper is added which has the source/dest MAC addresses so that destination knows if packet is for itself. (So intermediate routers won't read the packet).

———x———

Note: Ethernet:

- Its unreliable & connectionless.
- Unslotted CSMA/CD with binary backoff → the p is binary increase not exponential.

———x———

*Switches: (Switch Table)

- They learn through locations of senders sending packets to the switch.
- When a packet comes to the switch, the entry of source host (MAC addr) & interface on switch are stored in a table.
  - ↳ So if other packets are being sent to this source from other hosts, they can be routed easily.

* ⇒ If no entry is present initially in the table for the destination MAC addr, the switch floods the packet through all its interfaces.

———x———

* VLAN: (Virtual LAN)
(slides for pictures)

- Switches support multiple LAN's.

⇒ A single switch can be used to connect to any available VLAN.

→ traffic isolation → Each VLAN data is contained within ports connected to that VLAN.

- Trunk port → Can connect switches to each other.

———— x ————

* MPLS: (Multiprotocol label switching)

- IP address → part of it identifies the network & other part identifies the host.

⇒ MPLS → Unique label given for each host, so its faster than using IP to find the destination.

- MPLS enabled routers add an MPLS header to the packet.

↳ Supports fast reroutes
→ Precomputed backup paths unlike normal routing algos.

⇒ MPLS allows differential services (reserved bandwidth)

———— x ————

Note: Data Center networks

———x———

*Wireless & Mobile Networks:

- **Base Station:**
  - Sends packets from wireless to wired, and vice versa.

⇒ Networks with a structure has (and uses) base stations.

*Ad-Hoc Networks:

- Host can transmit to others within range.
- No base stations etc.
- No connection to the wider "internet"

———x———

*Wireless Links:

- Decreased signal strength, signal attenuation.
- Interference from other sources.
- Multiple senders & receivers cause more problems.

→ Hidden Terminal problem (Slides)

↳ A sees B
C sees B
A does not see C

———x———

*Code Division Multiple Access (CDMA)

- Use a unique header on messages corresponding to host.
- All hosts can talk to each other at the same time, and they just use the header for identification.

———x———

Note: 802.11 Channels & association

** → Follows CSMA/CA (slides)

→ Collision avoidance (RTS → request to send)

→ If no collision, then send data

→ Then send ACK so others can send RTS

———x———

* Note: Cellular Networks & 4G → All IP (Only 1 protocol from start to end)

———x———