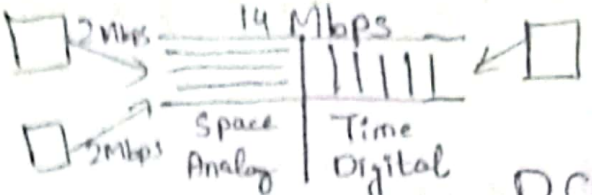
→ Data Rate Limits, (P×A) = or xxy no. of variations = Total.
→ Constillation Diagram.

DCCN:-

→ Models are used in every field of science.
→ We use different models to understand networks.
→ OSI

OSI
Application
Presentation
Session
Transport
Network
DataLine
Physical

→ Two approaches to study models:-
1) Top-Down approach.
2) Bottom-Top approach.
→ Email → IRAZA@cuilahore.edu.pk
→ Physical layer transfers data (in form of binary) from one computer/machine to the other in form of signals (i.e. Analog (~~~ waves) or Digital (⊓⊓⊓⊓)).
→ Book ⇒ Top-Down approach (but method 2 is also used)
→ Computer Networks featuring Top-Down approach.
→ Data is also of two types (i.e. Analog or Digital).
→ Waves (Analog signals) have some fundamental characteristics e.g. frequency, amplitude, phase etc.
→ By changing these characteristics (cycle/intervals), we can transfer signals (analog/digital) to other machines
→ We use a scheme to represent 1's and 0's (e.g. FSK scheme (Frequency shift key), represents 1 where there is more (larger) frequency and represents 0 where frequency is smaller
→ OOK = On Off Key (scheme represents 1 if on, 0 if off).
↳ If there is no amplitude then 0, if there is amplitude then 1.
→ ASK = Amplitude Shift Key (change amplitude with keeping other 2 constants and represent 1's and 0's).
→ PSK = Phase Shift Key (changing Phase represents 1's and 0's while keeping other 2 characteristics constant).
→ Bandwidth = is a property of signal and medium.
→ Difference between highest and lowest signal
→ Range of signals (Hertz) that a medium allows to pass.
→ Data Rate = no. of bits that are transferred in a cycle.
→ Formula to calculate rate = $\log_2(L)$ ⇒ L = variations.
→ L has to be multiple of 2 (in example ⇒ 2L = 5 or 10)
→ Changing characteristics too much can attract noise (something unwanted).
→ Hence we combine and change 2 properties (e.g. ASK+PSK
→ QAM (Quadrature Amplitude Modulation) is very widely used example).



→ 2 types of technologies can be used in data link
↳ Non-deterministic or Deterministic (You know who is submitting (You don't, who is transmitting).
→ Ethernet, WiFi are Non-det (chance of collisions)
→ Token Ring, FDDI are Deterministic.

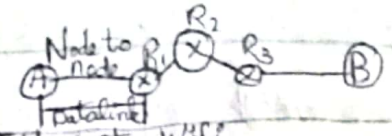
DCCN :-
→ Every Technology will have diff header fields.
→ WiFi has 4, Ethernet = 2.

→ Analog signal ∿ is very weak as interference / disturbance can affect it.
→ Composite signal ⊓⊓ (composed of diff signals).
→ Bandwidth can be calculated by highest signal - Lowest (eg 7 - 1 = 6 Hz).
→ A digital signal is also a composite signal (as if we keep on adding signals, we will end up with a perfect square wave).
→ A digital signal has an unlimited bandwidth.
→ Signals are represented as a function of time (in time domain).
→ ⊓⊓ this is time domain representation (x-axis = time).
→ Another representation of a signal can be in frequency - domain (x-axis = frequency) ⇒ e.g. ⊥ 3 5 7 (f).
→ We need to convert signal from (time domain to frequency).
↳ Because all details are in frequency domain, calculations and operations are much easier to perform. (For example time domain will give a diff. eq while freq domain will give simple algebraic eq).
→ This conversion process is known as Fourier transformation.
→ Inverse fourier transformation = to convert back from freq to time.
→ Any signal shape can be represented as sum of series of sin & cos waves.
↳ This is called Fourier series (many sine wave and cosine waves).
→ Digital encoding scheme for a digital signal +5, 0V | 1 | 0 | 1 |.
→ Another encoding scheme is Manchester (Ethernet), NRZ, NRZ-I, NRZ-L.
→ We evaluate all these schemes on 3 parameters ⇒ 1) Self-synchronization, 2) DC-component, 3) Baseline wandering.
→ We send redundant bits while using ethernet, with signals so that receiver can align its clock with sender.
→ Receiver can only adjust / align clock when change in signal occurs.
→ When clocks of receiver and sender are not synchronized, data corrupts.
→ Manchester ⇒ there will always be a change a signal in the middle / interval of clock (either from +ve to -ve or -ve to +ve).
↳ Change in signal will be just in middle or in start and middle both.
→ A good network is one that is being used at full capacity (rest of time).
→ To maximize network utilization, share network with other machines.
→ We can share the network in terms of space or in terms of time.
→ Whenever we have an analog singal, then share by space.
→ Whenever we have digital signal, then share by time.

DCCN Theory :-

→ FDM = Frequency Division Multiplexing, WDM = Wavelength.
→ Advanced Form ⇒ OFDM ⇒ Orthogonal Frequency.
→ These two are used with analog signals (space).
→ For time (digital signals). ⇒ TDM = Time Division Multi.
STDM =
→ Data link layer ⇒ combines the bits to form a structure (frame).
→ logically grouping the incoming bits. (Header fielded and main load).
→ Data link responsibility is to provide node to node connectivity.
→ This connection can be either wired or wireless. (many further options)
→ Data link has header = many other feilds + payload data. Imp 2 most Source MAC Address (Media Access Controller) and Destination MAC address.
→ Forwarding decision requires heirarchical information e.g. postal address.

DCCN :-



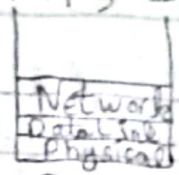
→ Collisions ⇒ a problem occurs b/w access of the machine.
  → This problem can be resolved by = Media Access Control Mechanisms.
→ CSMA = Carrier Sense Multiple Access (Common sense for machines)
  → Machine should wait until one signal is finished transmitting. Then transfer the other signal when machine is free.
→ But there can still be a collision if multiple people access shared machine at the same time when it is free.
→ So we use CD = (Collision Detector) with CSMA.
→ We compare transmitted signal with received signal and check their strengths (energy levels are same).
→ Ethernet uses CSMA/CD to counter/resolve collisions
→ The signals transmitted through wired network can remain same till 100m
→ But energy level/strength will be higher at the transmitter and strength will be low on reaching receiver.
→ So we then avoid through = CSMA with CA (collision avoidance)
→ Wifi = (Wireless network) uses CSMA/CA for collisions.
→ Deterministic (Token ring) use token passing as access mechanism.
→ Error Detection and Correction Mechanisms → (Check if data is same as it was transmitted as original.
→ Machine can detect errors using some mechanisms such as parity checking or CRC any other. For example,
→

Data	EDC
101 ⊻ 10 ⊻ 100	1001

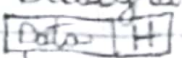
= Error Detection & Correction Bits. (Detect error by comparing EDC bits).
→ Ethernet, Wifi use CRC = Cyclic Redundancy Check (very accurate).
→ Checksum = we use 1's/2's complement/binary add/sub to calculate EDC bits.
→ Options for corrections ⇒ Retransmission (ask the user/sender), or use Forward Error Correction Mechanism (correct error itself).
→ The challenge in correcting error itself is to find position of bits that are incorrect data (then just flip them).
→ All of these technologies are LAN (some are wired or wireless).
  → Ethernet, Token Ring, FDDI are all LAN tech.
  → Which means Local Area Network technologies.

→ Types of IP ⇒ IPv_4, IPv_6

→ Network Layer



→ Datagram / Packet



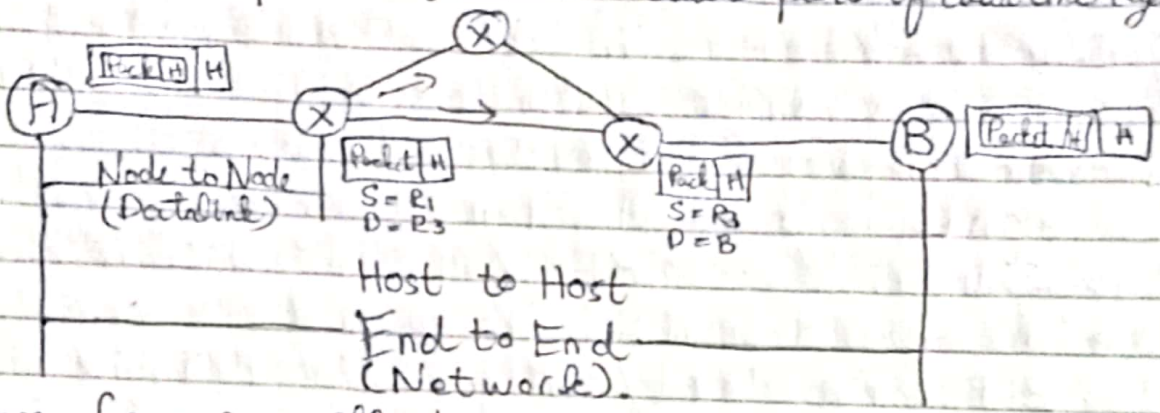
= (Data Part + Header)

↳ Segment / structure of Data.

→ PDU = Protocol Data Units. (Data Chunck / Fragments).

→ 2 most important fields in header (are addresses).
- → Source IP Address
- → Destination IP Address

→ As we move from Network layer to Data Link layer, a packet is encapsulated inside Data part of data link layer. (frame)



→ A frame from A will have Source MAC address of A machine and Destination MAC address of R_1.

→ The packet from A will have Source IP address of A machine and Destination IP address of B machine.

→ Nodes change on the way but packets do not.

→ Sorting algorithm decides whether R_1 will send data to R_2 or R_3.

→ IP → (Internet Protocol) (International network)

↳ Inter means b/w two diff network communicating.

↳ Mainly two types IPV4, and IPV6 (study both H).

→ Network has IP packet or datagram (then IP addressing).

→ [192, 168, ↓, ↓] → Example of IPV4 address (common)

↳ 32 bit ⇒ 2 types = classfull add, classless Add. (Class)

→ IP address has 2 parts i.e. [Network] [Host].

→ Class A = 10.0.0.0 = [N] [H] [H] [H] (32 bits)

→ Class B = 172.16 = [N] [N] [H] [H] (4 Octets)

→ Class C = 192.168.1.1 = [N] [N] [N] [H]

→ Class A = 8 bits for network part, 24 bits for host part.

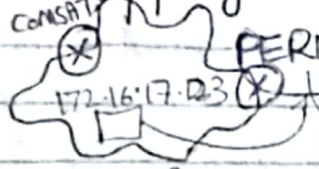
→ Class B = 16 bits for network part, 16 bits for host.

→ Class C = 24 bits for network part, 8 bits for host.

⇒ These classes are for unicasting (one unit).

→ Subnets are identified by subnet masks. (use for easier identification).
→ Scenario ⇒ → For same = no need of router but for diff network we need to use router for sending.

→ Class D for multi-casting (more than 1 machines).
→ Class E for testing purposes (total no. of classes = 5).
→ Ranges of numbers tell about the class of representation.
→ Range of first/Left most octet for class A = (0-126).
→ If we have an IP address of class A then we can assign diff IP addresses to at most 2^{24} machines.
↳ So class A IP address is most expensive to buy (rare).
→ Range of first/Leftmost octet for class B = 128-191.
→ 127 is not present in range because it is reserved for local host (i.e. 127.0.0.1 = local host) (both are same).
→ For B class IP address, we can assign 2^{16} machines.
↳ $2^{16} - 2$ (one for network address and one for broadcasting).
→ $2^{\text{(number of bits in the host portion)}} - 2$ to assign machines.
→ If all host bits are set to 0, then we get network add.
→ In an octet $\boxed{2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0}$, value doubles from right to left.
→ If all host bits are set to 1, then we will have broadcasting.
↳ Used to send a message to all machines connected with network address.
→ Class C = 192-224, Class D = 225-236.
→ For C class IP address, we can assign $2^8 - 2 = 254$.
→ A class = very large, B class = big, C class = small.
→ Class B is most desired/wanted/used due to size.
→ Some IP addresses are for free series access (not addressable).
↳ They are not unique (cannot be accessed from outside).
→ Free series for A = 10.0.0.0, B = 172.16.17.23, C = 192.168.1.0
→ We can't sent messages from free series IP addresses, so we do mapping with other network addresses and then send.

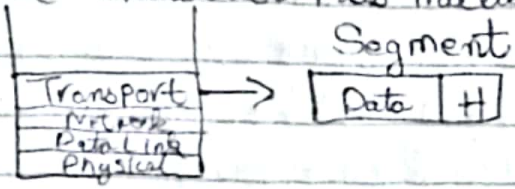


Subnetting is a process for creating subnets. (end result is subnets).

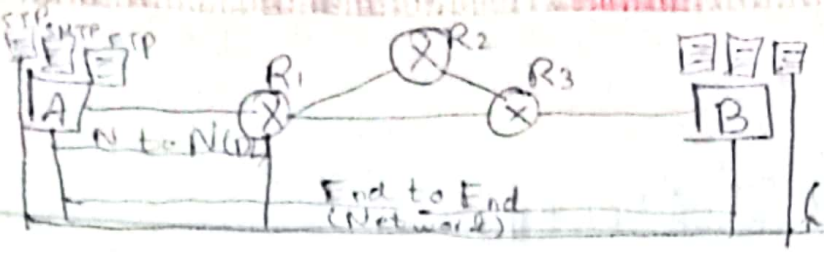
→ Classless addressing = 192.168.1.1/24 (no. of bits for network)
→ Subnet mask = 32 bit address (larger network into smaller).
↳ We turn all network bits to 1 and all host bits to 0.
→ Subnet mask for class A = 255.0.0.0 (default), B = 255.255.0.0, Class C 255.255.255.0
↳ Default arrangement (don't need any subnetting).
→ We have to give subnet mask to machine or default will be used.

DCCN :-

→ Network Layer ⇒ IPv4 = 32 bits, IPv6 = 128 bits.
→ Both IP versions are written in hexadecimal form.
→ IPv4 = (192.168.1.1), IPv6 = (2001:0db8:001:0000:0000:00b:2ABC:1200)
→ Each grap (2001) is called hextet, contains 4 hex digits.
→ 1 hex digit = 4 bits, 1 hextet = 4 hex digits = 16 bits.
→ Total hextets in a valid address = 8 (16 x 8 = 128 bits)
→ The zeroes can be omitted ⇒ 2001:0db8:1:(All zero):2ABC:1200
→ Subnetting ⇒ divide a larger network into smaller networks to improve effeciency and management.
→ Subnetting is a process → results in subnets → identified by subnet masks.
→ 2 types of subnetting (IPv4) ⇒ FLSM = (Fixed Length subnet masking), VLSM = (Variable Length Subnet Masking).
→ FLSM will have only a fixed no. of machines (needed or not)
→ NAT = Network Address Translation = (Mapping, replacing).
→ Segmentation and Reassembly.
→ Frame for ethernet can have a maximum size of 1500 bytes.
→ Frame for Wifi can have a maximum size of 2312 bytes.
  ↳ This is called MTU (Maximum Transmission Unit) size.
→ Ethernet cannot have less than 64 bytes in its frame.
→ Fragmentation (if size is greater than MTU than divide).
→ Reassemble at the receiver end to combine fragments.
→ Some parts in header part help in fragmentation + reassembly.
→ Routing Protocols = contains routing algorithms for forwarding decisions.
  ↳ 2 Types ⇒ Unicast (only one receiver) ⇒ RIP, OSPF and Multicast (many/multiple receivers) ⇒ MOSPF, PIM.
→ Scalability ⇒ after expanding system/users, performance remains same.
→ RIP, OSPF are not scalable, only used within network.
→ For routing over the Internet, we use BGP (Border Gateway Protocol) (as internet has millions of routers) rows in table.
→ Transport Layer



→ As we move from transport layer to network layer, the segment is encapsulated inside a datagram or packet.



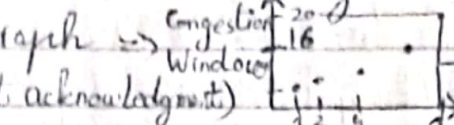
→ Every process has a unique address called port number.
→ They are also standardized → (80 = HTTP (browser), 25 = SMTP (emails), (20, 21 = FTP (File transfer)) (Range 0–1024).
→ Port numbers = 16 bits = 2^{16} port (PN) numbers (1024 standard).
→ Complete address info = (182.12.5.14 (IP), 80 (PN)) = Socket.
→ Socket means logical connection, PN is also logical.
→ Head = Source Process Address, and Destination Process Address.
→ Simplest representation of socket = (IP, Port number).
→ Two imp protocols at transport layer = TCP, UDP.
→ Any segment will be either TCP = Transmission Control Protocol or UDP = User Datagram Protocol.
→ TCP = connection oriented (Acknowledgement), reliable, UDP = not connection oriented (connection-less), non-reliable, simple socket type.
→ But UDP is faster because acknowledgments is not required.
→ TCP is more used (frequent) than UDP (most of apps).
→ 2 types of applications (reliable, no-loss), (non-reliable, no-delay).
→ Most of apps use TCP because of reliability (used more).
→ TCP → 1) Connection Management Mechanisms. (3 way hand shaking process).


2) Congestion Control Mechanisms.
3) Flow Control Mechanisms. (TCP is byte oriented stream).
→ 1st. Request to server, acknowledgement, then data send.
→ Connection cookies → hash number of 5 nos. → including a special number → used for SYN Flooding
→ Congestion = more traffic on the network (more) (more users).
→ Two types of congestion controls → End to End, Network-Assisted.
→ If there is no acknowledgement within timer → then machine resends S1.
↳ This is called Automatic Repeat Request (ARQ) (slower) → [Timer is very important and is calculated by Round Trip Time (RTT)] → (from A to B then B to A).
→ This timer is known as RTO (Retransmission Time Out) (very important).
→ If timer is longer (then network underutilized (inefficient)), if it is shorter (then more congestion).
↳ To calculate timer → Exponentially Weighted Moving Average (EWMA algo).
→ Triple Duplicate Acknowledgement (FAST Retransmission) → (3 DUP ACK).
→ First byte of data field is the sequence number in TCP.
→ When server sends 3 duplicate acknowledgement to sender, it retransmits that segment which was lost / not received.

→ How Real time apps handle delay (streaming) (Diff Serv)
→ Two types of services → 1) Integrated Services (IntServ), 2) Differentiated Services.
→ Next 3 layers (Session, Presentation, Application) will be studied collectively.
→ Down 4 layers deal with (Hence Imp) transportation of data

DCCN :-

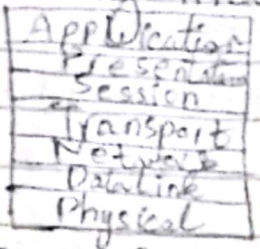
→ Commulative segments are send and Commulative Ack received.
→ 2 mechanisms for reliability of TCP → ARQ, 3 Dup Ack.
→ Variants of TCP → TCP Tahoe (3 Dup), TCP Reno, TCP New Reno, TCP SACK.
→ We need to slow down the sending rate to solve congestion.
→ Internet congestion graph → Congestion Window (Segments that can be send without acknowledgment)



→ (After reaching threshold, we don't double, we only add 1).
→ (After experiencing a congestion, we go back to slow start (Sop), additive increase (adding 1 after threshold), multiplicative (divide) decrease).
↳ Why would we go back to slow start after multi decrease.
↳ Hence New Reno TCP says to starts from new threshold.
→ Reno can't handle multiple segment losses hence New Reno TCP is used with FAST Delivery to handle.
→ We use a sliding window for sending multiple segments.
→ Algorithm - AIMD - Additive Increase Multiplicative Decrease
↳ AIAD - reaction to congestion is not justified (congestion again).
→ Sliding Window Syndrome → Window size large or too small.
→ Flow Control → if data is send at a speed more than the network capacity then data can get corrupt / lost.
→ when buffer gets full at server end then packets are dropped.
→ If application is slow to read from buffer than mismatch occurs b/w sending rate and receiving / reading rate.
→ Flow control is a matching technique / service where sender has to match its rate with drain rate of receiver.
→ If there is a mismatch, then receiver may drop packets.
→ There are 3 Flow Control Mechanisms (Generic) -
1) Stop and Wait ARQ → (Slow) Send a segment, then wait for acknowledge).
2) Go Back N ARQ → (Fast) sends n segments, commulative ack, if lost, it resends all segments from lost).
3) Selective Repeat ARQ → (sends n, come ack, only resends lost segments).
↳ It will go back to the lost segment and then resends all subsequent segments (keep track of start, end, current of window).
↳ Used in SACK TCP, sends NACK to sender (negative acknowledge).
→ Quality of Services (QoS) → can be different definations of quality.
↳ Its ability of network to handle different requirements / services.
↳ Queuing Mechanisms → to handle diff. services differently (not FCFS).

DCCN:-

→ We calculate and use drop probability to drop or retain packets of data (Droptail is used).
↳ Queing mechanisms are used to prevent buffer overloading.



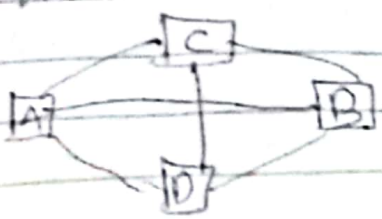
⇒

⇒ Web (http), Email (SMTP), File Transfer, Proxy (FTP, DNS) servers

→ The architecture of all these apps is based on client server architecture (software design means implementation of classes and their relationships, good design = highly cohesive and loosely coupled). (So changes can be easily made). (class should not be performing diverse functions (high cohesion)).
→ Architecture of app means placement of programs.
→ More files on server → server client architecture.
→ Thin or fat type architectures based on no. of files placed on either end ie. server or client.
→ Peer to Peer (P2P - Skype, BitTorrent) (are scalable).
→ Internet we use = Web 2.0 (client server architect).
→ New technologies → (Web 3.0 → decentralized).
→ Open System Interaction (OSI model) by ISO. (old 1980).
→ TCP/IP model ⇒
↳ (we will study a hybrid model of both)
→ OSI is a reference model.

Application	→ All 3 combined.
Transport	→ Similar
Internet	→ Network (Similar)
Network access	→ Phy + Data Link

→ OSI supports vertical development (made after TCP/IP).
→ Horizontal → add components in same layer.
→ Vertical → create new layers / new verticals (diff resources).
→ TCP/IP is more secure than OSI model.
→ TCP/IP - end machines are responsible for QoS.
→ OSI - core network is responsible for reliability, QoS.
↳ Hence it is more robust (diff to make changes).



→ ⇒ Can be implemented but not a good approach as (n-1) wires with 1 machine.

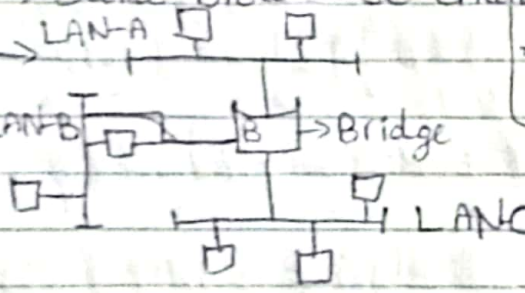
→ HUB / repeater - are physical layer devices.
→ Switch + Bridge are data link layer devices.

10 Mbps Ethernet (No scalability)

→ Contains shared medium (wire) hence collision can occur (more machines).
→ Hence, we should be using network devices.

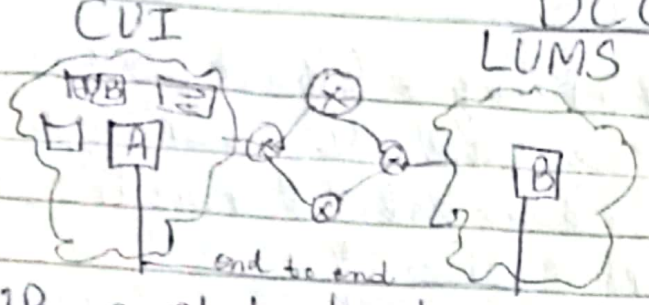
→ HUB = center (dumb, not intelligent).
↳ It only copy/pastes the data.
↳ it copies from incoming port and pastes on outgoing port.
→ Passive hub = copy/paste, active hub = regenerates data.
→ Cat 5 cable = good transfer of data till 100m, after that distanation starts (as signal starts to loose its shape).
→ Repeater (device) regenerates the data signal (more 100m).

→ Switch is an intelligent device as it makes decision to send data.
→ Switch uses MAC address (in MAC table form (one)).
↳ It determines port number which should receive data by using header of MAC address of sender of data.
→ Switch segments the collision domain (one machine = one domain)
→ Internet = broadcasting in nature, switch = switched ethernet.
→ Advantage of switched ethernet = intelligence (decision).
↳ Same broadcast channel (one switch = one port/machine).



→ Bridge is also intelligent. It is used to connect different LAN segments (one or more devices).
It filters and forwarding.
It also uses MAC addresses.
→ Compares source and destination MAC addresses.
→ It also maintains a MAC table for data transfer.
→ Switch connects 2 individual entities (machines/others).
→ Bridge connects 2/or more LAN segments.
→ Switch uses Cut through switching (as it reads the destination MAC address, it starts sending data).
→ Bridge does store and forwarding (waits for a full segment data to arrive and then sends).
↳ Has 35% Latency i.e. 35% slower than switch.
→ Switch has micro segmentation = a switch has multiple paths to send data (many busses).

DCCN:-

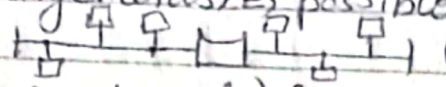
→ 

→ Bridge/switch cannot be used in this (end to end) as we need to operate on IP addresses (router).

→ HUB, switch, bridge are intra networking devices (within)

→ Routers are internetworking device (b/w networks), but routers can also be used as intra networking to segment broadcast domain.

→ Physical Topologies (Arrangements) ⇒ possible ways to arrange devices.

→ Bus arrangement ⇒  (It carries data of all devices of network) (not scalable, collision). (no control)

→ Ring arrangement ⇒ once data is transmitted then no control, need to travel more path to send. 

→ Mesh arrangement ⇒ connect every device with every other device (fast but if more machines then not good).

↳  (No network device can be used, no control over data once it is on wire).

→ Star arrangement ⇒  ⇒ it is scalable, switch can control the data, but what if switch crashes.

↳ It has single point of failure hence crash chances.

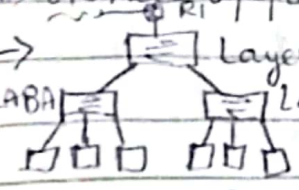
↳ But we can control/manage services + data transmission.

↳ Very widely used in common places (extended version).

→ Extended star arrangement ⇒ it has more switches (network devices)  (each itself is a star).

→ Upper layers are superior in functionality than lower ones.

↳ Hence upper layer devices are superior than lower layer.

→ 

⇒ Heirarchical arrangement ⇒ diff. layer of devices used, combine star, extended star and heirarchical.

→ Logical Topology = Logical arrangement of devices.

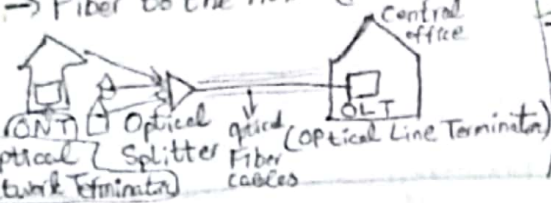
→ 2 types ⇒ Broadcast in nature = e.g. Ethernet.

→ 

⇒ If we use HUB every device gets data (i.e broadcasts to everyone), (broadcast nature).

→ So we use switch to limit breadcasting of ethernet, switch makes it switched ethernet (only dest gets data).

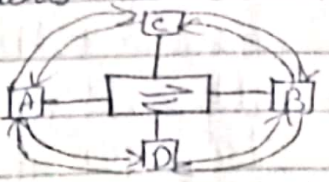
→ Fiber to the home (FTTH)



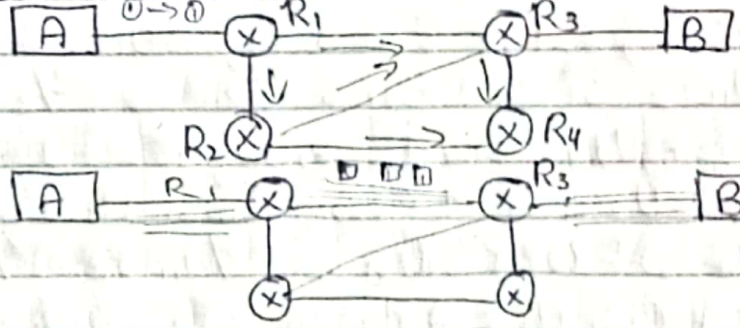
→ Optical networks = Active, Passive optical network
↳ widely used
Two optical distribution network architectures (AON, PON).
Switched ethernet

→ Logically the network forms a ring and data is transferred via a switch but only within a ring.
→ Token ring ⇒ we generate a token which keeps moving over network, if a machine wants to send data it has to seize data (otherwise it keeps moving). Hence only one machine can send data at a time.

→ FDDI ⇒ (100 Mbps)

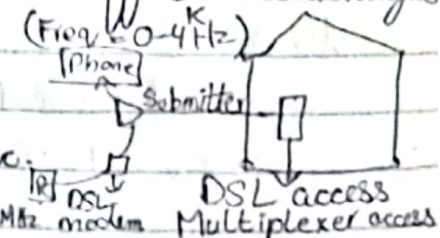


→ FDDI has a secondary ring which can be used if we want to keep our network running all the time.
→ Fiber Distributed Data Interface = FDDI.
→ Network protocols = which help communicate from 1 machine to other.
→ Packet switched networks, circuit switched networks.
→ Packet Switched Networks

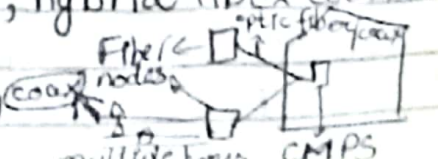


→ Circuit (Virtual) Switched Networks

→ Packet SN offers better sharing of transmission ⇒ it is more effecient and less costly + it is simpler.
→ Switching from circuit switching to packet switching.
→ LAN ⇒ home access is very important ⇒ we have different technologies
DSL, Cable, FTTH, Dial-up and satellite
↳ Digital Subscriber Line
→ Asymmytric = diff. upstream + downstream traffic.
→ The amount of speed to send data to internet = 50K Hz to 1 MHz for downstream
↳ is called upstream.

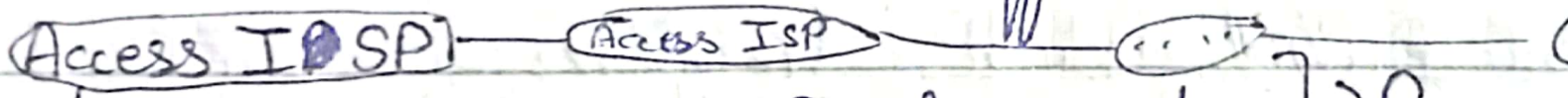


→ Not good for more than 5 to 10 miles, so use other/alternate technology
→ We also have some traditional telephone network swithing
→ Cable internet access = coaxial cable, fiber optic, hybrid fibex coax. (HFC)
→ CMTS = Cable modem transmission system
→ 2 Things = modulator (digital to analogue signal)
demodulator (analogue to digital signal)



DCCN Theory :-

→ Satellite Radio ⇒ wireless connection
→ Geo stationary satellites (36000 km) away from Earth,
Low Earth Orbitting (LEO) satellites.
→ Access in Enterprise and the home ⇒ Ethernet, Wifi, LAN.
→ Customer is connected with diff internet access networks ⇒

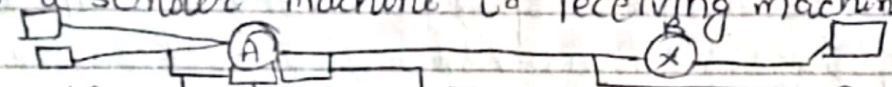


↳ DSL, FTTH, Dial-up etc ↳ Regional ISP's.

→ Internet Exchange Points ⇒ Routers and switches to provide access/connectivity ⇒ taking exchange point to content provider ⇒ energy / data centers ⇒ hundreds of connecters.

DCCN:-

→ LAN = (Local Area Network) - Limited geographical area (2 km), tech = Ethernet, Wifi, Token Ring, FDDI.
→ DSL types = VDSL, ADSL (mostly), SDSL, other FTTH, Cable, Dial-up.
→ ADSL = Assymetric = diff speed for up and downstream.
  (mostly user want and use + need downstream speed).
  ↳ It is changing to Symmetric (i.e. more upstream).
→ Wireless = Satellites → Geo stationary, LEO
  → Next shifting of internet = moving to stationary.
  ↳ Already started to shift (many big companies).
→ These tech will help to connect / access ISP.
→ To connect to content provider. (called access ISP).
→ Google does not need to pay for connectivity but access ISP pay to regional internet exchange point which is further connected to tear 1 (needs to pay).
→ This depends on BGP (Border Gateway Policy) (Router Policy).
→ LAN only provides Local but we need to connect to world.
→ Changing area also changes the technology.
→ Service providers to provide connectivity in diff areas.
→ MAN (Matropolitan Area Network) - possible to connect a network over a city / matropolitan).
→ WAN (Wide Area Network) = Very large area i.e. one in Pakistan other in China.
→ SAN (Storage Area Network) = data centers use, it is only for a specific purpose (storage or placed).
→ VPN (Virtual Private Networks) = to make communication private on a shared network i.e. try to protect data while sending so others can't understand (can not have your own network so use public network).
→ MANET, VANET, IoT (everything has IP address) (Internet of things), BAN (Body Area Network) + more.
→ SDN's (Software Design Networks) → we are shifting towards SDN's to make networks more programmable.
→ SAN for storing or moving data (data segmentation, backup) (every company has data center using SDN).
→ 10-15 km away from main site to keep data safe.

→ VPN types ⇒ Access VPN, Intranet VPN, Extranet.
→ Access VPN = 1 end is mobile, other is fixed.
→ Intranet = B/w a similar system (Comsats LHR + ISB)
→ Extranet = B/w two diff. systems. (USA to Comsats).
→ Data is encrypted before sending to other place.
→ VPN application is diff ⇒ requests send to other servers.
→ Bandwidth = capacity of network (no. of bits per sec)
  ↳ Characteristic of medium and signal (data rate is diff).
→ Bits (small b), Byte (large B) → General representation.
→ The actual measured rate is called throughput.
→ The actual bandwidth speed that we actually receive
→ Bandwidth is promised rate but throughput is actual
→ Why we don't get promised rate ⇒ many diff factors:
→ Delay in network (which occurs when sending data from a sender machine to receiving machine on other side).
→ 

Processing delay ↳ Queuing delay ↳ Transmission delay ↳ Propagation delay
→ So, Total delay = $d_{proc} + d_{queuing} + d_{transm} + d_{prop}$.
→ Processing = Time taken by router to check errors.
  ↳ When router receives a frame it needs to check (errors).
→ Transmission = Time required to transfer a packet, can be calculated by size of packet divided by rate
→ Transmission delay = Size of Packet / Rate of sending network.
  ↳ Complete message - i.e. all bits should be received.
→ Propagation delay = Travelling time, calculated by ⇒ Propagation delay = Distance covered / speed of packet.
→ Distance can vary if we change destination.
  ↳ Speed / Rate can also vary due to many factors.
→ Queuing delay = defined in terms of / dependent on traffic intensity ⇒ when packet enters queue of router it takes some time before transfer.
→ If queue if full ⇒ Traffic intensity = $(\frac{La}{R})^{N-1}$
→ Traffic intensity = $\frac{La}{R}$ ⇒ Packets have 'L' bits or L = bits per packet, a = arrival rate of packet

→ Same size of code = peer to peer architecture.
→ Some on one, more on other = client - server.
→ Thin/Flat client server architecture depending on where we place our design components.
→ Same = P2P, Diff = client - server.

Or rate at which packets arrive is called 'a'.
And R = bits/sec (how many bits per second).
→ If $\frac{La}{R} > 1$ → it means the arrival rate if greater than the transmission rate (R) i.e. (a > R).
→ Which results in queue being full → more queing delay.
→ So network design principle → make sure that $\frac{La}{R} \leq 1$. (promised rate).
→ Hence delay is a very Imp factor why we don't get promised rate.
→ Other factors = choice of network devices, type of data being transferred (downloading a text file vs watching a cricket match live), (based on speed our quality / playback speed is adjusted), Network topology (physical + Logical) (choice of topologies), No. of users on network (more users = more collisions) (we all use shared connections / not dedicated connections hence speed decreases for more users), User computers (if machine is old or malfunctioned), Server computer (more load on free server as compared to paid server) (sending rate of server effects), Power conditions (SNR = Signal to Noise Ratio = $\frac{S}{N} = \frac{\text{Signal}}{\text{Noise}}$ → effects intensity / quality of signal) (if $\frac{S}{N} = 0$ → it means that very much noise), (if $\frac{S}{N} = \infty$ → it means no noise = ideal condition), (very important factor to calculate data rate of network).
→ Application Layer :- (Ch2) → Network applications
→ 1970's - 80's → Text based Nt apps → Emails, newsgroups.
→ 1990's → Killer apps → World Wide Web → Search, ecommerce.
→ 2000's → VoIP (Voice over IP) e.g. Skype, Video over IP (Video conferencing), online multiplayer gaming, online content publishing (Youtube and many other apps).
→ Presently → Social Networking and still changing.
→ Apps are also changing → now more engaging (new algorithms = smart) (Facebook changed in 10 years).
→ Diff b/w HTTP / HTTPS / HTTP1 / HTTP1.1 / HTTP2.0
→ Network apps → one end peice of code on one machine and peice of code on other machine.

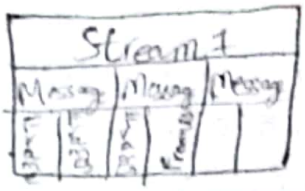
DCCN:-

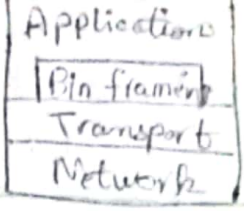
→ A part of application runs on client and other part on server = client server application.
→ Peer to Peer (P2P) = all clients have same amount of code running on them (multiple clients).
→ Hybrid = combine other 2 (examples Skype, Whatsapp).
→ Web 2.0 = Centralized (client server architecture).
→ LLM = Large Language Models (are not running on our machines as they can't handle) → based on supervised learning, we have done labelling for last 10 years through the help of machine learning.
→ Semi-Supervised labelling, Unsupervised labelling (where machine is intelligent enough to label itself).
→ Still we are trying to move to decentralized models (web 3.0) that do not have a specific/single authority.
→ P2P ⇒ same peice of code running on each code.
→ Most of apps we use are based on client-server.
① → Difficulties ⇒ Not ISP (Internet Service Provider) friendly.
→ [client] ⟷ (Upstream / Downstream) [ISP] — {Internet}
→ Most of time, we are downloading/receiving data.
→ Upstream and Downstream traffic ⇒ ISP's were built to provide more downstream traffic.
→ We mostly use assymetric data rate.
① → Challange = we need such P2P apps that are ISP friendly (to manage upstream + downstream).
→ Client always sends analogue data which needs to be converted to digital by ISP (as they connected digitally)
② → Security (we are safe with google accessing our data and saving, but we are not ok with random person).
→ So we need to come up with guarantee of security service
③ → Incentives (big companies give some services free and if we want we can purchase completely).
↳ As upstream and downstream are independent channels ⇒ so it does not have a mechanism for making money / providing incentives.

→ Web 3.0 ⇒ Decentralized (Peer to Peer P2P).
↳ E.g. Blockchain = Immutable, Verifiable (used for security) ↳ Cryptocurrencies (cannot be stolen/changed/exchanged due to diff. security no.s).
→ Incentives = one can make money such as by using NFT (Non-fungible Tokens or Fungible/unique).
→ Share my Computational + Storage resources.
↳ Because anyone wants to earn money by investing.
→ Peer to Peer = self scalable applications.
→ We are coming up with such tech that are symmetric.
→ Challenges ⇒ there is a fear of centralization or regularization (is it allowed in all countries or are all of them ready to regularize it (No)).
→ Scalability (bandwidth/computational power of each machine) (we are still trying to move to 3.0).
→ For every application a process is created which is then sent to TCP layer (transport layer) for transferring which is done by sockets.
→ Sockets are API's (Application Programming Interface) door for data transfer from app layer to transport layer (logical address).
→ We cannot control most of the things in transport layer (only change size of buffer etc).
→ It helps us in multiplexing/demultiplexing. (socket)
→ Every process has a unique address which is called port number (but why we cannot use the process Id that was generated by PC/OS).
→ Because it will then be OS dependent (machine).
→ Apps where we cannot accept data loss (these are called data sensitive applications), other type of apps are data insensitive for which data losses are [redacted] acceptable.
→ Other category ⇒ Time sensitive/insensitive.
↳ Is delay acceptable or not.

→ Throughput based - Elastic - uses whatever throughput they are getting), some other apps have very strict regarding the data rate to perform operations. (i.e. throughput they are getting).
→ Security (encryption needed or not needed) (data integrity)
→ All of these services are provided at transport layer
↳ To ensure data integrity (TCP), to counter delay (UDP), for security (SSL/TLS) (all of these can be met by providing some tech services at transport).
→ Apps will not be able to function properly if they are not getting their required throughput.
→ E-mail, Web, remote terminal access, file tranfer (all use TCP + SSL/TLS so they are HTTPS)
→ Protocal defines syntax, symantics and sequence of exchange (this is in RFC - Request for Comment).
→ Streaming multimedia (TCP or UDP), Internet telephondogy (UDP)
→ Connection-Oriented (TCP) - means acknowledgement
→ Flow control - if more data send than the capacity of receiver, then data will spill over. But if too slow than the buffer can overflow.
→ UDP is mostly used for high-speed apps.
→ Web and HTTP (RFC - 216, 1945 define this).
↳ Each object is addressable by a URL (www.some.com).
→ When we enter URL, we request the server to return all Web objects on that URL. (base page is included).
→ Web - data sensitive (HTTP request → HTTP response).
→ To send and receive HTTP, we need to establish a HTTP connection (i.e. 3 way handshaking) (TCP connection by sending some data and receive acknowledgement)
→ For HTTP (Port - 80), for HTTPS (Port - 443).
→ To ensure data integrity / loss is taken care by TCP.
→ Request has → (For HTTP 1.$\overset{any}{X}$) (request line, header lines, Carriage return) and response has (response lines, header lines and message i.e web objects).

→ 505 Version not supported (HTTP)
→* It also has connection (open/close) and Accept language command
→ Request line = GET, URL and version of HTTP.
↳ In header lines ⇒ contain Host (URL) (this is used for Proxy servers)(or web cache) (otherwise it is not needed) then comes User-Agent (i.e. the browser being used) (2 types of connections persistent or non-persistent) (this is a non-persistent as at removes connection after request and response) (Persistent will not be removed).
→ In response ⇒ status line (200 = o.k., 404 = page not found, 301 = moved permanently, 401 = not supported version) then header lines (date, server, last modified, Content length and type), then there is data.
→ HTTP/1.x = Pull Protocol and Stateless.
Is We pull the data from server. (400 = bad request).
→ E-mail is a type of push-protocol.
→ In line ⇒ sending the expected required data.
⇒ The server does not maintain its state (what was accessed previously by users) (more burden).
→ TCP connection ⇒ sender sends SYN segments while the server sends SYN Ack segment. (We do not maintain any state).
→ HTTP/2.0 = push protocol = server directly/automatically sends needed files (no need of inline exchange).
→ Connection close :- means that we are using non-persistent.
① Client initiates TCP connection to HTTP server at port 80.
② Server accepts connection and notifies client (SYN ack).
③ Client sends HTTP request to server. ④ Server sends respond message.
⑤ Server closes TCP connection.
⑥ Client receives file and decodes + displays.
→ This approach was used in HTTP/0.9/1.0/[redacted].
→ Response time total = 2RTT + transmit time.
→ RTT (Round Trip Time) ↳ for non-persistent.
→ By using Persistent connection we can avoid RTT (only happen once). (can send concurrent requests) (means using pipelining).



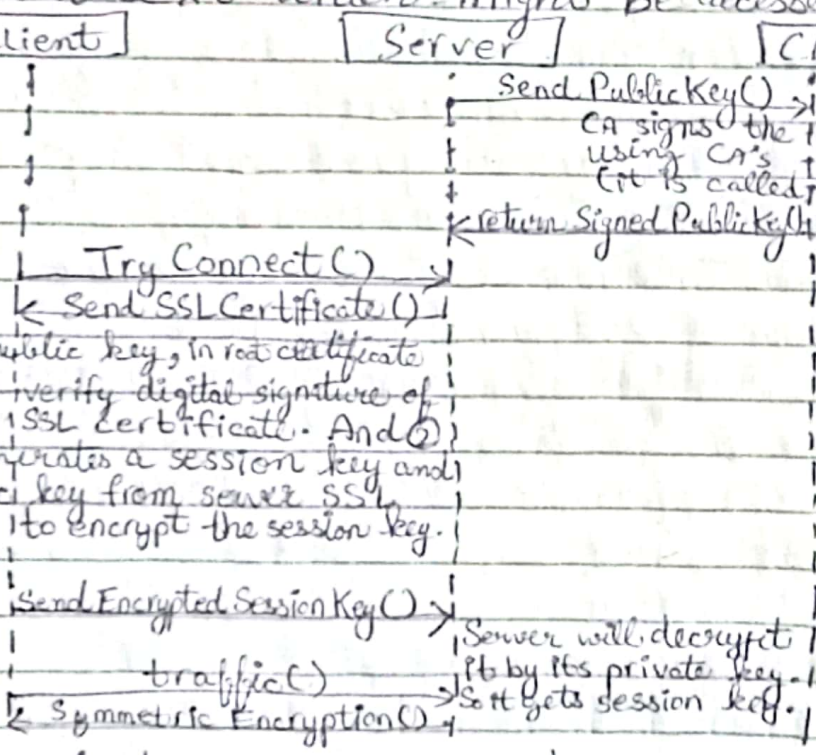


→ Non-persistent based on stop and wait.
→ But what if the 1st request is not yet completed when the 2nd request is already sent. (This is called HOL - Head Off Line). (Head of Line).
→ By using pipelining we want concurrency.
↳ This approach was used in HTTP/1.1.
→ If 1st is not entertained the others are held in queue.
→ Pipelining is better than stop and wait.
→ HTTP/1.X ⇒ for concurrent requests we need concurrent TCP connections (clients needs to use multiple connections to achieve concurrency and to reduce latency (can be improved by only using 1 (same) TCP connections for multiple requests) → HTTP2.0
→ HTTP/2.0 ⇒ we can send multiple requests and receive responses in the same TCP connection. (gzip)
→ HTTP/1.X ⇒ Does not compress request & response headers
→ HTTP/2.0 ⇒ We use headers field compression (HPACK).
→ HTTP/1.X ⇒ There is no request prioritization. (It does not allow effective resource prioritization).
→ HTTP/2.0 ⇒ It gives us Binary Framing layer (it extends the application layer) (not exactly new).
→ Request Line (Eg.) ⇒ Post /upload / HTTP 1.1
→ Header Lines ⇒ Host - www.cuilahore.edu.pk
Then connection Type ..... , Then Data .....
→ Header Frame
Data Frame
(This is binary frame).
→ 3 imp. concepts in HTTP/2.0 ⇒ streams, messages, and frame.
→ Stream = bidirectional flow of bytes (or data).
→ We can send n number of streams in one TCP connection. (Stream has messages which contain frames).
→ We send request and receive response in same stream.
→ By using binary framing and streams we can avoid HOL (head of line) problem.

→ It uses a random number and a cypher to make key.
→ We use symmetric keys (session keys are also known as).
→ Every browser has public key pre-loaded.
→ Public keys are known to all but private keys only to person.
→ Frame has a 9 byte header. (R = Reserved)

Bit	0-7	8-15	16-24	25-32
0	[Length		]	[Type]
32	[Flags]	//	//	//
40	R [Stream		Ident	ifier]
...	Frame Payload			

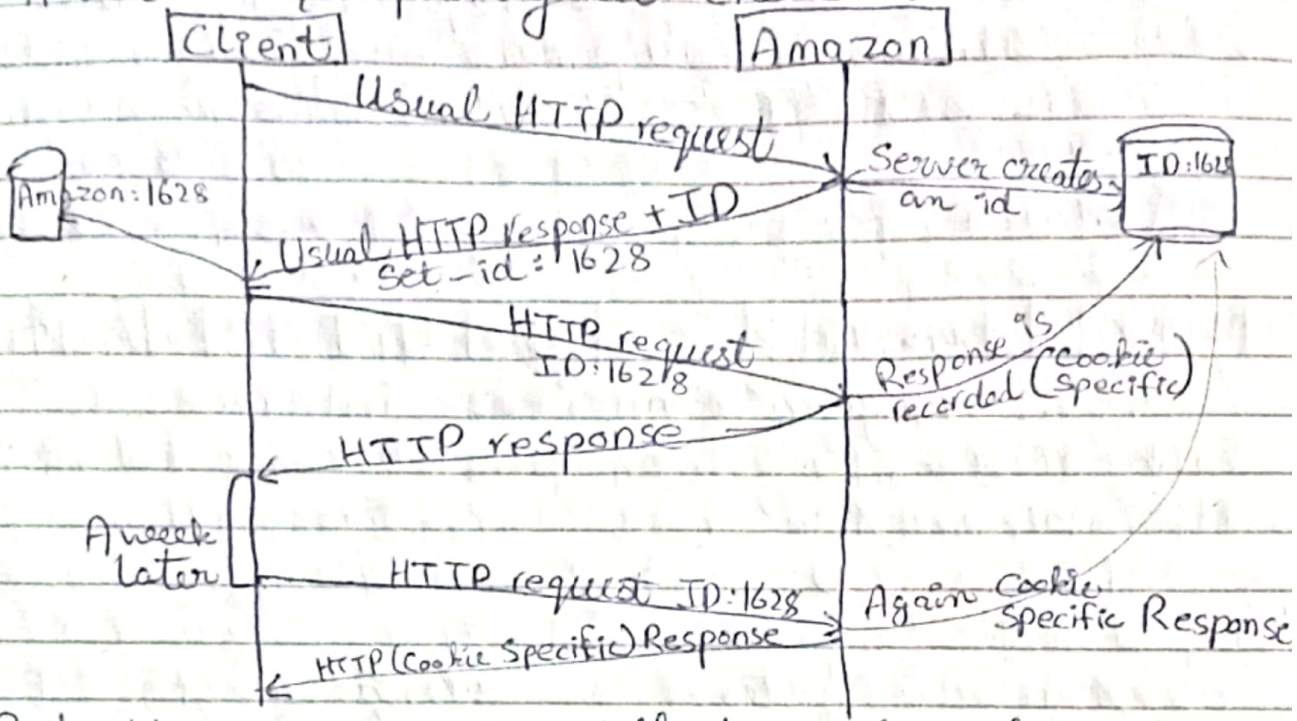
→ HTTPs - where s stands for security.
→ Secure Socket Layer (SSL) (old standard),
Transport Layer Secure (TLS) (new standard which supports better encryption algorithms).
→ Both of these technologies are in transport layer.
→ If none of them is used, data will be sent in plain text which might be accessed by anyone.



→ The next time server sends a message, it will use symmetric key and only client will be able to decrypt it (as client + server have same symmetric key).
→ When we (client) decrypts the SSL certificate, it gets the public key by using public key which is installed in your browser. This is done to verify that server is what it is claiming to be.
→ User-Server interaction :- Cookies (following 4 things needed):-

→ But cookies are controversial as they have invasion in privacy.
→ All of the activities that we do on website are being recorded.

① → A cookie header line in HTTP response message.
② → A cookie header line in HTTP request message.
③ → A file kept at user's machine by browser.
④ → A database / repository at the server.

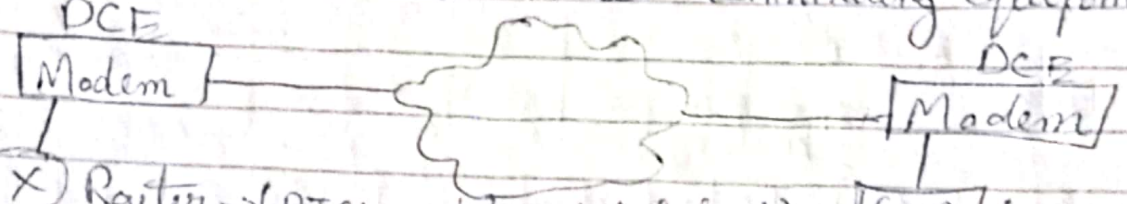


→ But there are many diff. types of cookies.
1) First Party cookies = maintained by primary website or server (e.g. google itself).
2) Third Party cookies = google has an add from Amazon and it is recording my activity (i.e. not maintained by primary server) (any other).
3) Session Cookies = Our selection that we did remain stored until our session is expired.
4) Persistent Cookies = always are stored even after session ends (can be deleted easily).
5) Tracking cookies = giving suggestions all the time.
6) Syper cookies = Maintained always in our machine (cannot be deleted easily) (very difficult to remove from browser) (use anti-virus to forcefully remove these cookies).

→ 2 Routing protocols to do :-
1) RIP (Routing Information Protocol).
2) OSPF (Open Shortest Path First).

→ Steps for RIP :-

→ Connect 2 routers by serial interfaces by using a serial cable. (On one side of connection we have a clock which represents a DTE (data terminator equipment) and other side becomes DCE (Data circuit terminating equipment)).



→ First configure connected interface fa 0/0.

→ The se 1/0 interface represents same network.

→ RIP uses Bellman Ford Algorithm.

→ OSPF uses Dijkstra's Algorithm.

→ OSPF need a process ID, and it uses a wild card mask instead of subnet masks and wild card is exactly opposite of subnet mask, it is also used in access control list.

→ In OSPF we divide the network into areas.

→ Area 0 = backbone area.

→ Bellman-Ford = Decentralized Algo (belongs to class of dynamic programming) - used in RIP.
→ Dynamic = break the problem into smaller problems, then solve and combine (so distributed / decentralized).
→ One machine only knows about its own neighbours.

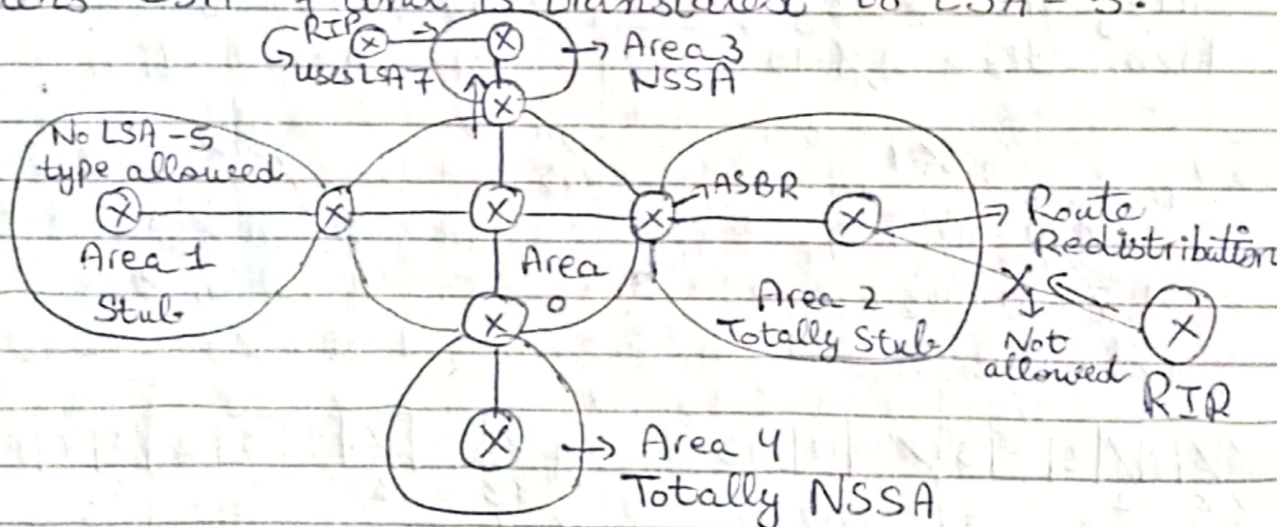
→ OSPF has information about all machines and cost to reach them (hence it has a very large table) This is global topology; so we try to avoid this by dividing network into areas. 0 - backbone area → main area from where data is send.

→ We can have 5 diff types of areas in OSPF:-

1) Backbone area (area 0)
2) Standard area
3) Stub area
4) Totally stubby area
5) Not so stubby area (NSSA)

→ Area 4 (Totally NSSA) ⇒ No type 3 LSA ; No type 5 LSA, But ASBR is allowed.

→ In OSPF, routers send updates to each other that are called Link State Advertisements (LSA).
→ LSA 1 ⇒ Router → advertising about another router.
→ LSA 2 ⇒ Network → advertising about connected network.
→ LSA 3 ⇒ Network summary → share the summary of ~~(routed)~~ connected network (s).
→ LSA 4 ⇒ Autonomous Systems Border Router (ASBR) summary ⇒ defines the routers connecting autonomous networks
→ LSA 5 ⇒ External summary.
→ LSA 7 ⇒ Advertises the external summary.
→ Network other than OSPF sends message, it enters LSA-7 and is translated to LSA-5.



→ Every other area is connected with back bone area (so data is routed from this area) (always).
→ If no areas ⇒ every router has info of every other ⊗.
→ In area 1 (stub) ⇒ No LSA5, No ASBR allowed.
↳ Hence it only contains info of routers within this area.
→ In area 2 (totally stub) ⇒ No type 3 summary / LSA, No type 5 LSA, No ASBR (any router connected by other than OSPF cannot communicate). And The network summary is also not allowed in area 2.
→ In area 3 (NSSA) ⇒ No type 5 LSA but we have type 7 LSA (RIP router can do route redistribution by using type 7 LSA which is converted to LSA 5). But ASBR is allowed (ASBR summary).

DCCN:-

→ Proxy Servers = store a copy of results (web objects).
  → Copy is not the same (it may have diffg. versions).
→ The conditional GET - gives newest or most updated version of our web results and stores.
→ Server generates a new HTTP header that contains:
  → Last modified: Date_Time (when request is forwarded)
  → if modified since: Date_Time (it contains no data/web objects).
→ For example ⇒ 304 Not modified (is such a header).
→ A proxy server is a server and client at same time.
→ Cache with Content Distribution Networks (CDN's)
  → Private CDN's ⇒ E.g. Google
  → Third Party CDN's ⇒ E.g. Limelight, Akamails
→ Data center placement policy (How to setup, where to set data center, can be centralized or other).
  ① → DataCenter placement philosophies :- (Types)
  → 1) Enter-Deep :- (Access ISP's (stormfiber, PTCL) connected with regional).
    → Place the data center near the access ISP's.
    → Concept of Akamai's ⇒ 1700 Data Center Clusters.
    → But monitoring and management is very difficult.
  → 2) Bring-HOME :- Deploy Data Center clusters near Tier-I ISP's (AT&T, Verizon, ...) (few miles away).
    → This approach is used by Google (has 8 big data centers)
    → (6 in US, 2 in EU) (each data center has 100,000 servers) +
    → It also has 300 Bring-HOME (each has magnitude 50 to 100 servers).
    → And it also has hundreds of Enter-Deep (magnitude of 10's).
→ The big (8) data centers are used to send personalized content, gmail messages, search results etc (Dynamic).
→ 300 bring-HOME servers are used for static video (Youtube).
→ Hundreds of Enter-Deep used to send static web content.
→ These hundreds of data centers are proxy of google.
→ When a request is made, how is CDN selected?
  ② → CDN Selection Strategies :-
  → i) Selection based on Geographical Location of LDNS Server. (LDNS = Local Domain Name System) ⇒ MaxMind, Quora.

→ But most clients are using remote DNS servers, i.e. client is somewhere else and has a remote DNS placed in a different location (so not a good strategy).
→ 2) Measuring link parameters i.e. Delay, throughput by using SYN, SYNACK, or DNS Queries.
→ Clients is doing Rate determination Algorithm while watching a movie on NETFLIX (so he can always change server / data center). NETFLIX uses DASH (Dynamic Adaptive Streaming over HTTP).
→ NETFLIX stores each video on Amazon's data center as a chunk of 4 seconds (based on connection provides quality).
→ 3) Any cast IP → content provider gives access to two or more servers (same IP) based on no. of hopes.
↳ Giving same IP's to two servers of two independent networks and then selecting on no. of hopes required to reach that data center.
i) → Content Ingestion = NETFLIX uplods its content on Amazon.
ii) → Content Processing = Amazon converts uploded files into different formats (so that DASH can be used).
iii) → Content Placement = Amazon places that content on different CDN's servers (not actual / all data) (proxy).
→ Client gives payment ⇒ NETFLIX verifies → then Amazon sends manifest file (Links of all data / content),
→ and then we place CDN and access data.
→ File Transfer Protocol (FTP) ⇒ uses Port 20/21.
→ Out of band because 2 seperate controls (TCP's).
→ HTTP, Email ⇒ In band as only one connection (TCP).
→ ↳ Stateless ⇒ does not save the state of data.
→ FTP is statefull (state of data is also stored).
→ Email (useragents, mail servers, SMTP = simple mail transfer protocol). (Always uses port = 25).
→ How mail is maintained on server ⇒ using POP3, IMAP4.
→ POP = Post Office Protocol, IMAP = Internet Mail Access Protocol.
→ User Agent ⇒ can be browser,

→ Next Thursday Quiz.

→ Gmail uses HTTP connection to transfer mails.
→ Email is a push protocol ⇒ when a new message arrives ⇒ it pushes the message into user agent.
→ HTTP is also moving towards Push protocol but at the moment it is based on pull-protocol.
→ Comparison b/w POP3/IMAP4, and HTTP/Email
→ In POP3 ⇒ we download a complete message i.e header and body but in IMAP4 ⇒ only header is downloaded so you can make a decision if we want to fetch body.
→ POP3 is heavy whereas IMAP seems light on server.
→ But if an ISP wants to implement one of them, there are so many factors that needs to be checked
→ Console cable ⇒ to open settings of switch directly.
→ Or connect directly with switch (secure shell - telnet's software form)
→ Enable → Config t → select interface vlan 1 (virtual intf).
→ We establish virtual connections with servers (at most 16).
→ vlan 1 is not an actual interface it is a virtual intf.
→ Give IP address → no shut down.
→ Now for telnet, do following steps:-
→ Line vty (virtual teletype) (max 16) 0 15 (logical)
→ assign password ⇒ password cisco → Log in → end.
→ Then we use telnet to access switch from PC (cmd)
→ SSH encrypts message before sending (Packet Tracer - RSA).
→ Telnet sends text as plain text (so SSH = Secure).
→ en → conf t → change hostname by ⇒ hostname DCCN
↳ ip domain name DCCN → crypto key generate rsa, asks how many bits (e.g. 1024) ↳ for encryption
→ As it has 2 versions ⇒ enable version 2 ⇒ ip ssh version 2.
↳ Then line vty 0 15 → password → transport input ssh
→ In command prompt ⇒ ssh -l DCCN 192.168.1.1
→ Console cables are rare so configurations through normal port are frequently used (as ethernet cables are now less, so we use console to USB port).

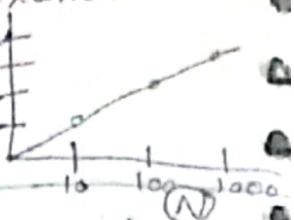
DCCN:-

→ SMTP uses persistent connections (push protocol).
→ SMTP requires message to be in 7-bit ASCII.
↳ But no such restriction in HTTP as it can use Unicode.
→ SMTP = one message for many objects (multipart).
whereas HTTP = one obj = one message (multiple objects by multiple streams).
→ POP3 (Port 1103), IMAP4 (Port 143) are used.
→ POP3 = messages downloaded, IMAP4 = uses headers + other parts.
→ When we access e-mail over web, which is it like = POP3 or IMAP4?
→ DNS (Domain Name System) is a Distributed Database in which we keep record of IP address's (domain name).
→ In IPv4 = only 1 IP needed, In IPv6 = 2 IP needed.
→ So DNS needed to map IP address with host/domain name.
→ Design Philosophy of DNS ⇒ why is it not centralized?
→ 1) Single Point of Failure. (not equidistant from every user can effect the speed of connection).
→ 2) Distant Centralized Database.
→ 3) Maintainence (If it is down for some time = many problems).
→ That is why DNS is a distributed database.
→ Other advantages (excluding above 3 points):-
→ 1) Host Aliasing (we can access either by gmail.com or www.gmail.com) (both are mnemonics).
→ In reality (relay1.west-coast.gmail.com) (Canonical).
→ Two types of host names ⇒ mnemonics, canonical.
→ Host Aliasing means to map 2 or more mnemonics host names against a single (one) canonical name.
→ 2) E-mail Server Aliasing ⇒ we can use same name for e-mail as well as for web server.
→ enterprize.com can be used as a web server and as e-mail as well using Record type ⇒ MX records → (Mail Exchange).
→ 3) Load Balancing ⇒ (Automatically) As we have replicated Servers (for large networks) (shared). (There are many different servers being used so when we access, we receive a list of

IP addresses) (hence the servers are also shuffled so only one is not used) so balance load.
→ It is not only distributed but hierarchical as well.
→ There are 13 root DNS locations and each 1 of them has many servers. Most of them are located in North America
→ Root DNS servers ⇒ keep track of top level domains.
↳ i.e. com DNS servers, org DNS, edu DNS (all TLDs)
→ These ~~to~~ DNS keeps tracks of authoritative domains.
↳ e.g. com DNS → yahoo.com DNS server, amazon.com.
→ Then these authoritative keep track of local DNS (that we are interacting) (many servers).
→ These are called Local Name Servers.
→ Host (cis.poly.edu) → sends request to local DNS (dns.poly.edu) (when it does not have that IP, it sends request to root DNS) → root DNS sends a response back to local DNS and sends IP address of required top level domain → local sends to that TLD server → TLD responds by sending desired IP of authoritative → local sends to that authoritative → which sends IP of system to local DNS → local DNS gives this IP to our machine so that using this IP both machines can make a connection.
→ A request sent corresponds to a DNS query and it receives a DNS response. (above mentioned)
→ DNS queries are iterative in nature (this type).
→ There is another type of DNS queries that are recursive ⇒ local sends request to root → root sends to TLD → TLD to authoritative → which sends to system and then same path is followed for every response till it reaches PC.
→ Which one of these is better to use?
→ Databases keep records of data so the data

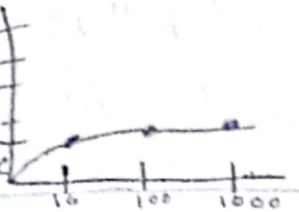
→ 1 Question from Wireshark in Lab Mid exam.
↳ This is just like Word.

→ (Need to increase servers). (Client – Server) →



stored in DNS is called Resource Records (RR's) (There are total 61 types of RR's).
→ It contains (Name, Value, Type, TTL).
→ Type ⇒ e.g. "A" record = Address Record
↳ "A" for IPv4, "AAAA" for IPv6.
→ E.g. DCCN.com, 192.168.1.1, A, 14400 (seconds).
→ 2nd Type ⇒ NS (stands for Name Server).
→ If there is AAAA, then 2 IP address will be in value.
→ In NS ⇒ DCCN.com, dns.DCCN.com, NS, 14400.
→ 3rd Type ⇒ C name (C stands for canonical).
→ C name ⇒ (DCCN.com, relay1.west-coast.DCCN.com, cName, 1440);
→ 4th Type ⇒ MX record = Mail Exchange.
→ In MX ⇒ gmail.com, relay1.west-coast.gmail.com, MX, 14400).
→ Some types have even more values / fields e.g. some types add a priority field.
→ Type determines what will be name and value.
→ P2P apps are ~~soft~~ self scalable so does not crash.
→ client – server can crash while sending large amount.
→ Every server has upload rate / sending capacity.
→ Every client has download capacity + & upload.
→ Time to send one copy of file ⇒ file size / upload.
↳ F/us (us = upload capacity of server).
→ For sending N no. of copies ⇒ NF/us.
→ Client download time ⇒ F/dmin (min. download capacity).
→ Time to send file to N clients using client – server.
$D_{c-s} \geq \max\{NF/u_s, F/d_{min}\}$ If N increases then server takes more time to send (i.e $\frac{NF}{u_s}$ ↑).
→ Distribution time increases linearly when N increases.
→ When using P2P many machines are sending file(s).
→ F/us and F/dmin remain same but as every client is contributing in file distribution process so for NF bits ⇒ max sending rate = $u_s + \sum u_i$
→ $D_{P2P} \geq \max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$
→ In N is increasing so is the no. of sending machines.

P2P ⇒



→ F = 30 Gbits ; Us = 60 Mbps ; dmin = 4 Mbps ;
Ui = 400 kbps, 700 kbps, 2 Mbps ; N = 10, 100, 1000.

→ Make a table as follows for P2P + C-S.

Ui(s)	10	100	1000
400k	7680	51200	512000
700k	7680	51200	512000
2Mb	7680	51200	512000

→ for client-server.

⇒ For Client-Server :-

⇒ $F/Us = 30 \times 1024 / 60 = 512.$

⇒ $F/dmin = 30 \times 1024 / 4 = 30 \times 256 = 7680.$

⇒ $N_1 F/Us = 10 \times 30 \times 1024 / 60 = 5120.$

⇒ $N_2 F/Us = 100 \times 30 \times 1024 / 60 = 51200.$

⇒ $N_3 F/Us = 1000 \times 30 \times 1024 / 60 = 512000.$

⇒ For P2P :-

⇒ $F/Us = 30 \times 1024 / 60 = 512$

⇒ $F/dmin = 30 \times 1024 \times 1024 / 400 = 78643.2$

⇒ $\frac{F}{(Us + \Sigma Ui)} \Rightarrow \frac{30 \times 1024}{60 + (\frac{400K}{1024} + \frac{700K}{1024} + 2\ Mb)}$

⇒ $\frac{F}{Us + \Sigma Ui} = \frac{30720}{60 + (0.390625 + 0.683594 + 2)}$

DCCN:-

→ P2P are self scalable, better stability, e.g. Bittorrent.
→ If you download a torrent, it contains group of peers/ list of peers exchanging file in that Linux distribution.
→ A peer contains 50 or less IP addresses (list) of PC's.
→ Based on tit for tat or give and take principle.
→ File is divided into many chunks of size 256Kb.
→ Not necessary that every peer has all chunks.
→ First of all the downloader makes connection with 4 PC's and requests them to send file.
→ Optimistically unchoked ⇒ other PC's give chunks in hope that the downloader that it will give also.
→ Downloader evaluate each of 4 neighbours after 10sec,
→ If that PC is not sending, it will be replaced.
→ After 30 sec, downloader optimistically unchokes another PC in hope that it will give data in return.
→ If this one is giving more data then it replaces existing.
→ Random unchoke ⇒ rotating and randomly choosing PC's (every 30s)
→ Which file to request from a neighbour ⇒ rearest one.
↳ e.g. 3 PC's have file 2 but only PC 1 has file 1 so request PC 1 to send file 1 as it is the rearest one.
→ We hope that unchoked PC provides file at higher rate.
↳ If it is providing at higher rate then replace PC.
→ Hence we need to find such peers that provide higher rate.
→ But the selection of neighbours is very important and it does not only depend on higher transfer rate, also depends on what chunks of file PC has.
→ Condition ⇒ Assuming that $n_b \geq n_a$, then
⇒ $\frac{C(N-n_a, n_b-n_a)}{C(N, n_b)}$ ⇒ to know whether alice has those chunks that bob has.
⇒ $Prob(n_a) = \sum_{n_b=n_a}^{N-1} \frac{1}{N} \frac{C(N-n_a, n_b-n_a)}{C(N, n_b)}$.
⇒ To see if bob has a chuch that interests any of his 4/5 neighbours, we use -
⇒ $P = \sum_{n_a=0}^{N-1} \frac{1}{N} P(n_a)$ ⇒ for calculating common chunks
⇒ for uncommon 1 − this. (P)

⇒ Combination notation :-

⇒ $\binom{n}{r} = C(n,r) = \frac{N!}{r!(n-r)!}$ (ncr where n selects r).

↳ binomial co-effecient (using binomial theorem).

→ E.g. calculate prob. that head occurs 3 times out of 5.

→ $P(\text{head}) = \binom{5}{3} = \frac{\binom{5}{3}}{2^5} = \boxed{\frac{6}{32}}$ → Probability

→ The selection of neighbours determines the quality of bit torrent algorithm. (Rate comes afterwards, first we need to make a connection (neighbours)).

→ We want to design a P2P app as database i.e. which stores data on peers / devices connected.
↳ i.e. we want to place a database on peers.

→ There is a record (key, value) which needs to be stored or read from peer e.g. (Video file, 210.20.15.3).

→ We need to find where to store and from where to read (two problems ⇒ storage and retreival).
↳ As there are millions of peers (so need to decide).

→ One solution ⇒ store file on all PC's and they have IP's of others
↳ But this is not scalable and will not work for more PC.

→ Second solution ⇒ peers will always be in range $(0, 2^N - 1)$ and key is also in range $(0, 2^N - 1)$.

→ As key is a video file / name i.e. not a number, so we need to convert it using hash function (hashing).

→ Then store that key on PC with equivalent number or the number which is closest to key number.

↳ Always choose successor if exact not exists (not predecessor).

→ But for retreival ⇒ still a problem, as to retreive, one PC needs to have IP's of all other PC's.
↳ Other solution ⇒ PC only needs to maintain IP of its predecessor and its successor.

→ This way of storing is known as Distributed Hash Tables (DHT's). After applying structure for retreival ⇒ we use Circular DHTs.

→ But still for worst case it will have to travel n number of devices ⇒ again not good enough.
→ So, in addition to its predecessor and its successors, it can have info of some other PC's.
→ 2 types of networks ⇒ underlays (using switches or other devices), overlays (only use logical connection for connecting devices).
→ Now this becomes overlay DHT.
→ How many PC info should be kept → so that the time complexity never goes over O(log N).
→ Skype is another example of P2P using supernodes + other.
→ Applications can be data sensitive or time sensitive.
→ Transport layer provides process to process communication, responsibility is to provide logical communication b/w processes running on different hosts ⇒ using segments and source port number + destination port number.
→ Two main important protocols TCP, UDP.
→ TCP = reliable as it has acknowledgement, it provides us with byte stream, 3 imp factors congestion control, flow control, connection setup (using connection cookies).
→ UDP ⇒ unreliable, not in-order delivery but it has delay guarantees and bandwidth guarantees.
→ Multiplexing / Demultiplexing ⇒ As we use multiple apps that are sending and receiving data, so how we should differentiate b/w the data for these diff apps.
↳ Hence multiplexing is used, Application Program Interface / sockets are changed for communication.
→ Diff ports / sockets are used for diff applications.

DCCN:-

→ Sockets can be of 2 types ⇒ TCP socket or UDP socket.
→ ↳ It has Port number and process address of machine.
→ TCP socket has 4 entities ⇒ Source/Dest IP + Port Number.
→ So UDP socket - 2-Tuple, and TCP socket = 4-Tuple.
→ User Datagram Protocol = no order, not connection oriented, but fast, delay-guarantees, no need for establishing a connection (some apps we do not need reliability such as for sending a DNS query).
→ We keep track/record of some variables at client + server.
↳ For TCP, but in UDP we do not need to keep any record.
→ The header size of UDP is also small (usually 20, 40, 60 Bytes)
→ UDP has simple header (it has source port #, destination port #, length in bytes (process address), checksum (to calculate error bits).
→ Checksum ⇒ 1100 0011 ⇒ 1100 + 0011 = 1111 ⇒ 1's complement ⇒ 0000 → Checksum
↳ Then ⇒ 11000011 | 0000 | Data | EDC → Error bits.
→ Then at receiving end it adds all data segments with EDC bits and if 1's complement of answer = 0000 then no error.
→ Why we need to check for error in every layer?
↳ At the time of storing data at routers, there can be errors, so to make sure that stored data has no errors we check.
→ Most of internet traffic is based on TCP (75% apps are also based), it is reliable
→ UDP has no congestion control mechanism, so TCP is used.
→ Network will experience deadlock, blockade if we keep sending too much data without any congestion control.
→ TCP Header ⇒ contains Source + Destination port #, sequence number (starting data byte number as TCP is byte oriented) (e.g 1004 - 1499 ⇒ sequence no = 1004), acknowledgent number = (last byte no + 1) (so 1499+1 = 1500) (it means I have received all data bytes till 1499 so now send from start = 1500), header length (variable, not fixed = 20, 40, 60 bytes because of options field ⇒ if not used = 20B, if used ⇒ 40-60 bytes), Options field ⇒ Time stamp, Negotiate MSS (Maximum Segment Size ⇒ that can be put into segment of layer).

DCCN -

→ MSS is taken from MTU (Maximum Transmission Unit → The maximum size that can be send/transmitted by network).
→ Selective ACK - options Field. (to activate / enable SACK).
→ To redifine sequence number (using fast networks, sometimes seq. no repeats, so we try to avoid by redifining it in the options field).
→ One field not used, then 5 Flags → 1 for SYN flag ⇒ SYN segment sent, and SYN ACK flag set to 1 by receiver, to reset connection - R - reset flag.
FIN flag (F) ⇒ finish or terminate connection.
→ Q - Why we need (A) flag if we have acknowledgement no?
→ Push Flag (P) ⇒ if push is set then receiver pushes the packet to priority to application in its buffer/queue.
→ Urgent Flag (U) ⇒ set ⇒ urgent data is contained and placed at starting byte no, so tell where it ends using the urgent data pointer field (not used normally).
→ Receive window (available) ⇒ space left in receiver buffer.
→ Checksum, then application data (variable length).
→ Exam ⇒ Section 3.4 (Ch3), Ch 2 complete + any additional notes.