Computer Network(Mid)

Short Q: "What is a protocol in computer networks?"

*A **protocol** is a set of rules that define how two or more entities in a communication system exchange data.*

*It specifies **syntax (format), semantics (meaning), and timing (when/how to send data).***

Long Q: "Explain the need for protocols with examples."

*A **protocol** is a set of rules that define how two or more entities in a communication system exchange data.*

*It specifies **syntax (format), semantics (meaning), and timing (when/how to send data).***

Functions of Protocols

1. **Data Encapsulation** – Adding headers/trailers to carry control info.
2. **Addressing** – Identifying sender/receiver (IP, MAC, Port numbers).
3. **Error Control** – Detecting & correcting errors (TCP retransmission, CRC).
4. **Flow Control** – Ensuring fast sender doesn't overwhelm a slow receiver.
5. **Synchronization** – Establishing when communication starts/stops (handshakes).
6. **Routing** – Deciding best path (IP, BGP, OSPF).

Elements of a Protocol

1. **Syntax** – Data format/structure (e.g., header fields, packet length).
2. **Semantics** – Control information meaning (e.g., ACK = acknowledgment).
3. **Timing** – Coordination of speed & sequence (when to send/expect).

Cable-Based Access Networks

Cable-based access networks use **wired media** (copper or fiber) to connect homes/offices to the ISP.

Types

1. **DSL (Digital Subscriber Line)** – uses telephone copper lines.
2. **Cable Internet (HFC – Hybrid Fiber Coaxial)** – fiber in backbone + coaxial in last mile.
3. **FTTH (Fiber-to-the-Home)** – optical fiber directly to the user.

Multiplexing

- **FDM (Frequency Division Multiplexing):** Each user gets a frequency band.
- **TDM (Time Division Multiplexing):** Each user gets a time slot.
- **CDM (Code Division Multiplexing):** Each user gets a unique code; all transmit at once.

Infrastructure

- **HFC (Hybrid Fiber Coaxial):** Combines fiber + coax.
- **CMTS (Cable Modem Termination System):** ISP device managing cable modems, bandwidth, and converting signals to IP packets.

Wireless Access Networks

Wireless access networks use **radio waves or satellites** instead of cables to connect end-users to the ISP.

Types

1. **Wi-Fi (Wireless Fidelity):** Local wireless LAN for homes, offices, hotspots.
2. **Cellular Networks (2G/3G/4G/5G):** Wide-area coverage using base stations.
3. **Satellite Networks:** Uses geostationary or low-earth orbit satellites for remote access.

Access Techniques

- **FDMA (Frequency Division Multiple Access):** Users share spectrum by frequency bands.

- **TDMA (Time Division Multiple Access):** Users share by time slots.
- **CDMA (Code Division Multiple Access):** Users share by unique codes.
- **OFDMA (Orthogonal Frequency Division Multiple Access):** Advanced FDMA used in 4G/5G for higher efficiency.

Packet Switching vs Circuit Switching

Circuit Switching

- **Definition:** A dedicated communication path is established between sender and receiver for the entire duration of communication.
- Example: Traditional telephone networks.
- **Steps:** Setup → Communication → Teardown.
- **Features:**
  - Guaranteed bandwidth & quality of service.
  - Inefficient for bursty traffic (wasted resources if no data is sent).
  - Connection-oriented.

Packet Switching

- **Definition:** Data is broken into packets, each packet travels independently across the network, and is reassembled at the destination.
- Example: Internet.
- **Features:**
  - No need to reserve a dedicated path.
  - Resources are shared → efficient for bursty traffic.
  - Can face delay, jitter, and packet loss.
  - Connectionless (IP) or connection-oriented (TCP).

Why the Internet Uses Packet Switching

1. Traffic is *bursty* → packet switching shares resources better.
2. More scalable → thousands of users can share the same backbone.

3. More fault-tolerant → if one path fails, packets can be rerouted.
4. Cost-efficient compared to reserving dedicated circuits for everyone.

If Not Packet Switching → Circuit Switching?

- If packet switching didn't exist, we'd need to use **circuit switching**.
- But then:
  - Internet would be extremely inefficient (imagine reserving a full line just to send a WhatsApp message).
  - Limited number of users → scalability issues.
  - Waste of bandwidth during idle times.

Difference between Circuit Switching and Packet Switching

Circuit Switching	Packet Switching
In-circuit switching, each data unit knows the entire path address which is provided by the source.	In Packet switching, each data unit just knows the final destination address intermediate path is decided by the routers.
In-Circuit switching, data is processed at the source system only	In Packet switching, data is processed at all intermediate nodes including the source system.
The delay between data units in circuit switching is uniform.	The delay between data units in packet switching is not uniform.
Circuit switching is more reliable.	Packet switching is less reliable.
Wastage of resources is more in Circuit Switching	Less wastage of resources as compared to Circuit Switching
Circuit switching is not convenient for handling bilateral traffic.	Packet switching is suitable for handling bilateral traffic.
In-Circuit Switching there is a physical path between the source and the destination	In Packet Switching there is no physical path between the source and the destination

Feature	Circuit Switching	Packet Switching
Path	Dedicated path (once established)	Packets take different routes
Efficiency	Wastes bandwidth if idle	Efficient, resources shared
Delay	Setup delay, then fixed transmission	Possible variable delay (queuing, routing)
Reliability	Guaranteed QoS	Best effort, not guaranteed
Example	Telephone network (PSTN)	Internet (TCP/IP)

Delays in Packet-Switched Networks

When a packet travels through a node (router/switch), it experiences **4 types of delay**.

1. **Processing Delay (d_proc)**

The time required to examine the packet's header and determine where to direct the packet is part of the processing delay

2. **Queuing Delay (d_queue)**

- Time the packet spends **waiting in queue** before being transmitted.

3. Transmission Delay (d_trans)

- Time to push all packet bits onto the link.
- Formula:

$$d_{trans} = \frac{L}{R}$$

where

- L = packet length (bits),
- R = transmission rate (bps).

Example:

If packet size $L = 1{,}000$ bits, link rate $R = 1$ Mbps:

$$d_{trans} = \frac{1000}{10^6} = 0.001 \text{ sec} = 1 \text{ ms}$$

---

4. Propagation Delay (d_prop)

- Time for a bit to **physically travel** across the medium.
- Formula:

$$d_{prop} = \frac{d}{s}$$

where

- d = distance (meters),
- s = propagation speed (~2×10^8 m/s in fiber).

Example:

If distance $d = 2000$ km, speed $s = 2 \times 10^8$ m/s:

$$d_{prop} = \frac{2 \times 10^6}{2 \times 10^8} = 0.01 \text{ sec} = 10 \text{ ms}$$

📌 Total Nodal Delay

The **end-to-end delay per node** is:

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Q5. Consider the network shown, where Router A is the source and Router D is the destination; R1, R2, and R3 are the transmission rates in the network, and if the d_{proc} = 30.58ms, d_{queue} 11.05ms, and d_{prop} = 112.69ms and the length of the packet are 200 bytes, then find the transmission delays of A to B, B to C, C to D, and end-to-end delay ($d_{end\text{-}end}$).

Q6. In a packet-switched network, consider the following path from source A to destination G, where packets travel through intermediate routers D, F, and E. The transmission rates between nodes are as follows: A to D: R1 = 2 Mbps, D to F: R2 = 1.5 Mbps, F to E: R3 = 1 Mbps, E to G: R4 = 0.8 Mbps

The network has the following delays for each node:

Processing delay (d_{proc}): 4.12 ms, Queuing delay (d_{queue}): 3.05 ms, Propagation delay (d_{prop}): 5.32 ms. The length of the packet being transmitted is 100 bits.

a) Calculate the transmission delay for each segment: A to D, D to F, F to E, E to G

b) Find the total end-to-end delay ($d_{end\text{-}to\text{-}end}$) from A to G, considering the transmission delay, processing delay, queuing delay, and propagation delay for each link.

Q5.

Given,

$d_{proc} = 30.58ms$, $d_{queve} = 11.05ms$, $d_{prop} = 112.69ms$

Length of packet, $L = 200$ bytes. $N = 3$

Transmission delays,

$$d_{trans(A-B)} = \frac{L}{R_1} = \frac{200 \times 8 \text{ bits}}{100 \times 10^6 \text{ bps}} = 1.6 \times 10^{-5} \text{ s}$$

$$d_{trans(B-C)} = \frac{L}{R_2} = \frac{200 \times 8 \text{ bits}}{105 \times 10^6 \text{ bps}} = 1.52 \times 10^{-5} \text{ s}$$

$$d_{trans(C-D)} = \frac{L}{R_3} = \frac{200 \times 8 \text{ bits}}{201 \times 10^6 \text{ bps}} = 7.96 \times 10^{-6} \text{ s}$$

Total transmission delay, $d_{trans} = 1.6 \times 10^{-5} + 1.52 \times 10^{-5} + 7.96 \times 10^{-6}$

$= 3.916 \times 10^{-5}$ s

$$d_{(end\text{-}to\text{-}end)} = N \times (d_{proc} + d_{queve} + d_{prop}) + d_{trans}$$

$$= 3 \times (30.58 \times 10^{-3} + 11.05 \times 10^{-3} + 112.69 \times 10^{-3}) + 3.916 \times 10^{-5}$$

$$= 0.4629 \text{ s}$$

Q6. In a packet-switched network, consider the following path from source A to destination G, where packets travel through intermediate routers D, F, and E. The transmission rates between nodes are as follows: A to D: R1 = 2 Mbps, D to F: R2 = 1.5 Mbps, F to E: R3 = 1 Mbps, E to G: R4 = 0.8 Mbps

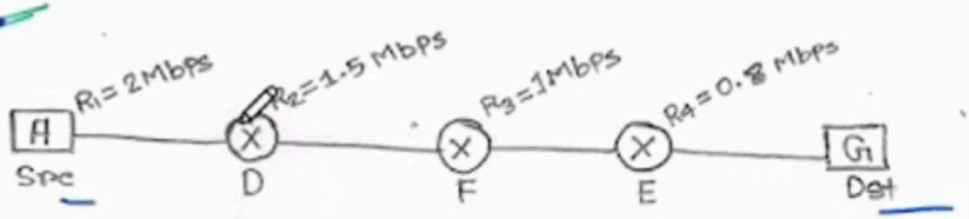
The network has the following delays for each node:

Processing delay (d_{proc}): 4.12 ms, Queuing delay (d_{queue}): 3.05 ms, Propagation delay (d_{prop}): 5.32 ms. The length of the packet being transmitted is 100 bits.

a) Calculate the transmission delay for each segment: A to D, D to F, F to E, E to G

b) Find the total end-to-end delay ($d_{end\text{-}to\text{-}end}$) from A to G, considering the transmission delay, processing delay, queuing delay, and propagation delay for each link.

Q6.



Given,

d_{proc} = 4.12ms, d_{queue} = 3.05ms, d_{prop} = 5.32ms

Length of packet, L = 100 bits. N = 4

a) Transmission delays,

a) Transmission delays,

$$d_{trans}(A-D) = \frac{L}{R_1} = \frac{100 \text{ bits}}{2 \times 10^6 \text{ bps}} = 5 \times 10^{-5} \text{ s}$$

$$d_{trans}(D-F) = \frac{L}{R_2} = \frac{100 \text{ bits}}{1.5 \times 10^6 \text{ bps}} = 6.666 \times 10^{-5} \text{ s}$$

$$d_{trans}(F-E) = \frac{L}{R_3} = \frac{100 \text{ bits}}{1 \times 10^6 \text{ bps}} = 1 \times 10^{-4} \text{ s}$$

$$d_{trans}(E-G) = \frac{L}{R_4} = \frac{100 \text{ bits}}{0.8 \times 10^6 \text{ bps}} = 1.25 \times 10^{-4} \text{ s}$$

b)

Total transmission delay,

$$d_{trans} = (d_{trans(A-D)} + d_{trans(D-F)} + d_{trans(F-E)} + d_{trans(E-G)})$$

$$= (5 \times 10^{-5} + 6.666 \times 10^{-5} + 1 \times 10^{-4} + 1.25 \times 10^{-4}) \text{ s}$$

$$= 3.416 \times 10^{-4} \text{ s}$$

$$\therefore d_{(end-to-end)} = N \times (d_{proc} + d_{queue} + d_{prop}) + d_{trans}$$

$$= 4 \times (4.12 \times 10^{-3} + 3.05 \times 10^{-3} + 5.32 \times 10^{-3}) + 3.416 \times 10^{-4}$$

$$= 0.050 \text{ s}$$

Throughput

1. **Definition**: Rate at which data is received (bits/sec).
    - Average = `F/T` (file size ÷ transfer time).
    - Instantaneous = at a specific time.
2. **Importance**:
    - Real-time apps → need steady throughput above threshold.
    - File transfers → high throughput preferred, delay less critical.
3. **Two-link path**: Throughput = **min{Rs, Rc}** (bottleneck link).
4. **N-link path**: Throughput = **min{R1, R2, ..., RN}**.
5. **Today's Internet**: Core is very fast → bottleneck usually at **access link**.
6. **Shared link case**: If multiple flows share one link, throughput = link rate ÷ number of flows

---

Example

- 10 cars (packets),
- Each car = 1000 bits,
- Toll booth rate $R = 1000$ bps,
- Road distance = 100 km,
- Propagation speed $s = 100$ km/sec.

👉 Transmission delay per car:

$$d_{trans} = \frac{1000}{1000} = 1 \text{ sec}$$

👉 Time for 10 cars:

$$T = 10 \times 1 = 10 \text{ sec}$$

👉 Propagation delay:

$$d_{prop} = \frac{100}{100} = 1 \text{ sec}$$

👉 So first car arrives after:

$$1(transmission) + 1(propagation) = 2 \text{ sec}$$

👉 Last car arrives after:

$$10(alltransmissions) + 1(propagation) = 11 \text{ sec}$$

Takeaway

- **Throughput** = how many packets per second get through.
- Bottleneck = the **slowest toll booth (link)** in the path.
- This shows that **link capacity limits throughput, not propagation speed**.

Latency / Delay

Definition:

- Total time for a packet to travel from source to destination.

Main Components:

1. **Processing Delay** – Time routers take to examine the packet header and decide where to forward it. (Usually microseconds).
2. **Queuing Delay** – Time packet waits in the router's queue before being transmitted (depends on congestion; can range from negligible to seconds).
3. **Transmission Delay** – Time to push all bits of the packet onto the link.
    - Formula: `Packet size (L) / Link bandwidth (R)`
4. **Propagation Delay** – Time for signal to travel through the medium.
    - Formula: `Distance (d) / Propagation speed (s)`

Total Nodal Delay:

`d_total = d_proc + d_queue + d_trans + d_prop`

Encapsulation & Decapsulation

1. Encapsulation (Sender Side 🚀)

Encapsulation = the process of **wrapping data with protocol information** at each layer of the OSI/TCP-IP model before sending it across the network.

1. **Application Layer** → Data (message)
2. **Transport Layer** → Adds **TCP/UDP header** (Port numbers) → Segment
3. **Network Layer** → Adds **IP header** (Source/Destination IP) → Packet
4. **Data Link Layer** → Adds **MAC header + trailer** → Frame
5. **Physical Layer** → Converts to bits → 0s and 1s transmitted

⚡ End product: **Bits on the wire**

---

2. Decapsulation (Receiver Side 🎯)

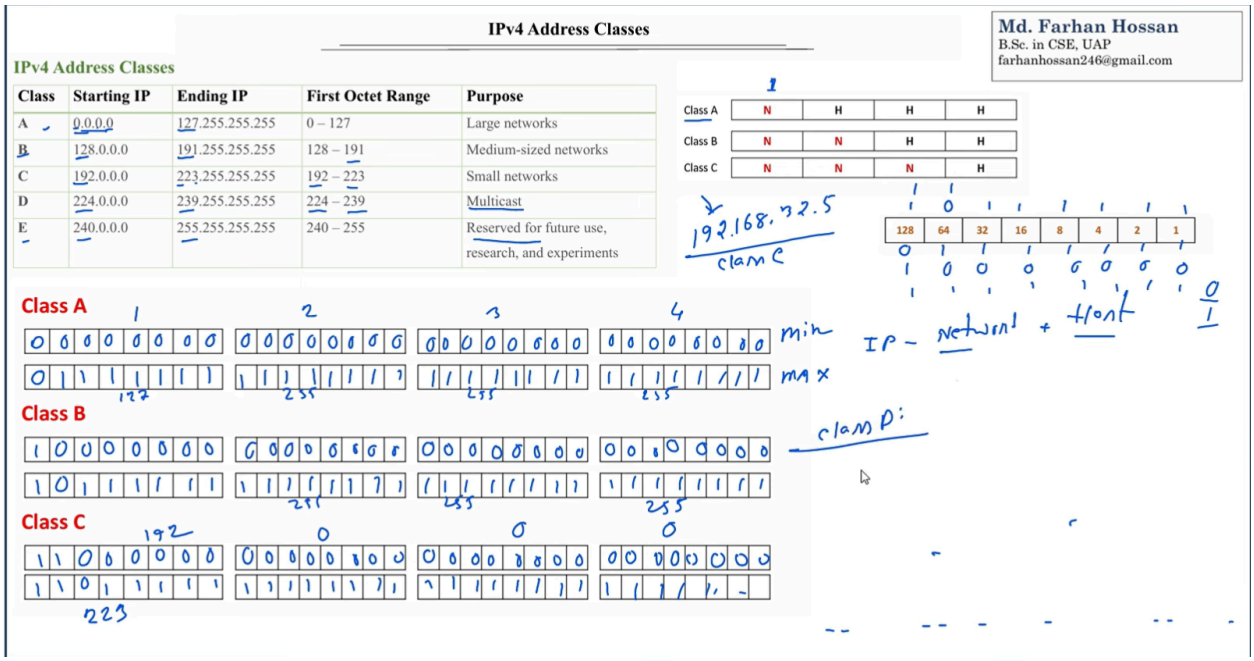
Decapsulation = the **reverse process** at the receiver end → each layer **unwraps and reads only its own header**, then passes the remaining data up.

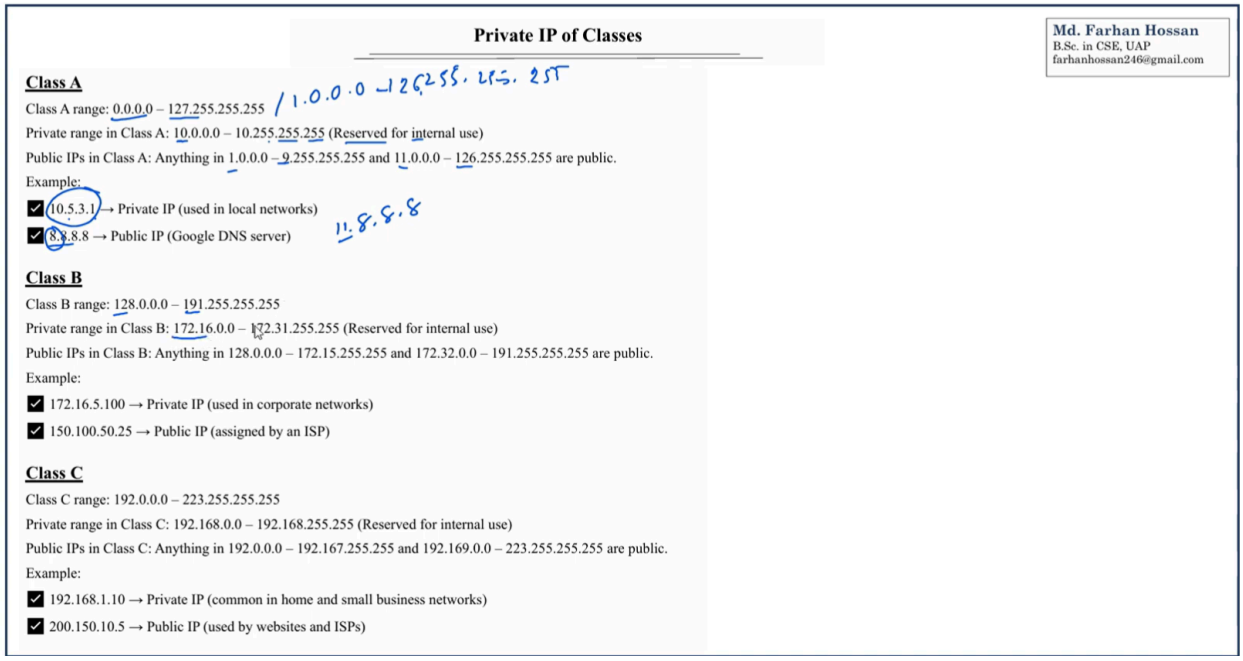
Step-by-step (bottom → up):

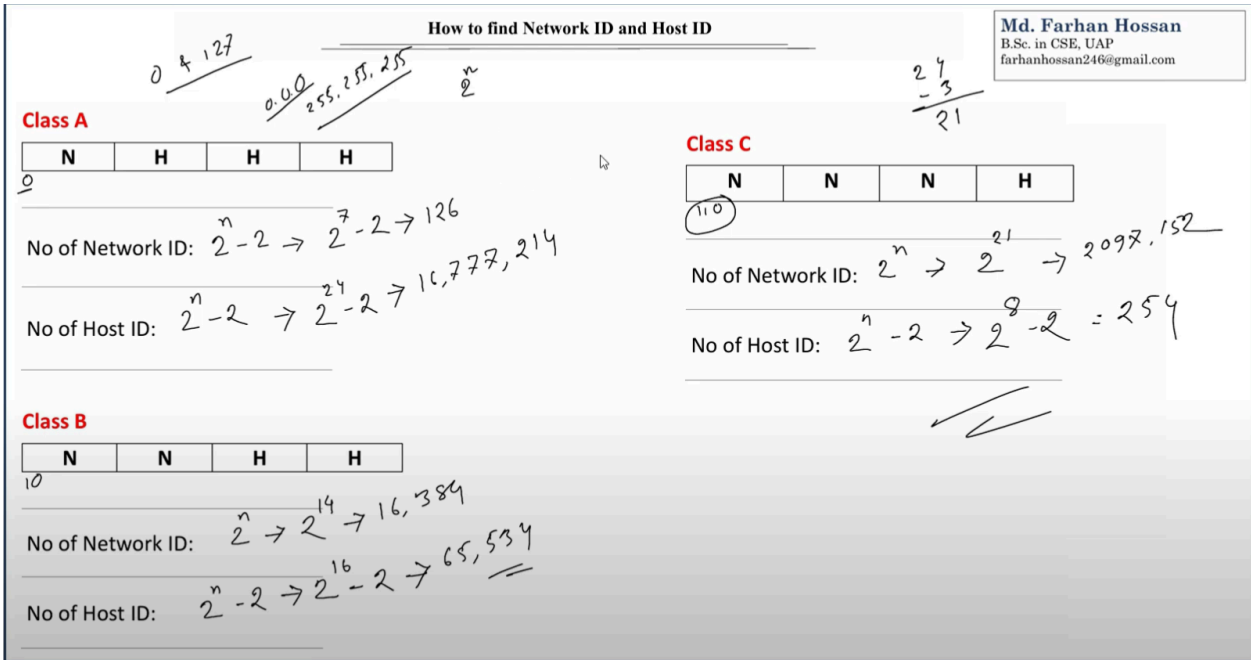
1. **Physical Layer** → Receives bits
2. **Data Link Layer** → Removes **MAC header/trailer**, passes up a packet
3. **Network Layer** → Removes **IP header**, passes up a segment
4. **Transport Layer** → Removes **TCP/UDP header**, gives raw data
5. **Application Layer** → Final message is read

⚡ End product: **Original message delivered**

Ip Addreess




Private IP of Classes
Md. Farhan Hossan
B.Sc. in CSE, UAP
farhanhossan246@gmail.com
Class A
Class A range: 0.0.0.0 – 127.255.255.255
Private range in Class A: 10.0.0.0 – 10.255.255.255 (Reserved for internal use)
Public IPs in Class A: Anything in 1.0.0.0 – 9.255.255.255 and 11.0.0.0 – 126.255.255.255 are public.
Example:
10.5.3.1 → Private IP (used in local networks)
8.8.8.8 → Public IP (Google DNS server)
Class B
Class B range: 128.0.0.0 – 191.255.255.255
Private range in Class B: 172.16.0.0 – 172.31.255.255 (Reserved for internal use)
Public IPs in Class B: Anything in 128.0.0.0 – 172.15.255.255 and 172.32.0.0 – 191.255.255.255 are public.
Example:
172.16.5.100 → Private IP (used in corporate networks)
150.100.50.25 → Public IP (assigned by an ISP)
Class C
Class C range: 192.0.0.0 – 223.255.255.255
Private range in Class C: 192.168.0.0 – 192.168.255.255 (Reserved for internal use)
Public IPs in Class C: Anything in 192.0.0.0 – 192.167.255.255 and 192.169.0.0 – 223.255.255.255 are public.
Example:
192.168.1.10 → Private IP (common in home and small business networks)
200.150.10.5 → Public IP (used by websites and ISPs)


How to find Network ID and Host ID
Md. Farhan Hossan
B.Sc. in CSE, UAP
farhanhossan246@gmail.com
Class A
N H H H
No of Network ID:
No of Host ID:
Class B
N N H H
No of Network ID:
No of Host ID:
Class C
N N N H
No of Network ID:
No of Host ID:

Subnet Mask

Md. Farhan Hossan
B.Sc. in CSE, UAP
farhanhossan246@gmail.com

A subnet mask helps a computer know which part of an IP address is the network and which part is the host.

Class A

N	H	H	H
11111111	0	0	0

255. 0. 0. 0 → Default Subnet mask

N - 1
H - 0

N N N H
192.168.1.10
255. 255. 255. 0

Class B

N	N	H	H
255	255	0	0
11111111	11111111		

255. 255. 0. 0

Class C

N	N	N	H
255	255	255	0

255. 255. 255. 0

Subnet Mask

Md. Farhan Hossan
B.Sc. in CSE, UAP
farhanhossan246@gmail.com

128 64 32 16 8 4 2 1
1 0 0 0 0 0 0 0

N N N H (255.255.255.192)

192.168.1.129/26

1. What will be the subnet mask?
2. Find out number of subnets? → 4
3. Find out number of host? → 62
4. What is the block size? → 64
5. What is the network address?
6. What is broadcast address?
7. What is first valid host?
8. What is the last valid host?

Number of Subnets = 2^(Borrowed Bits)
Number of host = 2^host bits - 2
Block Size = 256 - (Subnet Mask Value)
Network ID = (IP from customized Octet ÷ Block Size) × Block Size
Broadcast = Network ID + (Block Size - 1)
First Host = Network ID + 1
Last Host = Broadcast - 1

① 11111111.11111111.11111111.11000000
→ 255. 255. 255. 192

② $2^2 \rightarrow 4$

③ no of host → $2^6 - 2$
→ 64 - 2
→ 62

④ 256 - 192
= 64

⑤ N. ID = (129 ÷ 64) × 64
= 2 × 64
= 128
(192.168.1.128)

⑥ B. ID → 192.168.1.128 + 63
→ 192.168.1.191

⑦ F. v. H → 192.168.1.129

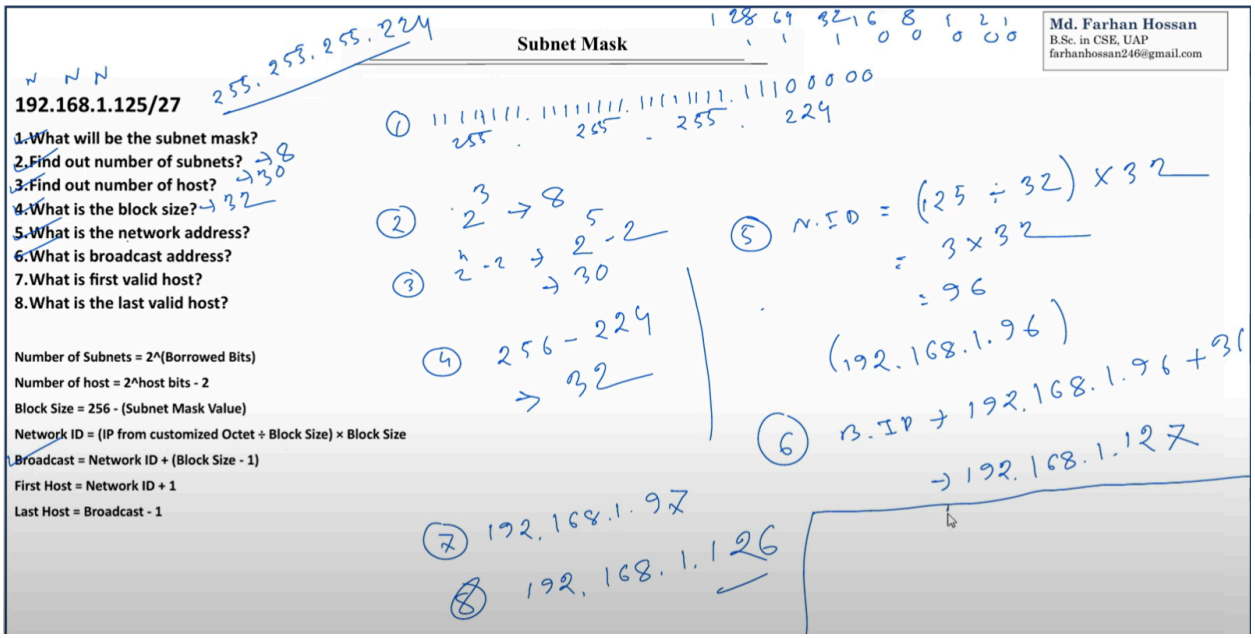
⑧ 192.168.1.190
(B. I)

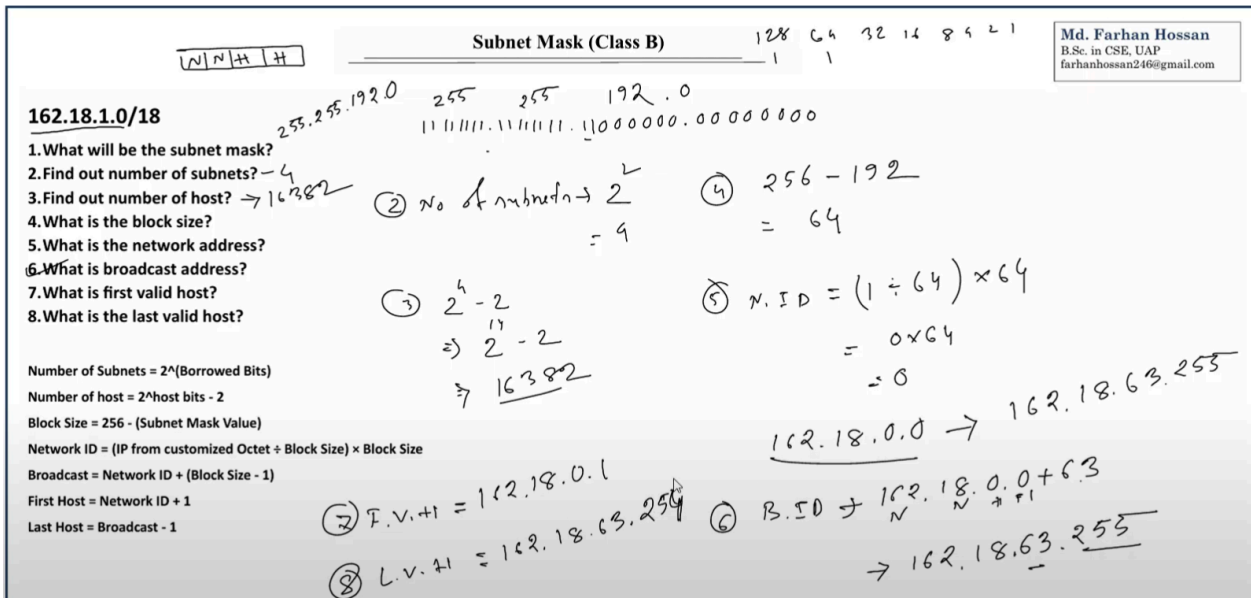
6. For each subnet what will be the network id, first host id, last host id and broadcast id?

192.168.1.129/26

$2^x = 2^4$

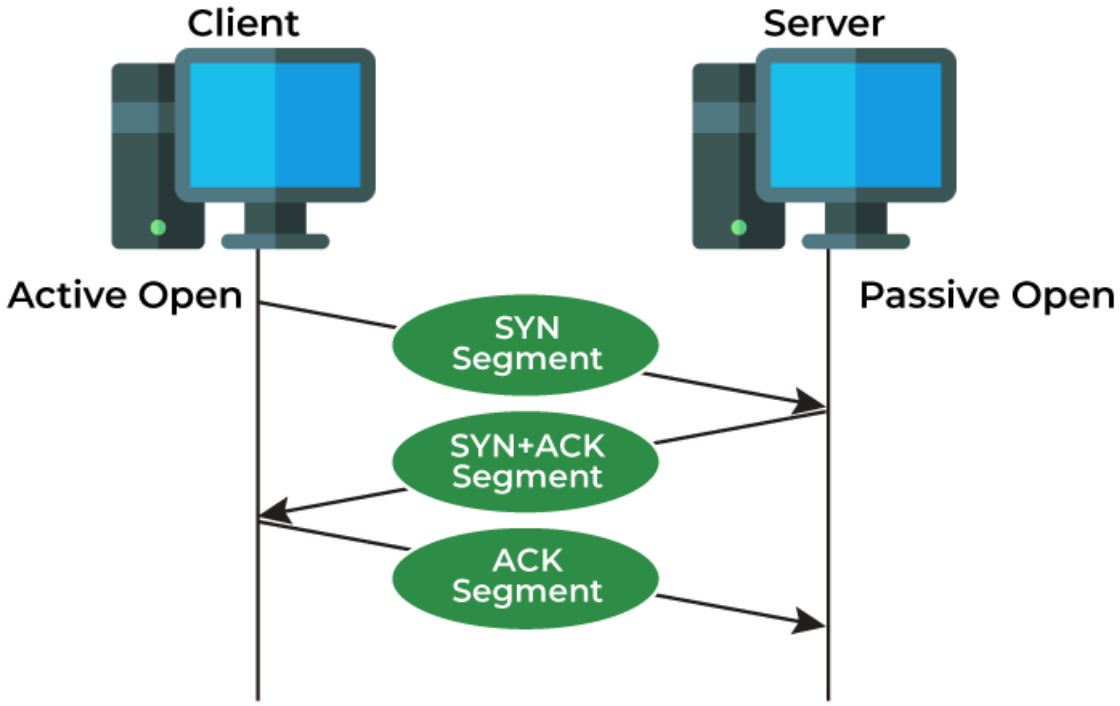
Network id	First host id	Last host id	Broadcast id
192.168.1.0	192.168.1.1	192.168.1.62	192.168.1.63
192.168.1.64	192.168.1.65	192.168.1.126	192.168.1.127
192.168.1.128	192.168.1.129	192.168.1.190	192.168.1.191
192.168.1.192	192.168.1.193	192.168.1.254	192.168.1.255





What is Transmission Control Protocol (TCP)?

TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.

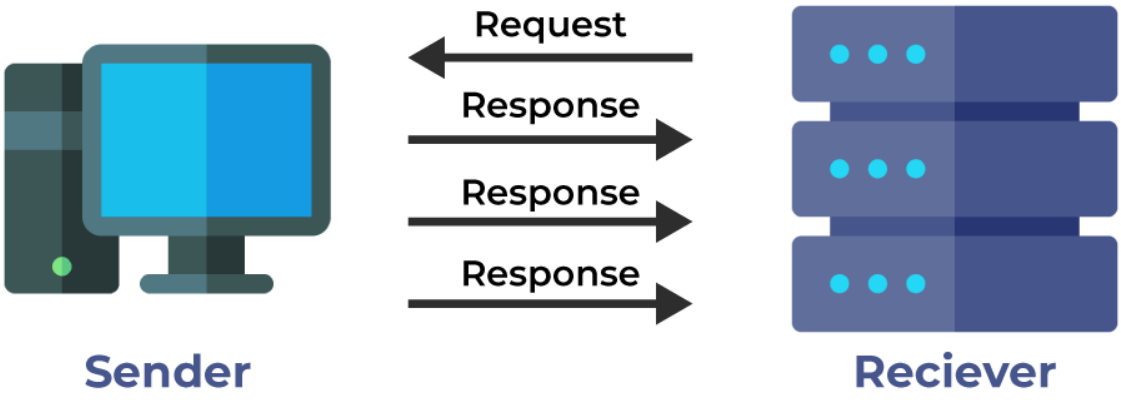


Applications of TCP

- **World Wide Web (WWW)** : When you browse websites, TCP ensures reliable data transfer between your browser and web servers.
- **Email** : TCP is used for sending and receiving emails. Protocols like **SMTP** (Simple Mail Transfer Protocol) handle email delivery across servers.
- **File Transfer Protocol (FTP)** : FTP relies on TCP to transfer large files securely. Whether you're uploading or downloading files, TCP ensures data integrity.
- **Secure Shell (SSH)** : SSH sessions, commonly used for remote administration, rely on TCP for encrypted communication between client and server.
- **Streaming Media** : Services like Netflix, YouTube, and Spotify use TCP to stream videos and music. It ensures smooth playback by managing data segments and retransmissions.

What is User Datagram Protocol (UDP)?

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as the UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection before data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network. The UDP enables process-to-process communication.



Application of UDP

- **Real-Time Multimedia Streaming** : UDP is ideal for streaming audio and video content. Its low-latency nature ensures smooth playback, even if occasional data loss occurs.
- **Online Gaming** : Many online games rely on UDP for fast communication between players.
- **DNS (Domain Name System) Queries** : When your device looks up **domain names** (like converting “www.example.com” to an IP address), UDP handles these requests efficiently

.

- **Network Monitoring** : Tools that monitor network performance often use UDP for lightweight, rapid data exchange.
- **Multicasting** : UDP supports packet switching, making it suitable for multicasting scenarios where data needs to be sent to multiple recipients simultaneously.
- **Routing Update Protocols** : Some routing protocols, like RIP (Routing Information Protocol), utilize UDP for exchanging routing information among routers.

Differences between TCP and UDP

Basis	Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Type of Service	**TCP** is a connection-oriented protocol. Connection orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.	**UDP** is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, or terminating a connection. UDP is efficient for broadcast and multicast types of network transmission.
Reliability	TCP is reliable as it guarantees the delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
Error checking mechanism	TCP provides extensive **error-checking** mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error-checking mechanism using **checksums.**
Acknowledgment	An acknowledgment segment is present.	No acknowledgment segment.
Sequence	Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in order at the receiver.	There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer.
Speed	TCP is comparatively slower than UDP.	UDP is faster, simpler, and more efficient than TCP.
Retransmission	Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in the User Datagram Protocol (UDP).

Basis	Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Header Length	TCP has a (20-60) bytes variable length header.	UDP has an 8 bytes fixed-length header.
Weight	TCP is heavy-weight.	UDP is lightweight.
Handshaking Techniques	Uses handshakes such as SYN, ACK, SYN-ACK	It's a connectionless protocol i.e. No handshake
Broadcasting	TCP doesn't support Broadcasting.	UDP supports Broadcasting.
Protocols	TCP is used by **HTTP,** **HTTPs** , **FTP** , **SMTP** and **Telnet** .	UDP is used by **DNS** , **DHCP** , TFTP, **SNMP** , **RIP** , and **VoIP** .
Stream Type	The TCP connection is a byte stream.	UDP connection is a message stream.
Overhead	Low but higher than UDP.	Very low.
Applications	This protocol is primarily utilized in situations when a safe and trustworthy communication procedure is necessary, such as in email, on the web surfing, and in military services.	This protocol is used in situations where quick communication is necessary but where dependability is not a concern, such as VoIP, game streaming, video, and music streaming, etc.

Domain Name System (DNS)

DNS is a hierarchical and distributed naming system that translates domain names into IP addresses. When you type a domain name like **www.geeksforgeeks.org** into your browser, DNS ensures that the request reaches the correct server by resolving the domain to its corresponding IP address.

Without DNS, we'd have to remember the numerical IP address of every website we want to visit, which is highly impractical.

How Does DNS Work?

The DNS process can be broken down into several steps, ensuring that users can access websites by simply typing a domain name into their browser

Step 01: User requests and local cache is checked

Computer browser requests to visit https://geeksforgeeks.org

Local Cache is checked for requested address

Step 02: Host Files are checked

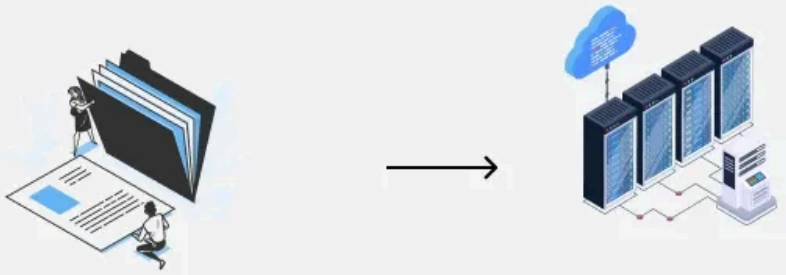
Checking Hosts File

Hosts File

If IP address not found in cache

The hosts file is a system file that maps domain names to IP addresses locally

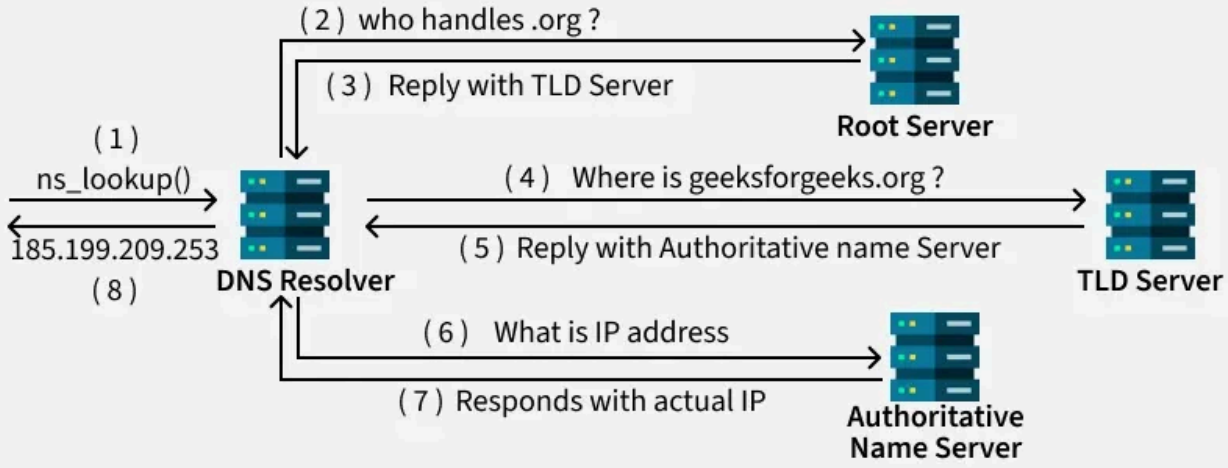
Step 03: Query DNS Resolver

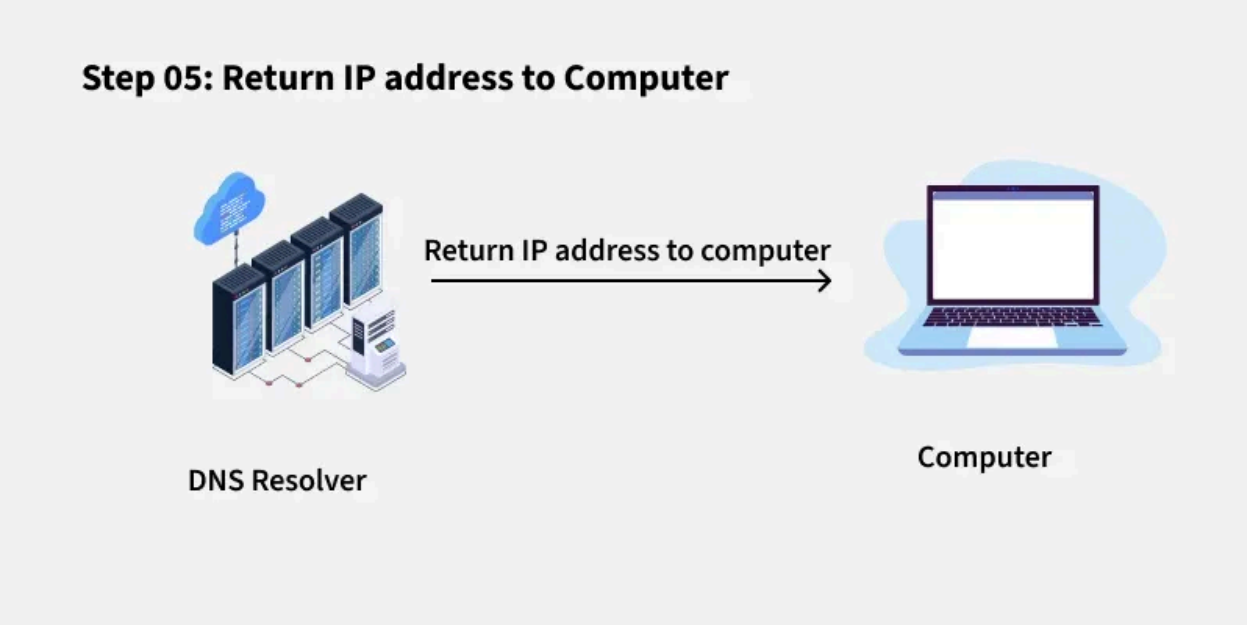


If IP address not found locally

DNS Resolver is a server provided by ISP

Step 04: DNS Search by DNS Resolver





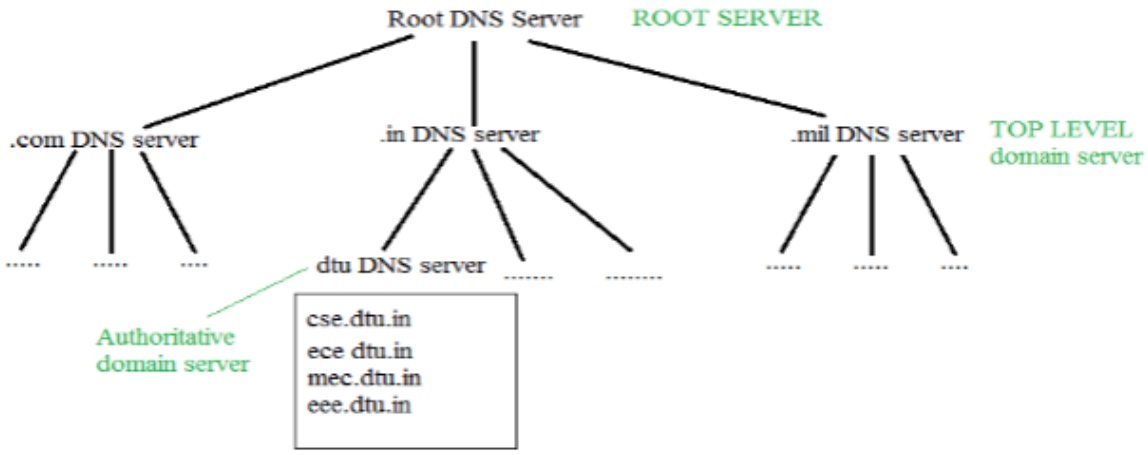
1. **User Input:** You enter a website address (for example, www.geeksforgeeks.org) into your web browser.
2. **Local Cache Check:** Your browser first checks its local cache to see if it has recently looked up the domain. If it finds the corresponding IP address, it uses that directly without querying external servers.
3. **DNS Resolver Query:** If the IP address isn't in the local cache, your computer sends a request to a DNS resolver. The resolver is typically provided by your Internet Service Provider (ISP) or your network settings.
4. **Root DNS Server:** The resolver sends the request to a root **DNS server**. The root server doesn't know the exact IP address for www.geeksforgeeks.org but knows which Top-Level Domain (TLD) server to query based on the domain's extension (e.g., .org).
5. **TLD Server:** The TLD server for .org directs the resolver to the authoritative DNS server for geeksforgeeks.org.
6. **Authoritative DNS Server:** This server holds the actual DNS records for geeksforgeeks.org, including the IP address of the website's server. It sends this IP address back to the resolver.

7. **Final Response:** The DNS resolver sends the IP address to your computer, allowing it to connect to the website's server and load the page.

Structure of DNS

DNS operates through a hierarchical structure, ensuring scalability and reliability across the global internet infrastructure. Here's how it's organized:

- **Root DNS Servers:** These are the highest-level DNS servers and know where to find the TLD servers. They are crucial for directing DNS queries to the correct locations.
- **TLD Servers:** These servers manage domain extensions like .com, .org, .net, .edu, .gov and others. They help route queries to the authoritative DNS servers for specific domains.
- **Authoritative DNS Servers:** These are the servers that store the actual DNS records for domain names. They are responsible for providing the correct IP addresses that allow users to reach websites.



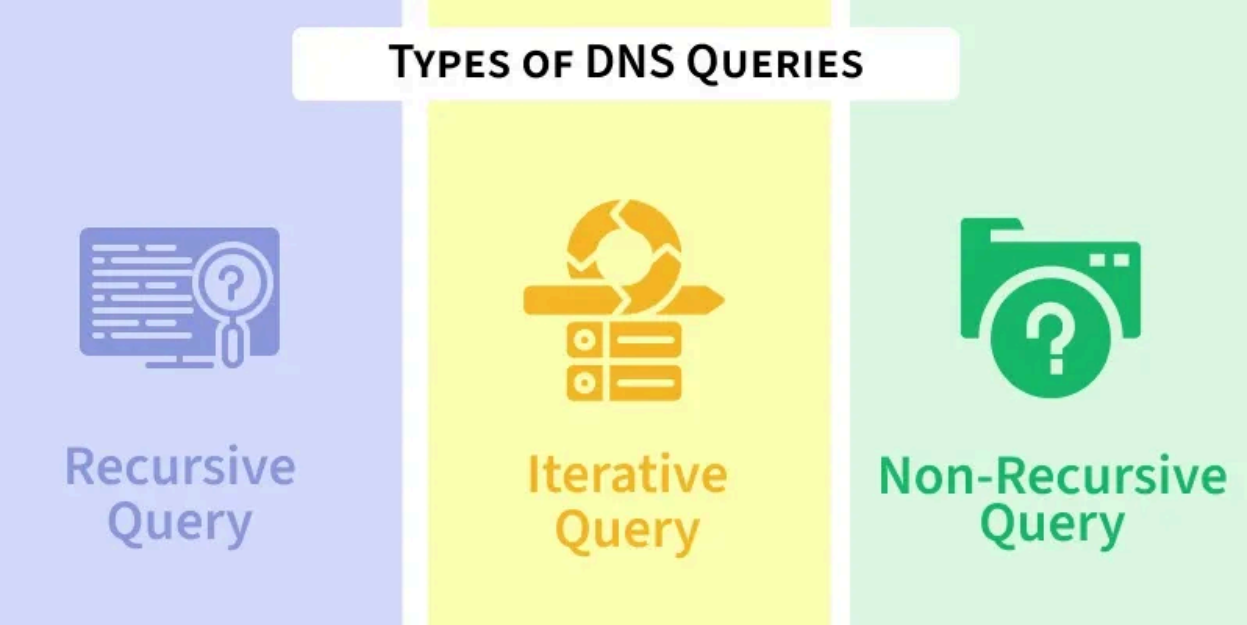
This hierarchical approach allows DNS to handle billions of queries every day, ensuring the stability and scalability of the internet.

DNS Resolver

DNS Resolver is simply called a DNS Client and has the functionality for initiating the process of DNS Lookup which is also called DNS Resolution. By using the DNS Resolver, applications can easily access different websites and services present on the Internet by using domain names that are very much friendly to the user and that also resolves the problem of remembering IP Address.

Types of DNS Queries

There are basically three types of DNS Queries that occur in DNS Lookup. These are stated below.



- **Recursive Query:** In this query, if the resolver is unable to find the record, in that case, DNS client wants the DNS Server will respond to the client in any way like with the requested source record or an error message.
- **Iterative Query:** Iterative Query is the query in which DNS Client wants the best answer possible from the DNS Server.
- **Non-Recursive Query:** Non-Recursive Query is the query that occurs when a DNS Resolver queries a DNS Server for some record that has access to it because of the record that exists in its cache.

DNS Record Types (A, CNAME, MX, TXT, etc.)

DNS records are essential for defining how domain names are used and how services are configured. Here are some of the most commonly used DNS record types:

- **A Record:** This record maps a domain name to an IPv4 address (e.g., geeksforgeeks.org to 185.199.109.153). This is the most common DNS record used to point a domain to its website's IP address.
- **CNAME Record:** The Canonical Name (CNAME) record allows you to alias one domain name to another. For example, **www.geeksforgeeks.org** can be an alias for geeksforgeeks.org.
- **MX Record:** The Mail Exchange (MX) record defines which mail servers are responsible for receiving emails for a domain. This is crucial for setting up email services.
- **TXT Record:** The Text (TXT) record stores text-based information. It is commonly used to verify domain ownership and to implement email security protocols like SPF (Sender Policy Framework) and DKIM (DomainKeys Identified Mail).

1. **A (Address)**
   - Maps hostname → IPv4 address
   - Example:

     ```
     www.flashNetwork.com   A   128.119.12.55
     ```

2. **AAAA (Quad-A)**
   - Same as A record but for **IPv6 addresses**
   - Example:

     ```
     www.flashNetwork.com   AAAA   2001:db8::1
     ```

3. **NS (Name Server)**
   - Specifies **authoritative DNS servers** for a domain

- Example:

```
flashNetwork.com   NS   dns1.flashNetwork.com
```

4. **MX (Mail Exchange)**
   - Defines **mail servers** for the domain (used for email delivery)
   - Example:

```
flashNetwork.com   MX   mail.flashNetwork.com
```

5. **CNAME (Canonical Name / Alias)**
   - Maps an alias name to a “real” hostname
   - Example:

```
www.flashNetwork.com   CNAME   flashNetwork.com
```

6. **TXT (Text)**
   - Arbitrary text, often used for **SPF, DKIM, domain verification**
   - Example:

```
flashNetwork.com   TXT   "v=spf1 include:_spf.google.com ~all"
```

🔷 Exam Shortcuts to Remember

- **A** → IPv4
- **AAAA** → IPv6
- **NS** → DNS authority
- **MX** → Email
- **CNAME** → Alias

- **TXT** → Verification/security

Question Company:

- dns1.flashNetwork.com → IP: 128.119.12.40
- flashNetwork.com → IPs: 128.119.12.55, 128.119.12.56(Also accessible by www.flashNetwork.com)
- mailserver.flashNetwork.com → IP: 128.119.12.60
- Email address format: username@flashBD.com

a) What to provide to the upper-level ?

b) What to put in your company's ?

Answer (a) – What to provide to the upper-level / Registrar

- **Purpose:** Registrar needs to know **who is responsible for your domain** so it can delegate queries properly.

Records to provide:

Type	Name	Value	Reason
NS	flashNetwork.com	dns1.flashNetwork.com	Specifies your authoritative DNS server for the domain
A	dns1.flashNetwork.com	128.119.12.40	IP of the DNS server so recursive resolvers can reach it

Key Point: Without the A record for your DNS, recursive resolvers wouldn't know how to contact your server.

Answer (b) – Company's DNS Zone Records

Purpose: The company's DNS zone contains **all records necessary to resolve hostnames for internal and external users**, including web, mail, and name servers.

Example DNS Records:

Type	Name	Value	Notes
NS	flashNetwork.com	dns1.flashNetwork.com	Primary authoritative DNS server
NS	flashNetwork.com	flashNetwork.com (secondary)	Secondary authoritative DNS server (optional)
A	dns1.flashNetwork.com	128.119.12.40	Maps DNS server hostname to IP
A	flashNetwork.com	128.119.12.55 / 128.119.12.56	Main website / domain
A	www.flashNetwork.com	128.119.12.55 / 128.119.12.56	Alias for main website
A	mailserver.flashNetwork.com	128.119.12.60	Mail server IP
MX	flashBD.com	mailserver.flashNetwork.com	Email routing for your domain

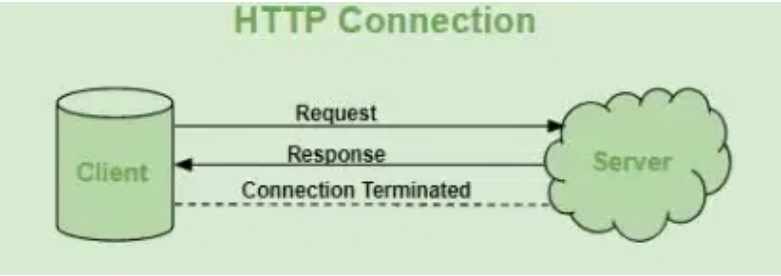
Step-by-Step Explanation

1. **NS records:** Tell the world which servers are authoritative for your domain.
2. **A records:** Map hostnames to their IP addresses.
3. **MX records:** Specify the mail server responsible for handling emails for `flashBD.com`.
4. **Aliases (CNAME, if used):** e.g., `www.flashNetwork.com` can point to `flashNetwork.com`.

What is HTTP ?

HTTP (Hypertext Transfer Protocol) is a fundamental protocol of the Internet, enabling the transfer of data between a client and a server. It is the foundation of data communication for the World Wide Web.

HTTP provides a standard between a web browser and a web server to establish communication. It is a set of rules for transferring data from one computer to another. Data such as text, images, and other multimedia files are shared on the World Wide Web. Whenever a web user opens their web browser, the user indirectly uses HTTP. It is an application protocol that is used for distributed, collaborative, hypermedia information systems.



Methods of HTTP

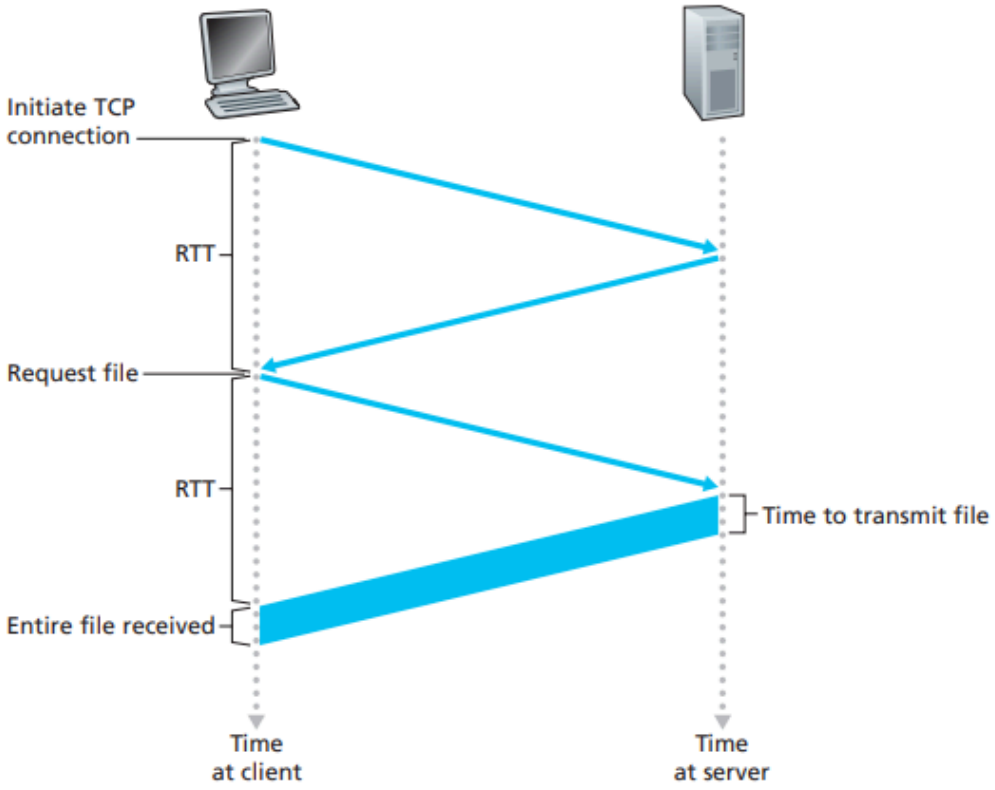
- **GET**: Used to retrieve data from a specified resource. It should have no side effects and is commonly used for fetching web pages, images, etc.
- **POST**: Used to submit data to be processed by a specified resource. It is suitable for form submissions, file uploads, and creating new resources.
- **PUT**: Used to update or create a resource on the server. It replaces the entire resource with the data provided in the request body.
- **PATCH**: Similar to PUT but used for partial modifications to a resource. It updates specific fields of a resource rather than replacing the entire resource.
- **DELETE**: Used to remove a specified resource from the server.
- **HEAD**: Similar to GET but retrieves only the response headers, useful for checking resource properties without transferring the full content.
- **OPTIONS**: Used to retrieve the communication options available for a resource, including supported methods and headers.

Difference between Persistent and Non-Persistent Connections:

Non-Persistent HTTP Connection

A connection where **each object (HTML, image, CSS, etc.) is sent over a separate TCP connection**.

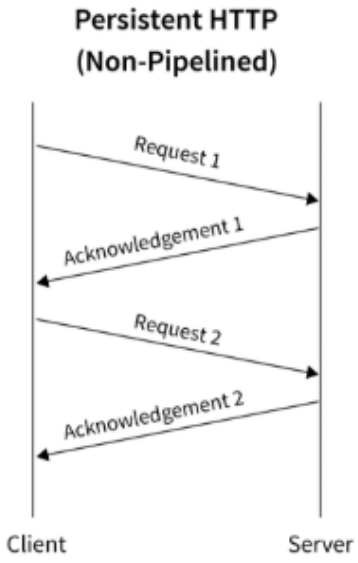
- One TCP connection = **only one request + one response**.
- After response, the connection is closed.



Persistent HTTP Connection

A connection where **the same TCP connection is kept open** and used to send **multiple request–response pairs** between client and server.

- A single connection can carry an entire webpage (HTML + images + scripts).
- Supports **pipelining** (multiple requests sent without waiting for earlier responses).



Feature	Non-Persistent HTTP	Persistent HTTP
Connection Type	New TCP connection **for each request/response**	Single TCP connection **used for multiple requests/responses**
Overhead	High → multiple TCP handshakes	Low → single handshake for multiple requests
Performance	Slower for pages with many objects	Faster → loads multiple objects efficiently
Connection Lifetime	Short-lived → closes after each response	Long-lived → remains open until timeout or all requests are done
Use Case	Rarely used today	Standard in modern HTTP/1.1+

HTTP Request/Response:

HTTP is a request-response protocol, which means that for every request sent by a client (typically a web browser), the server responds with a corresponding response. The basic flow of an HTTP request-response cycle is as follows:

- **Client sends an HTTP request**: The client (usually a web browser) initiates the process by sending an HTTP request to the server. This request includes a request method (GET, POST, PUT, DELETE, etc.), the target URI (Uniform Resource Identifier, e.g., a URL), headers, and an optional request body.

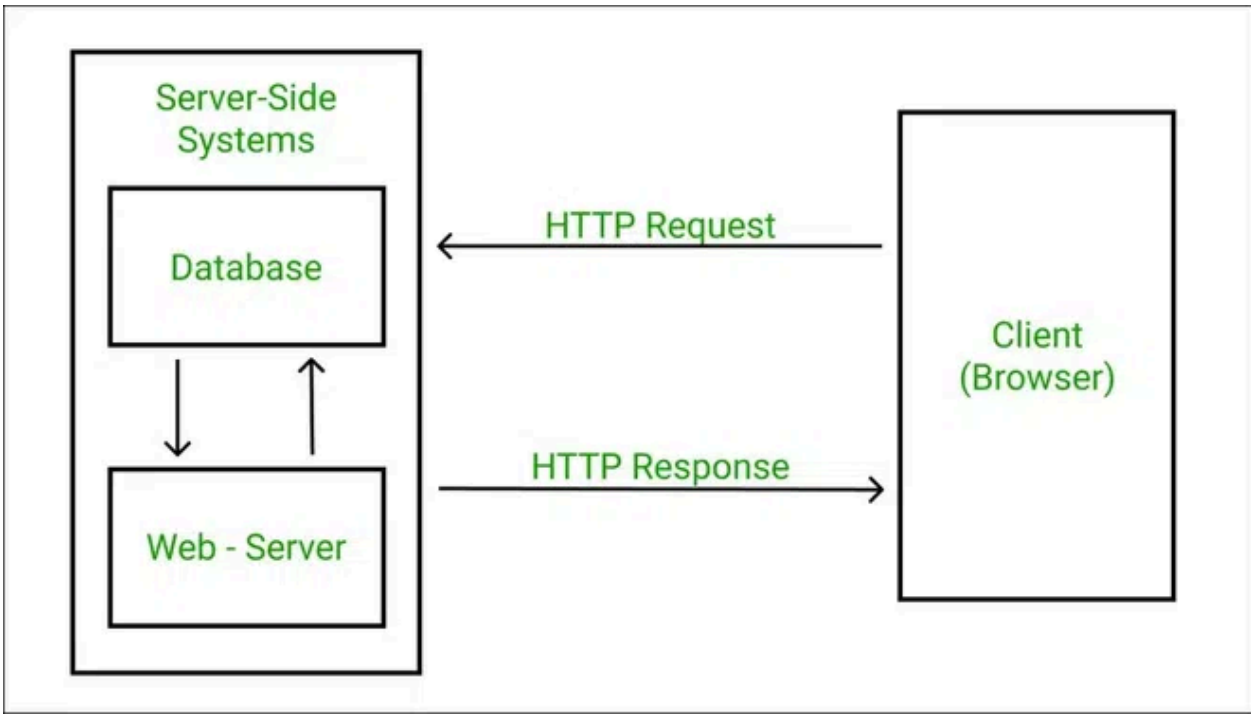
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

- **Server processes the request**: The server receives the request and processes it based on the requested method and resource. This may involve retrieving data from a database, executing server-side scripts, or performing other operations.

- **Server sends an HTTP response:** After processing the request, the server sends an HTTP response back to the client. The response includes a status code (e.g., 200 OK, 404 Not Found), response headers, and an optional response body containing the requested data or content.

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data ...)
```

- **Client processes the response**: The client receives the server's response and processes it accordingly. For example, if the response contains an HTML page, the browser will render and display it. If it's an image or other media file, the browser will display or handle it appropriately.



Active HTTP Response Codes

Code	Meaning	Short Explanation
200	OK	Request succeeded
201	Created	Resource created successfully
301	Moved Permanently	URL permanently moved
302	Found / Temporary Redirect	URL temporarily moved
400	Bad Request	Client sent wrong request
401	Unauthorized	Login/auth required
403	Forbidden	Server refuses access
404	Not Found	Resource doesn’t exist
500	Internal Server Error	Server failed to process request
503	Service Unavailable	Server temporarily down / overloaded

Cookies (Working Principle)

Definition

A **cookie** is a small piece of data that a web server stores on a client's browser to **remember information about the user** across requests.

How Cookies Work (Step-by-Step)

1. **Server Sends a Cookie**
   - When you visit a website, the server responds with:

     ```
     Set-Cookie: sessionId=abc123; Expires=Sat, 14 Sep 2025 23:59:59 GMT
     ```

   - Browser stores this cookie locally.
2. **Browser Sends Cookie Back**
   - On subsequent requests to the same site, the browser automatically sends:

     ```
     Cookie: sessionId=abc123
     ```

   - Server identifies the user session using this cookie.
3. **Cookie Usage**
   - Sessions (login/authentication)
   - Personalization (themes, language preferences)
   - Tracking & analytics

Cache

Server-Side

Definition

- **Server-side caching** stores **recently generated content or data** on the server, so that **repeated requests** don't require expensive processing (like querying a database).
- Improves **performance, reduces server load, and speeds up response time**.

How It Works (Step-by-Step)

1. **Client Request**
   - User requests a resource (e.g., `GET /products`).
2. **Cache Check**
   - Server first checks if the requested data is in the **server-side cache** (memory or disk).
3. **Cache Hit / Miss**
   - **Cache Hit:** Return cached response → no database query.
   - **Cache Miss:** Query database → generate response → store it in cache → send to client.
4. **Cache Expiry**
   - Cached data has a **Time-To-Live (TTL)** or is invalidated when data changes.

Example

- Request: `/user/123/profile`
- Server-side cache stores: `{user123_profile: {...}}`
- Next request for `/user/123/profile` → served from cache **without hitting database**

Benefits

- Reduces **database load**
- Faster **response time** for repeated requests
- Efficient for **high-traffic websites**

Exam Tip

- Always mention **cache hit / cache miss**

- Can combine with **HTTP caching headers** (like `Cache-Control`) for extra credit
- Example answer line:

> “If the server has the requested data in cache (cache hit), it returns it directly; otherwise (cache miss), it queries the database and stores the result in cache.”

Email Protocols

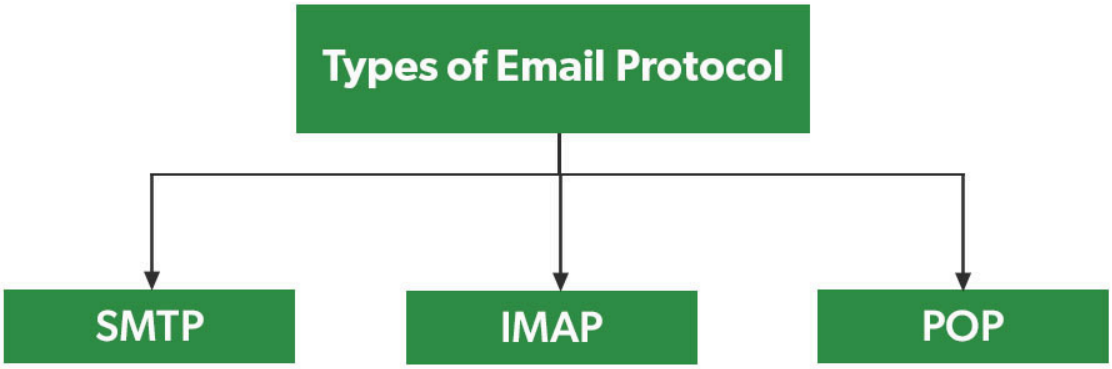
Email protocols are a collection of protocols that are used to send and receive emails properly. The email protocols provide the ability for the client to transmit the mail to or from the intended mail server.

Email protocols are a set of commands for sharing mails between two computers. Email protocols establish communication between the sender and receiver for the transmission of email. Email forwarding includes components like two computers sending and receiving emails and the mail server. There are three basic types of email protocols.

Types of Email Protocols:

Three basic types of email protocols involved for sending and receiving mails are:

- SMTP
- POP3
- IMAP



SMTP (Simple Mail Transfer Protocol):

Simple Mail Transfer Protocol is used to send mails over the internet. SMTP is an application layer and connection-oriented protocol.

SMTP is efficient and reliable for sending emails. SMTP uses TCP as the transport layer protocol. It handles the sending and receiving of messages between email servers over a TCP/IP network. This protocol along with sending emails also provides the feature of notification for incoming mails.

When a sender sends an email then the sender's mail client sends it to the sender's mail server and then it is sent to the receiver mail server through SMTP. SMTP commands are used to identify the sender and receiver email addresses along with the message to be sent.

Some of the SMTP commands are HELLO, MAIL FROM, RCPT TO, DATA, QUIT, VERIFY, SIZE, etc. SMTP sends an error message if the mail is not delivered to the receiver hence, reliable protocol.

POP(Post Office Protocol):

Post Office Protocol is used to retrieve email for a single client. POP3 version is the current version of POP used. It is an application layer protocol. It allows to access mail offline and thus, needs less internet time. To access the message it has to be downloaded. POP allows only a single mailbox to be created on the mail server. POP does not allow search facilities

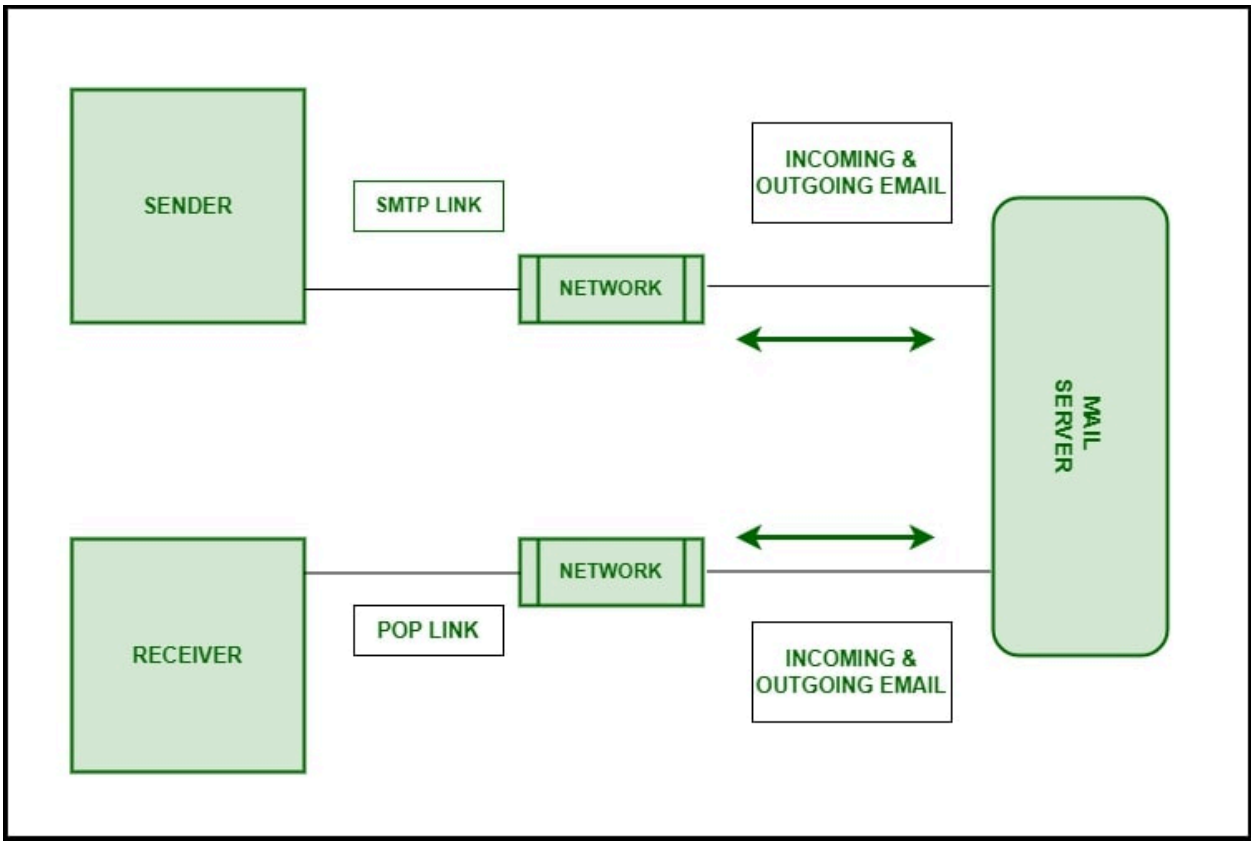
Some of the POP commands are LOG IN, STAT, LIST, RETR, DELE, RSET, and QUIT. For more details please refer to the **POP Full-Form** article.

IMAP(Internet Message Access Protocol):

Internet Message Access Protocol is used to retrieve mails for multiple clients. There are several IMAP versions: IMAP, IMAP2, IMAP3, IMAP4, etc.

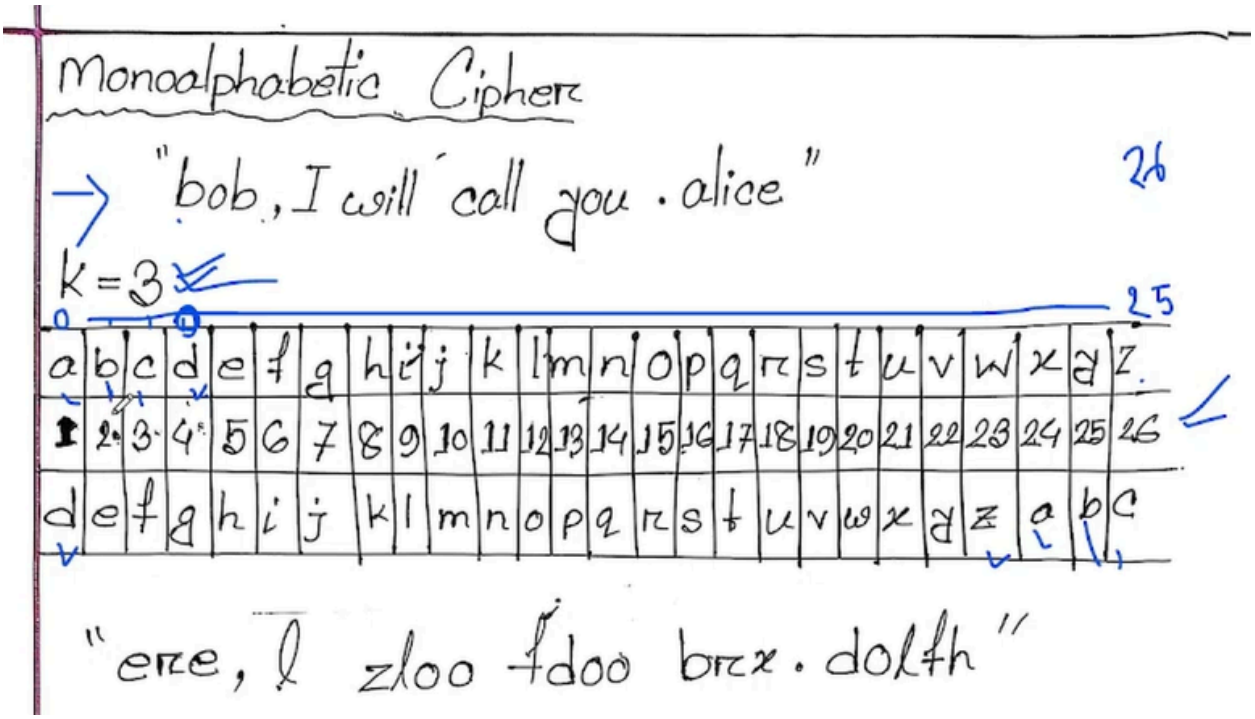
IMAP is an application layer protocol. IMAP allows to access email without downloading them and also supports email download. The emails are maintained by the remote server. It enables all email operations such as creating, manipulating, delete the email without reading it.

IMAP allows you to search emails. It allows multiple mailboxes to be created on multiple mail servers and allows concurrent access. Some of the IMAP commands are: IMAP_LOGIN, CREATE, DELETE, RENAME, SELECT, EXAMINE, and LOGOUT.



Network Security

monoalphbetic cipher



Polyalhpabetic

Polyalphabetic :

$C_1C_2C_3C_3$

$C_1 = k=3$

$C_2 = k=4$

$C_3 = k=12$

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
k=3	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
k=4	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
k=12	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l

"bob, I will call you. alice"

$C_1C_2C_3\ C_3\ C_1C_2C_3C_3\ C_1C_2C_3C_3\ C_1C_2C_3\ C_3C_1C_2C_3C_3$

"ese,

"esn, u zmxx texx. bsg. momoq"