Preface

Over the past year working as a Cloud Engineer, I have gained valuable insights into computer networking and its real-world applications. Throughout my journey, I've gathered and compiled these notes to serve as a resource for those preparing for technical roles such as Software Development Engineer (SDE), DevOps, or any other technical position.

These notes are a collection of publicly available information, and I make no claims of ownership over the material. My goal is simply to provide a consolidated guide to help others in their preparation journey.

I hope these notes prove helpful in your learning journey and bring you one step closer to your career goals.

Special thanks to ChatGPT, my tireless robot assistant, for handling all the formatting and rewriting while I focused on being a human! 🤖✍️

Thank you !

Wishing you success,

Devansh Jain

What is Computer Network ?

A computer network is a collection of two or more computer systems that are linked together and allows them to share data. A network connection can be established using either cable or wireless media.

Types of network :

1. Local Area Network (LAN):

- **Definition:** A LAN is a network that connects devices within a limited geographic area, like a home, school, or office.
- **Example:** Office network connecting computers, printers, and servers.

2. Wide Area Network (WAN):

- **Definition:** A WAN spans a large geographical area, connecting multiple LANs across cities, countries, or continents.
- **Example:** The internet or a company's network connecting offices in different cities.

3. Metropolitan Area Network (MAN):

- **Definition:** A MAN is larger than a LAN but smaller than a WAN, typically covering a city or a large campus.
- **Example:** City-wide Wi-Fi or university campus networks.

4. Personal Area Network (PAN):

- **Definition:** A PAN is a small network for personal devices within close proximity, such as within a few meters.
- **Example:** Bluetooth connections between a smartphone and a headset.

What is Virtual Private Network (VPN) ?

A **VPN (Virtual Private Network)** is a technology that creates a secure and encrypted connection over a less secure network, such as the internet. It allows users to send and receive data across shared or public networks as if their devices were directly connected to a private network, enhancing privacy and security.

How VPN Works:

- **Encryption**: VPNs encrypt the data sent between the user and the VPN server, ensuring that any data intercepted by malicious entities cannot be easily understood.
- **Tunneling**: VPNs use a technique called tunneling, where they encapsulate and transport data securely between networks.
- **IP Address Masking**: VPNs also hide the user's IP address by replacing it with the VPN server's IP address, protecting the user's online identity.

Types of VPNs:

- **Remote Access VPN**:
  - **Purpose**: Allows individual users to connect to a private network remotely over the internet.
  - **Usage**: Commonly used by remote employees or individuals who want to securely connect to a corporate network.
  - **Example**: A worker connecting to their company's internal network from home.
- **Site-to-Site VPN**:
  - **Purpose**: Connects two or more separate networks (such as different office locations) over the internet.
  - **Usage**: Typically used by large businesses with multiple branch offices.
  - **Example**: Connecting an organization's offices in different cities to a centralized company network.
  - **Types**:
    - **Intranet VPN**: Connects different locations within a single organization.
    - **Extranet VPN**: Connects an organization to external business partners or suppliers.

Internet vs World wide web (www) :

The internet is a **global network** of interconnected computers and devices that can communicate with each other while *the Web*, also referred to formally as World Wide Web (www) is a collection of information

that is accessed via *the Internet*. Another way to look at this difference is that *the Internet* is infrastructure while *the Web* is served on top of that infrastructure.

Network Topologies:

Network topology refers to the arrangement of network devices and how data flows between them. Below are the key types of network topologies, along with their pros and cons.

1. Bus Topology:

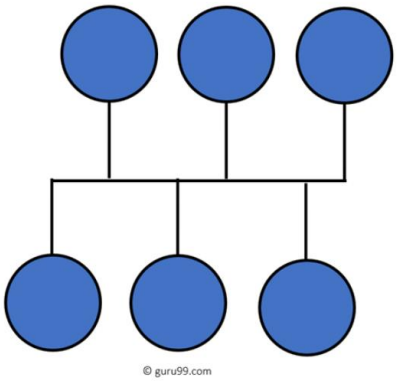
In a bus topology, all devices are connected to a single central cable (the "bus"). Data is sent in both directions along the bus until it reaches the destination.

Pros:

- Easy and inexpensive to set up for small networks.
- Requires less cable compared to other topologies.

Cons:

- Limited scalability; performance degrades as more devices are added.
- A failure in the main cable disrupts the entire network.
- Troubleshooting can be difficult.



2. Star Topology:

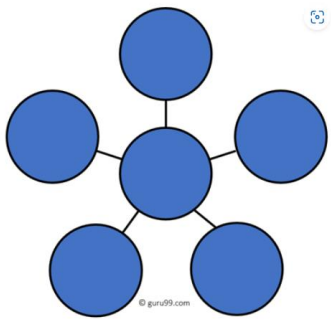
In star topology, all devices are connected to a central hub or switch. The hub manages and directs communication between devices.

Pros:

- Easy to set up and manage.
- Failure of one device does not affect the rest of the network.
- Centralized management simplifies troubleshooting.

Cons:

- The central hub is a single point of failure.
- Requires more cables, increasing cost.
- Performance depends on the capacity of the central hub.



Star Topology Diagram

3. Ring Topology:

In a ring topology, each device is connected to two other devices, forming a circular data path. Data travels in one direction around the ring.

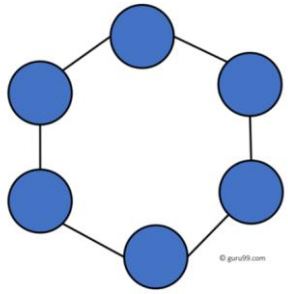
Pros:

- Data flows in an orderly fashion, reducing collisions.
- Easy to identify faults, as each node can report data flow issues.

Cons:

- Failure of one device or cable can disrupt the entire network.

- Adding or removing devices can cause downtime.
- Performance degrades as the number of devices increases.



Ring Topology Diagram

4. Mesh Topology:

In mesh topology, every device is connected to every other device. There are two types:

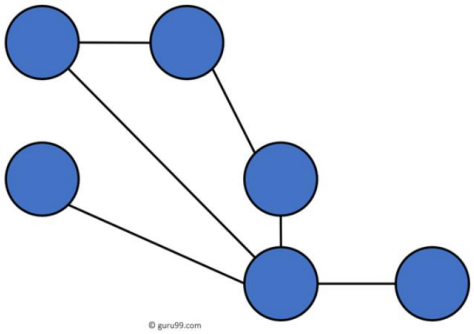
- **Full Mesh**: Every node is directly connected.
- **Partial Mesh**: Some nodes are directly connected.

Pros:

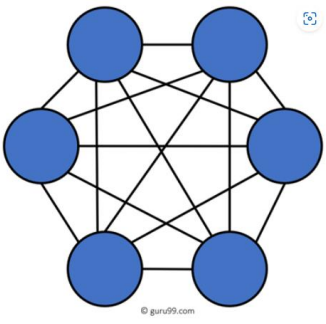
- Provides high redundancy; if one connection fails, data can take another path.
- Highly reliable and secure.

Cons:

- Expensive and complex to set up due to the large number of connections.
- Requires more cables and hardware, increasing maintenance costs.



Partially Connected Mesh Topology



Fully Connected Mesh Topology

5. Tree Topology:

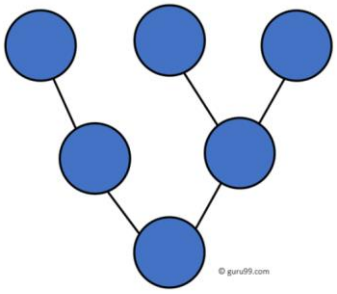
Tree topology combines elements of star and bus topologies. Devices are connected in groups (branches) to a central hub, forming a hierarchical structure.

Pros:

- Scalable; easy to add more nodes or branches.
- Centralized management and hierarchical structure simplify troubleshooting.

Cons:

- Failure of the root hub or backbone cable can disrupt the entire network.
- More expensive and complex due to hierarchical structure.



Tree Topology

6. Hybrid Topology:

Hybrid topology is a combination of two or more topologies, like star-bus or star-ring.

Pros:

- Flexible and scalable; can be customized for specific needs.
- Combines the strengths of different topologies.

Cons:

- More complex to design and maintain.
- Costly due to combining multiple topologies.

Networking Devices:

Networking devices are hardware components used to connect computers, servers, and other devices to form a network, ensuring data communication and transmission. Below are key networking devices, their functions, and typical use cases.

1. Router:

- **Function**: Routes data between different networks and directs traffic efficiently.
- **Usage**: Connects a local network (LAN) to the internet or another network.
- **Key Role**: Determines the best path for data packets using IP addresses.
- **Example**: Home or office router for internet access.

2. Switch:

- **Function**: Connects multiple devices within the same network (LAN) and forwards data to the correct destination.
- **Usage**: Typically used to improve network performance and manage data traffic within a local network.
- **Key Role**: Operates at Layer 2 (Data Link Layer) of the OSI model and uses MAC addresses for data transmission.
- **Example**: A switch connecting computers in an office network.

3. Hub:

- **Function**: Broadcasts data to all devices in a network.
- **Usage**: Rarely used today due to inefficiency and has been largely replaced by switches.
- **Key Role**: Does not filter data, leading to network congestion.
- **Example**: Legacy networks where all devices receive the same data.

4. Modem:

- **Function**: Converts digital data to analog signals for transmission over telephone lines (modulation) and converts analog signals back to digital (demodulation).
- **Usage**: Used to connect home or office networks to the internet via an ISP (Internet Service Provider).

- **Key Role**: Provides the internet connection by translating data between the local network and ISP.
- **Example**: DSL or cable modem for internet connectivity.

5. Access Point (AP):

- **Function**: Provides wireless connectivity (Wi-Fi) to devices within a network.
- **Usage**: Used in Wi-Fi networks to extend wireless coverage and connect mobile devices wirelessly.
- **Key Role**: Acts as a central transmitter and receiver for wireless devices.
- **Example**: Wi-Fi access points in homes or public areas (cafes, offices).

6. Firewall:

- **Function**: Monitors, filters, and controls incoming and outgoing network traffic based on security rules.
- **Usage**: Protects a network from unauthorized access and cyber threats.
- **Key Role**: Acts as a barrier between a trusted internal network and untrusted external networks (like the internet).
- **Example**: Firewall protecting an organization's internal network from malicious attacks.

7. Gateway:

- **Function**: Connects two different networks with different protocols and translates data between them.
- **Usage**: Used to connect a local network to external networks like the internet.
- **Key Role**: Acts as a translator between two different networking systems.
- **Example**: A gateway connecting a company's internal network to a supplier's network.

8. Network Interface Card (NIC):

- **Function**: It is peripheral card that allows a computer or device to connect to a network (wired or wireless).
- **Usage**: Every device on a network requires a NIC to communicate with other devices. Every NIC has its own MAC address that identifies the PC on the network.
- **Key Role**: Provides physical connection to the network.
- **Example**: Ethernet or Wi-Fi card in a laptop or desktop computer.

9. Repeater:

- **Function**: Amplifies or regenerates signals to extend the range of a network.
- **Usage**: Used in large networks to maintain signal strength over long distances.
- **Key Role**: Prevents signal degradation in long-distance communication.
- **Example**: Wi-Fi repeater extending wireless coverage in large buildings.

10. Bridge:

- **Function**: Connects and filters traffic between two or more network segments, operating at the Data Link Layer (Layer 2) of the OSI model.
- **Usage**: Used to divide a large network into smaller segments to reduce network traffic and improve performance.
- **Key Role**: Forwards data based on MAC addresses and prevents unnecessary traffic from crossing network segments.
- **Example**: A bridge connecting two separate LAN segments within a building to reduce collisions and improve communication.

Basic Networking Terminologies :

1. Client

- **Definition**: A client is a device or software that requests services or resources from a server in a network. Clients typically rely on servers for data processing or access.
- **Example**: A web browser (client) requesting a webpage from a web server.

2. Host

- **Definition**: Any device on a network that communicates with other devices. Hosts can both send and receive data.
- **Example**: Computers, smartphones, servers.

3. Server

- **Definition**: A server is a computer, device, or software that provides services, data, or resources to other devices (clients) over a network. Servers can handle requests from multiple clients simultaneously, offering resources such as data storage, websites, applications, or processing power.
- **Example**: When you access a website, your browser (client) sends a request to a web server. The server processes the request and sends the web page back to your browser for display.

4. Peer

- **Definition**: A peer is a device in a peer-to-peer (P2P) network that both provides and consumes resources without relying on a centralized server.
- **Example**: File-sharing networks like BitTorrent.

5. Bandwidth

- **Definition**: The maximum amount of data that can be transmitted over a network in a given amount of time, usually measured in bits per second (bps).
- **Example**: A 100 Mbps connection can transfer 100 megabits of data per second.

6. Jitter

- **Definition**: The variation in packet arrival times, often caused by network congestion or route changes. Jitter can lead to poor audio/video quality in real-time communications.
- **Example**: Voice call quality dropping due to inconsistent data packet delays.

7. Packet

- **Definition**: A packet is a small unit of data transmitted over a network, containing both the payload (data) and control information (like source and destination).
- **Example**: A packet of data being sent from your computer to a server when you access a website.

8. Frame

- **Definition**: A frame is a data packet at the data link layer (Layer 2) of the OSI model. It includes header and trailer information for error checking and flow control.
- **Example**: Ethernet frames used in local area networks (LANs).

9. Local Host

- **Definition**: Refers to the device you are currently using or running network services on. The term often refers to the loopback IP address (`127.0.0.1`).
- **Example**: Running a local web server on `localhost` for development.

10. Noise

- **Definition**: Any unwanted signal or interference that disrupts communication. It can cause data errors or reduce the quality of a transmission.
- **Example**: Electrical interference from devices like microwaves affecting Wi-Fi signals.

11. Attenuation

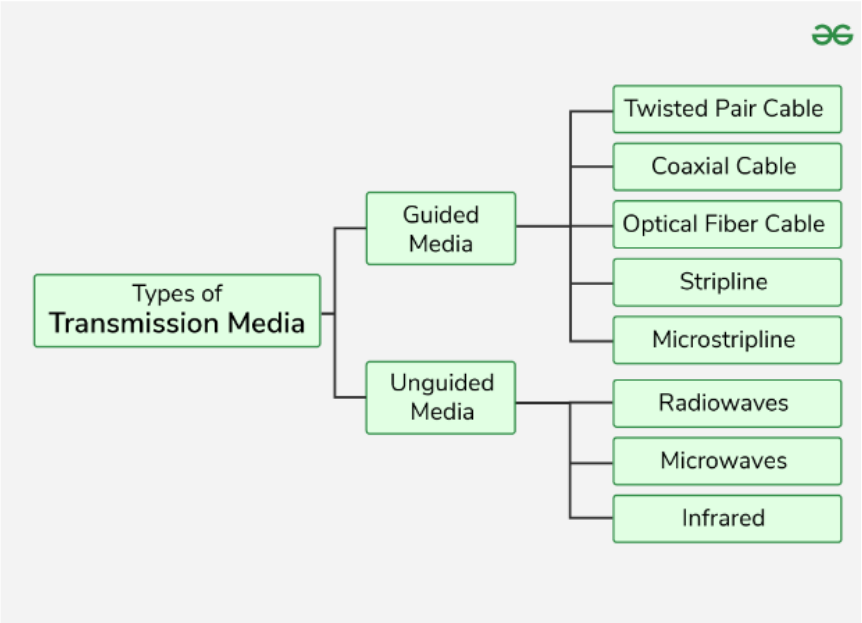
- **Definition**: The weakening of a signal as it travels over a distance, often due to physical properties of the transmission medium.
- **Example**: Signal strength reducing as it travels along a copper wire.

12. Distortion

- **Definition**: The alteration of the original shape or other properties of a signal. It can result from signal attenuation, interference, or other factors.
- **Example**: Distorted audio or video signals due to poor cable quality.

What is Transmission Media?

A transmission medium is a physical path between the transmitter and the receiver, i.e. it is the channel through which data is sent from one place to another. Transmission Media is broadly classified into the following types:



Unicasting vs Anycasting vs Multicasting vs Broadcasting :

1. Unicasting

- **Definition**: Unicasting is the process of sending data from one sender (source) to one specific receiver (destination). It is a **one-to-one** communication method.
- **Usage**: Most internet communications use unicasting, such as web browsing, email, and file downloads.
- **Example**: When you access a website, your computer (client) sends a unicast request to the web server, and the server responds with the web page directly to your device.

2. Anycasting

- **Definition**: Anycasting involves sending data from one sender to **the nearest or best receiver** in a group of potential receivers. It is **one-to-one-of-many** communication.

- **Usage**: Often used in content delivery networks (CDNs) and DNS (Domain Name System) to route user requests to the nearest or most efficient server.
- **Example**: A DNS request may be anycasted to the closest available DNS server, ensuring low latency and faster responses.

3. Multicasting

- **Definition**: Multicasting is the process of sending data from one sender to **multiple specific recipients** who have expressed interest in receiving that data. It is **one-to-many** communication, but only to a defined group.
- **Usage**: Used in applications like video conferencing, live streaming, and IPTV (Internet Protocol Television), where only a select group of devices need the data.
- **Example**: A live sports event streamed to multiple subscribed users using IP multicast.

4. Broadcasting

- **Definition**: Broadcasting involves sending data from one sender to **all devices** in a network segment. It is a **one-to-all** communication method.
- **Usage**: Used in local area networks (LANs) for tasks such as device discovery and network services like ARP (Address Resolution Protocol).
- **Example**: A DHCP request (to assign an IP address) is broadcasted to all devices on the network.

OSI Model

The **OSI (Open Systems Interconnection) model** is a conceptual framework that standardizes the functions of a network system into seven distinct layers. Each layer has specific responsibilities and communicates with the layers directly above and below it.

The 7 Layers of the OSI Model:

1. **Physical Layer (Layer 1)**
    a. **Role**: Deals with the **physical transmission** of data over a network. It involves the hardware aspects of networking like cables, switches, and electrical signals.
    b. **Key Functions**:
        i. Transmission of raw bits over the physical medium.
    c. **Example**: Ethernet cables, fiber optics, and hubs.

2. **Data Link Layer (Layer 2)**
    a. **Role**: Responsible for **node-to-node data transfer** and error detection/correction in frames. It ensures reliable communication between devices on the same local network.
    b. **Key Functions**:
        i. Divides data into **frames** and provides error detection (e.g., CRC).
        ii. Controls access to the physical medium via **MAC addresses**.
    c. **Example**: Ethernet (for LANs), MAC addresses, switches, and frames.

3. **Network Layer (Layer 3)**
    a. **Role**: Handles **routing** of data across different networks and manages logical addressing (**IP addresses**). It determines the best path for data to travel from source to destination.
    b. **Key Functions**:
        i. **Routing** of data packets across multiple networks.
        ii. Uses **IP addresses** for logical addressing.
    c. **Example**: IP (Internet Protocol), routers, and IP addresses.

4. **Transport Layer (Layer 4)**
    a. **Role**: Ensures **end-to-end communication** reliability and data integrity. It breaks down large data chunks into smaller segments and ensures proper sequencing and error correction.
    b. **Key Functions**:
        i. **Segmentation**, reassembly, and sequencing of data.
        ii. Provides **error recovery** and **flow control**.

c. **Example**: TCP (for reliable data transfer), UDP (for faster, less reliable data).

5. **Session Layer (Layer 5)**
    a. **Role**: Manages **sessions** or connections between two devices. It controls dialogues between devices, including establishing, maintaining, and terminating connections.
    b. **Key Functions**:
        i. Establishes, maintains, and closes sessions between applications.
        ii. Ensures proper data synchronization and session recovery.
    c. **Example**: Session establishment in FTP or SMB file transfer.

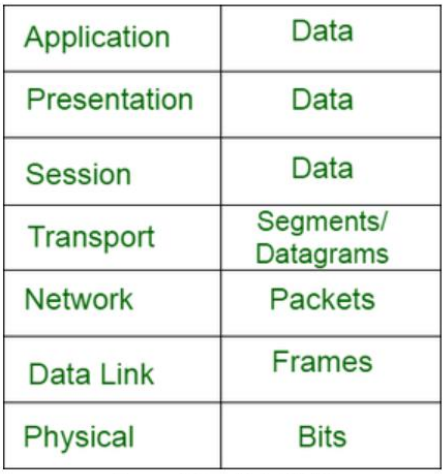
6. **Presentation Layer (Layer 6)**
    a. **Role**: Responsible for **data formatting** and translation, encryption, and compression. It ensures data is presented in a readable format for the application layer.
    b. **Key Functions**:
        i. **Data translation** between different formats (e.g., ASCII to binary).
        ii. **Encryption** and **compression** of data to ensure security and efficiency.
    c. **Example**: Data encryption in HTTPS, JPEG or PNG image formatting.

7. **Application Layer (Layer 7)**
    a. **Role**: Provides **services** for end-user applications and is the closest layer to the end user. It enables communication with software applications that interact with the network.
    b. **Key Functions**:
        i. Provides network services to applications (e.g., web browsing, email).
    c. **Example**: HTTP (web browsing), SMTP (email), FTP (file transfer), DNS.

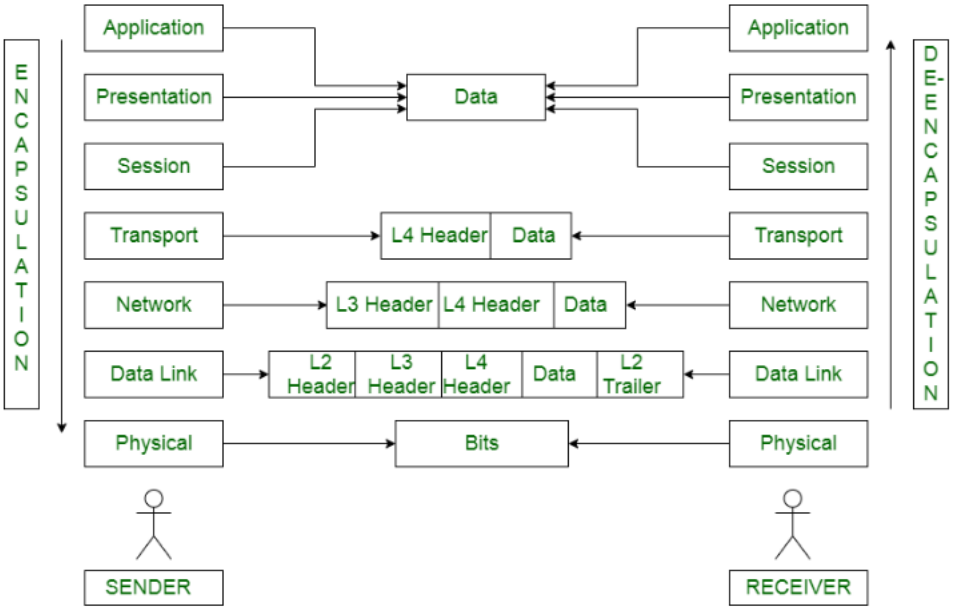
Summary of Layer Functions:

Layer	Role	Key Protocols/Examples

7. Application	Interface for applications and network services	HTTP, SMTP, FTP, DNS
6. Presentation	Data formatting, encryption, and compression	SSL/TLS, JPEG, ASCII
5. Session	Establishes, maintains, and terminates sessions	SMB, NetBIOS, RPC
4. Transport	Reliable data transfer, flow control, error recovery	TCP, UDP
3. Network	Routing and forwarding of packets	IP, ICMP, Routers
2. Data Link	Node-to-node communication, error detection	Ethernet, MAC addresses, Switches
1. Physical	Physical transmission of raw bitstreams	Cables, Fiber, Hubs, Radio Waves



Protocol Data Unit (PDU)



Encapsulation & De-encapsulation

L2 Header - contains information like source MAC address, destination MAC address
L2 Trailer - used for error checking
L3 Header - contains information like source IP, destination IP
L4 Header - contains information like source port, destination port, sequence number

TCP/IP Model :

The TCP/IP model consists of **four layers :**

- **Application Layer**: Supports end-user applications (uses HTTP, FTP, DNS).
- **Transport Layer**: Ensures reliable or fast data transmission (uses TCP, UDP).
- **Internet Layer**: Handles IP addressing and routing (uses IP, ICMP).
- **Network Access Layer/ Link Layer**: Manages physical data transmission

Comparison with OSI Model:

The TCP/IP model has **four layers**, compared to the **seven layers** of the OSI model. The TCP/IP model combines certain OSI layers:

- The **Application Layer** in TCP/IP corresponds to the **Application**, **Presentation**, and **Session** layers in the OSI model.
- The **Network Access Layer** covers the **Data Link** and **Physical** layers of the OSI model.

Protocols :

Protocols are standardized rules and conventions for communication between devices in a network. They define how data is formatted, transmitted, and received, ensuring reliable and accurate data exchange.

Types of Protocols:

1. Application Layer Protocols:

These protocols operate at the top layer of the TCP/IP or OSI model, providing network services directly to end-user applications.

- **HTTP (Hypertext Transfer Protocol)**:
    - **Function**: Used for transferring web pages on the internet.
    - **Port**: 80
    - **Role**: Fetches data (web pages) from a web server and displays it in a browser.

- **HTTPS (Hypertext Transfer Protocol Secure)**:
    - **Function**: Secure version of HTTP, encrypts data for secure transmission.
    - **Port**: 443
    - **Role**: Used for secure communication over the internet (e.g., online banking, shopping).

- **FTP (File Transfer Protocol)**:
    - **Function**: Transfers files between computers.
    - **Port**: 21
    - **Role**: Used for uploading/downloading files to/from a server.

- **SMTP (Simple Mail Transfer Protocol)**:
    - **Function**: Sends emails between servers.
    - **Port**: 25
    - **Role**: Handles email transmission between mail servers.

- **DNS (Domain Name System)**:
    - **Function**: Translates domain names (like google.com) into IP addresses.
    - **Port**: 53
    - **Role**: Resolves human-readable addresses to machine-readable IPs.

- **DHCP (Dynamic Host Configuration Protocol)**:
    - **Function**: Automatically assigns IP addresses to devices on a network.
    - **Port**: 67 (server), 68 (client)
    - **Role**: Ensures that devices receive an IP address when they join a network.

- **POP3 (Post Office Protocol 3):**
    - **Function**: POP3 is an **application layer protocol** used for retrieving email from a remote mail server to a local client. It downloads emails to the device and, by default, **removes them from the server** after retrieval.
    - **Port**: **110** (unencrypted), **995** (POP3S – secure, using SSL/TLS)

- **IMAP (Internet Message Access Protocol):**

- **Function**: IMAP is an **application layer protocol** that allows users to access and manage their emails directly on the mail server, enabling **synchronization across multiple devices**. Emails remain on the server unless explicitly deleted by the user.
- **Port**: **143** (unencrypted), **993** (IMAPS – secure, using SSL/TLS)

2. Transport Layer Protocols:

These protocols manage the delivery of data across networks, ensuring that data is sent and received correctly.

- **TCP (Transmission Control Protocol)**:
  - **Function**: Provides reliable, connection-oriented communication between devices.
  - **Port**: Varies
  - **Role**: Ensures data is delivered in the correct order without errors by establishing a connection between sender and receiver.

- **UDP (User Datagram Protocol)**:
  - **Function**: Provides faster, connectionless communication but without reliability.
  - **Port**: Varies
  - **Role**: Suitable for time-sensitive applications like video streaming, where speed is more important than accuracy.

3. Internet Layer Protocols:

These protocols deal with the logical addressing and routing of data across multiple networks.

- **IP (Internet Protocol)**:
  - **Function**: Assigns IP addresses and routes data between networks.
  - **Versions**: IPv4 (32-bit address), IPv6 (128-bit address).
  - **Role**: Defines how data packets are structured and how they are addressed for delivery across networks.

- **ICMP (Internet Control Message Protocol)**:
  - **Function**: Used for sending error messages and operational information about network issues.
  - **Role**: Commonly used by tools like **ping** to diagnose network connectivity.

- **ARP (Address Resolution Protocol)**:
    - **Function**: Maps an IP address to a physical MAC address.
    - **Role**: Ensures that data sent over a network reaches the correct device by translating logical addresses (IP) to physical addresses (MAC).

4. Security Protocols:

These protocols ensure the security of data being transmitted over a network by encrypting it or ensuring it hasn't been tampered with.

- **SSL/TLS (Secure Sockets Layer / Transport Layer Security)**:
    - **Function**: Encrypts communication between a web browser and a web server.
    - **Role**: Ensures secure communication by encrypting sensitive data like passwords, credit card numbers.

- **IPSec (Internet Protocol Security)**:
    - **Function**: Secures IP communication by authenticating and encrypting each packet in a data stream.
    - **Role**: Commonly used for creating secure VPN tunnels over the internet.

- **SSH (Secure Shell)**:
    - **Function**: Provides a secure way to access remote devices over an unsecured network.
    - **Port**: 22
    - **Role**: Used for remote login, secure file transfer, and command execution on remote servers.

IPv4 Address:

- **Definition**: IPv4 (Internet Protocol version 4) is a 32-bit addressing system used to identify devices on a network. It uses decimal notation, separated by dots (e.g., 192.168.0.1), allowing for **4.3 billion unique addresses**.
- **Structure**: Divided into four octets (8-bit numbers), each ranging from 0 to 255.

Reserved IP Ranges:

- **Private IP Addresses** (non-routable on the internet):
  - **10.0.0.0 – 10.255.255.255** (Class A)
  - **172.16.0.0 – 172.31.255.255** (Class B)
  - **192.168.0.0 – 192.168.255.255** (Class C)
- **Loopback Range** (for local testing):
  - **127.0.0.0 – 127.255.255.255** (commonly **127.0.0.1** as "localhost").
- **Link-Local Addresses** (for automatic local networking):
  - **169.254.0.0 – 169.254.255.255** (used when DHCP fails).

IPv4 Address Classes:

Class	Starting IP	Ending IP	Default Subnet Mask	Network Bits	Host Bits	Purpose
A	1.0.0.0	126.255.255.255	255.0.0.0	8	24	Large networks
B	128.0.0.0	191.255.255.255	255.255.0.0	16	16	Medium-sized networks
C	192.0.0.0	223.255.255.255	255.255.255.0	24	8	Small networks
D	224.0.0.0	239.255.255.255	N/A	N/A	N/A	Multicast addresses
E	240.0.0.0	255.255.255.255	N/A	N/A	N/A	Experimental, reserved

Classless Inter-Domain Routing (CIDR) Prefix:

CIDR replaces the old class-based system to allow more efficient and flexible IP address allocation. It uses a suffix (CIDR prefix) to indicate how many bits represent the network portion of the address, regardless of the class.

- **CIDR Notation**: `IP Address/Prefix Length`
  - Example: `192.168.1.0/24`
    - This means the first 24 bits are used for the network, and the remaining 8 bits are for hosts.
- **Flexible Allocation**: Allows for subnets of varying sizes, enabling more efficient use of IP addresses (e.g., `/16`, `/28`).

Subnet Masking:

A **subnet mask** is used to divide an IP address into a network portion and a host portion.

Example:

- **IP Address**: `192.168.1.10`
- **Subnet Mask**: `255.255.255.0` (which is `/24` in CIDR notation)
- This means the first 24 bits (`192.168.1`) identify the network, and the last 8 bits (`.10`) identify the host on that network.

NAT (Network Address Translation) :

NAT (Network Address Translation) is a process used in networking to modify IP address information in packet headers while they are in transit, allowing multiple devices on a private network to share a single public IP address. It is commonly used in routers and firewalls.

Public vs. Private IP:

Attribute	Public IP	Private IP
Usage	Accessing devices on the internet	Internal network communication
Uniqueness	Globally unique	Can be reused across networks
Routability	Routable on the internet	Not routable on the internet, Need **Network Address Translation (NAT)** to map the private IPs to a public IP for internet connectivity.
Assigned By	ISPs and Internet Assigned Numbers Authority (IANA)	Local Network Administrators (e.g., DHCP)

IPv6 Address:

- **Definition**: IPv6 (Internet Protocol version 6) is a 128-bit addressing system developed to overcome the limitations of IPv4 (address exhaustion). It provides a vastly larger address space (approximately **340 undecillion addresses**).
- **Structure**: Written in hexadecimal notation and separated by colons (e.g., **2001:0db8:85a3:0000:0000:8a2e:0370:7334**). Leading zeros in each segment can be omitted (e.g., **2001:db8::8a2e:370:7334**).

MAC Address:

- **Definition**: A **Media Access Control (MAC) address** is a unique 48-bit identifier assigned to network interface cards (NICs) by manufacturers. It is used for communication within the same network (Layer 2 in the OSI model).

- **Structure**: Typically represented in hexadecimal format, with six pairs of digits separated by colons or hyphens (e.g., **00:1A:2B:3C:4D:5E**).
- **Function**: MAC addresses are used for data link layer communication within local networks. They are globally unique for each device.

Sockets:

- **Definition**: A **socket** is a combination of an IP address and a port number that represents a specific communication endpoint in a network. Sockets are used to establish and manage connections between devices on a network.
- **Structure**: Typically represented as **<IP Address>:<Port Number>** (e.g., **192.168.1.10:80**).

Types of Sockets:

- **Stream Sockets (TCP)**: Used for reliable, connection-oriented communication (e.g., web browsing).
- **Datagram Sockets (UDP)**: Used for connectionless, unreliable communication (e.g., streaming).

Ports:

- **Definition**: A **port** is a logical channel through which communication happens. Ports allow multiple services to run on a single IP address by distinguishing between different types of network traffic.
- **Port Numbers**: Represented as 16-bit numbers, ranging from **0 to 65535**.

Port Categories:

1. **Well-Known Ports (0–1023)**: Reserved for specific services and applications.
   Examples:
   i. **80**: HTTP (web browsing)
   ii. **443**: HTTPS (secure web browsing)
   iii. **25**: SMTP (email sending)
   iv. **22**: SSH (secure remote access)
   v. **53**: DNS (domain name resolution)

2. **Registered Ports (1024–49151)**: Used by software applications or processes that are not system-specific but still widely used.

Example: **3306** (MySQL database).

3. **Dynamic/Private Ports (49152–65535)**: Assigned dynamically to client-side applications when establishing outbound connections.
   Example: A browser connecting to a web server may use a dynamic port like **49160**.

HTTP Status Codes:

HTTP status codes are issued by a server in response to a client's request. They indicate whether the request was successful or encountered issues.

Categories of HTTP Status Codes:

1. **1xx (Informational)**:
   a. **Purpose**: Indicates that the request has been received and is being processed.
   b. **Example**:
      i. **100 Continue**: The server received the initial part of the request and the client can continue.
2. **2xx (Success)**:
   a. **Purpose**: Indicates that the request was successfully received, understood, and accepted.
   b. **Examples**:
      i. **200 OK**: The request succeeded.
      ii. **201 Created**: The request succeeded, and a resource was created.
3. **3xx (Redirection)**:
   a. **Purpose**: Indicates that further action is needed to complete the request.
   b. **Examples**:
      i. **301 Moved Permanently**: The resource has been moved to a new URL permanently.
      ii. **302 Found**: The resource is temporarily located at a different URL.
4. **4xx (Client Errors)**:
   a. **Purpose**: Indicates that the client's request contains bad syntax or cannot be fulfilled.
   b. **Examples**:
      i. **400 Bad Request**: The server cannot process the request due to client error.

    ii. **401 Unauthorized**: Authentication is required.
    iii. **403 Forbidden**: The server refuses to fulfill the request.
    iv. **404 Not Found**: The requested resource could not be found.
5. **5xx (Server Errors)**:
    a. **Purpose**: Indicates that the server failed to fulfill a valid request.
    b. **Examples**:
        i. **500 Internal Server Error**: A generic server error occurred.
        ii. **503 Service Unavailable**: The server is temporarily unable to handle the request.
        iii. **504 Gateway Timeout:** Server is acting as a gateway and cannot get a response in time.

HTTP Methods:

HTTP methods define the type of operation to be performed on a resource.

- **GET**: Retrieves data from the server (read-only).
- **POST**: Sends data to the server to create a new resource.
- **PUT**: Updates an existing resource with new data.
- **DELETE**: Deletes a specified resource on the server.
- **PATCH**: Partially updates an existing resource.
- **HEAD**: Similar to GET, but only retrieves headers, not the body.

Cookies:

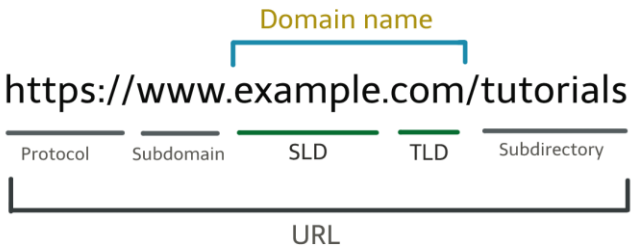
- **Definition**: Cookies are small pieces of data stored on the client's browser by a website. They help in storing user preferences, tracking sessions, and managing user authentication.
- **Types**:
    - **Session Cookies**: Temporary and deleted when the browser is closed.
    - **Persistent Cookies**: Stored on the user's device for a set period.
    - **Third-Party Cookies**: Set by domains other than the one the user is visiting (often used for advertising).

Use Cases:

- **Session Management**: Storing login information.
- **Personalization**: Remembering user preferences (e.g., language).
- **Tracking**: Monitoring user behavior across websites for analytics or marketing purposes.

Domain Name System (DNS):

The **Domain Name System (DNS)** is a system that translates human-readable domain names (e.g., `www.example.com`) into IP addresses (e.g., `192.0.2.1`). DNS serves as the "phonebook" of the internet, ensuring users can access websites and other services using familiar names instead of numerical IP addresses.



Key Components of DNS:

1. **DNS Resolver (Recursive Resolver)**:
    a. Acts as an intermediary between the client (user) and the DNS system.
    b. It handles the recursive queries, asking various DNS servers for the IP address.
2. **Root DNS Server**:
    a. The top of the DNS hierarchy.
    b. It directs queries to the appropriate **Top-Level Domain (TLD)** servers (e.g., `.com`, `.org`).
3. **TLD DNS Server**:
    a. Responsible for domains under a specific TLD (like `.com`, `.net`).
    b. It directs queries to the **Authoritative Name Server** for the specific domain.
4. **Authoritative DNS Server**:
    a. Holds the DNS records for the specific domain.
    b. It provides the requested IP address to the DNS resolver.

Types of DNS Records:

- **A Record**: Maps a domain name to an IPv4 address.
- **AAAA Record**: Maps a domain name to an IPv6 address.
- **CNAME Record**: Alias of one domain name to another.
- **NS Record**: Indicates the authoritative DNS servers for a domain.

DNS Flowchart:

1. **User** enters `www.example.com` in the browser.
2. The browser queries the **DNS Resolver**.
3. The **DNS Resolver** queries the **Root DNS Server**.
4. The **Root DNS Server** directs to the **TLD DNS Server** (for `.com`).
5. The **TLD DNS Server** directs to the **Authoritative DNS Server** (for `example.com`).
6. The **Authoritative DNS Server** provides the IP address for `www.example.com`.
7. The **DNS Resolver** returns the IP address to the browser.
8. The **Browser** connects to the website using the IP address.

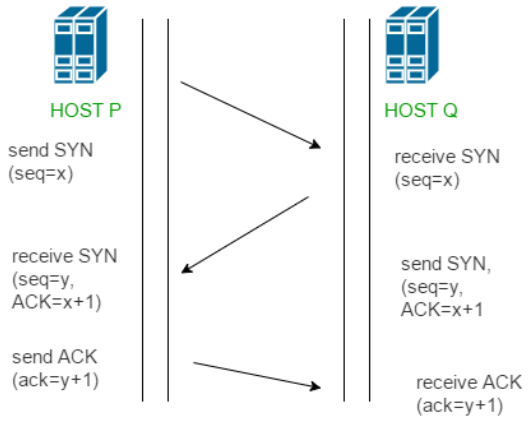
Three-Way Handshake (TCP Connection Establishment):

The **three-way handshake** is a process used by the **Transmission Control Protocol (TCP)** to establish a reliable connection between a client and a server. This process ensures that both the client and server are synchronized and ready for data transmission, avoiding packet loss or data corruption.

Steps of the Three-Way Handshake:

1. **SYN (Synchronize)**:
    a. **Client → Server**: The client sends a **SYN** packet (synchronize sequence number) to the server. This packet contains an initial sequence number (ISN) chosen by the client, which will be used for ordering the data during transmission.
    b. **Purpose**: The client is requesting a connection to the server.
2. **SYN-ACK (Synchronize + Acknowledgment)**:
    a. **Server → Client**: The server responds with a **SYN-ACK** packet. It acknowledges the client's SYN by setting the ACK flag and sending its own sequence number (server ISN).
    b. **Purpose**: The server acknowledges the client's request and sends its own sequence number to establish a connection from both ends.
3. **ACK (Acknowledgment)**:

a. **Client → Server**: The client sends an **ACK** packet back to the server. This packet acknowledges the server's SYN, completing the handshake.
b. **Purpose**: The connection is now established, and both the client and server are synchronized and ready to start data transmission.



How ARP Works ?

1. ARP Request: When a device wants to communicate with another device on the same local network, it knows the target's IP address but not its MAC address. It broadcasts an ARP request packet to all devices on the network, asking "Who has this IP address?"

2. ARP Reply: The device with the requested IP address responds with an ARP reply containing its MAC address.

3. Caching: The sender stores the IP-MAC mapping in its ARP cache to avoid repeatedly broadcasting ARP requests for future communication with the same device.

watch this Gif for better visualization :
https://www.practicalnetworking.net/wp-content/uploads/2017/01/traditional-arp-process.gif

Maximum Transmission Unit (MTU) :

The Maximum Transmission Unit (MTU) is the largest size, in bytes, of a data packet that can be sent over a network interface without needing to be fragmented. It includes both the headers (e.g.,

IP, TCP/UDP) and the data payload. For Ethernet networks, the standard MTU is typically 1500 bytes.

Checksum:

A checksum is a value used to verify the integrity of data in network communication. It is calculated from the contents of a data packet before it is sent and is included in the packet. When the packet is received, the checksum is recalculated and compared to the one sent. If the checksums match, the data is assumed to be intact; if not, it indicates that the data was corrupted during transmission. Checksums are used in both TCP and UDP for error detection.

Timer:

A timer in networking, especially in TCP, is used to manage the timing of data transmission and ensure reliable communication.
Retransmission Timer: If an acknowledgment (ACK) for a packet is not received within a specified time, the timer triggers a retransmission of the packet.
Timeouts: Timers also control how long a connection should wait before considering it has failed or timed out due to no response.

TTL (Time to Live) :

TTL (Time to Live) is a field in the header of an IP packet that indicates how long the packet is allowed to stay in the network before being discarded. It prevents packets from endlessly circulating in case of routing loops.

What happens when you enter google.com in the web browser?

1) Check the browser cache first if the content is fresh and present in the cache display the same.
2) If not, the browser checks if the IP of the URL is present in the cache (browser and OS) if not then requests the OS to do a DNS lookup using UDP to get the corresponding IP address of the URL from the DNS server to establish a new TCP connection.
3) A new TCP connection is set between the browser and the server using three-way handshaking.
4) An HTTP request is sent to the server using the TCP connection.

5) The web servers running on the Servers handle the incoming HTTP request and send the HTTP response.
6) The browser processes the HTTP response sent by the server and may close the TCP connection or reuse the same for future requests.
7) If the response data is cacheable then browsers cache the same.
8) Browser decodes the response and renders the content.

Types of Routing :

1. Static Routing:

- **Manually Configured**: Routes are manually defined by a network administrator.
- **Fixed Routes**: Once configured, static routes remain constant unless manually changed.
- **Best for Simple Networks**: Ideal for small, stable networks where routes do not change frequently.

Advantages:

- Simple to configure.
- No overhead from route computation or exchange.
- Offers more control and security.

Disadvantages:

- Not scalable for large, dynamic networks.
- No automatic route updates if the network changes (e.g., link failures).

2. Dynamic Routing:

- **Automatically Configured**: Routers learn and update routes automatically using routing protocols.
- **Adaptive**: Routes can change dynamically based on the network conditions (e.g., link failures, congestion).
- **Best for Large or Complex Networks**: Suitable for networks with frequent topology changes.

Advantages:

- Automatically updates routes as the network changes.
- Scalable for large and complex networks.
- Reduces the administrative burden.

Disadvantages:

- Consumes more CPU, memory, and bandwidth due to routing protocol overhead.
- Requires configuration and management of routing protocols.

3. Default Routing:

A specific type of static routing where a default route is defined (e.g., `0.0.0.0/0`), telling the router to send packets to a default gateway if there's no specific route for the destination.

AAA (Authentication, Authorization, and Accounting) :

AAA stands for **Authentication, Authorization, and Accounting**, a framework used in computer networking to manage access control, user identity verification, and usage tracking. It ensures secure and efficient management of network resources.

1. Authentication:

- **What it is**: The process of verifying the identity of a user or device trying to access the network.
- **How it works**: Users provide credentials (e.g., username, password, or digital certificates), which are validated by an authentication system.
- **Example**: A user logs into a VPN by providing a username and password, which are checked against a database.

2. Authorization:

- **What it is**: Determines what resources and services an authenticated user or device is allowed to access.
- **How it works**: Once authenticated, the system checks what privileges or permissions the user has and grants access based on predefined policies.
- **Example**: A user is authenticated to the network but is only authorized to access certain files or applications.

3. Accounting:

- **What it is**: The process of tracking the usage of resources and services by users.
- **How it works**: Records details such as login time, data usage, and commands executed by the user, which can later be analyzed for auditing or billing.
- **Example**: A network administrator can track how much bandwidth a particular user consumed during a session.

Forward Proxy:

A forward proxy sits in front of the client and forwards requests to various servers on behalf of the client. It is used to mask the client's identity, provide filtering, or cache requests.

Reverse Proxy:

A reverse proxy sits in front of the servers and forwards client requests to the backend servers. It masks the backend servers and provides load balancing, caching, and security functions.

What is an API (Application Programming Interface)?

An API is a set of rules and protocols that allows different software applications to communicate with each other. It defines the methods and data formats for interacting with a system or service, allowing developers to use predefined operations instead of building functionality from scratch.

Use Case: APIs are used to connect various software components, enabling services to communicate, such as retrieving weather data from a service or connecting a mobile app to a cloud database.

What is an API Gateway?

An API Gateway is a server that acts as an entry point for client requests to multiple backend services. It is often used in microservices architectures to handle routing, request transformations, security, rate limiting, and monitoring.

Key Functions:

Request Routing: Directs client requests to the appropriate backend services.
Security: Provides features like authentication, authorization, and SSL termination.
Load Balancing: Distributes incoming requests across multiple servers to balance the load.
Rate Limiting: Controls the number of requests a client can make in a given period.
Protocol Translation: Converts between different protocols (e.g., HTTP to WebSocket).

REST API :

A REST API (Representational State Transfer API) is an architectural style for designing networked applications. It allows clients to interact with server-side resources via standard HTTP methods such as GET, POST, PUT, and DELETE. REST APIs are stateless, meaning each request from the client to the server must contain all the information needed to process the request, and responses typically return data in formats like JSON or XML.

Key principles of REST include:

Stateless: No session data is stored on the server between requests.
Resource-based: Resources are identified by URLs.
Uses HTTP: Relies on HTTP methods for interactions with resources.

REST APIs are widely used for web services due to their simplicity and scalability.

Load Balancer:

A load balancer is a device or software that distributes incoming network traffic across multiple servers to ensure no single server is overwhelmed. Its primary purpose is to optimize resource use, improve performance, and increase the availability of applications by ensuring that all servers share the load evenly.

Key Functions of a Load Balancer:

1. **Distributes Traffic**: Balances incoming requests across multiple servers (also called a server farm or pool).
2. **Enhances Availability**: Ensures high availability by redirecting traffic to healthy servers if one fails.
3. **Improves Performance**: By spreading the load, it helps prevent overloading a single server, leading to faster response times.
4. **Fault Tolerance**: Detects server failures and reroutes traffic to functioning servers to ensure service continuity.

Types of Load Balancing:

- **Layer 4 (Transport Layer)**: Balances traffic based on network information like IP address and port.
- **Layer 7 (Application Layer)**: Balances traffic based on more complex application-level data, such as HTTP headers or URLs.

Horizontal Scaling vs Vertical Scaling:

Feature	Horizontal Scaling	Vertical Scaling
Method	Adding more machines	Increasing power of a single machine
Scalability	Highly scalable by adding more servers	Limited by physical machine capacity
Cost	Higher ongoing costs (more instances)	May require fewer upgrades initially
Complexity	Requires load balancing and distributed architecture	Simpler, as everything stays on one server
Fault Tolerance	High, as the load is shared across multiple servers	Low, single point of failure
Use Case	Web applications, cloud computing, microservices	Databases or apps needing more CPU/RAM

Common Networking Commands:

1. `ping`

- **Use Case**: Check the connectivity between your device and another device (e.g., server or router) on the network.
- **What It Checks**:
  - Verifies if the target host is reachable.
  - Measures round-trip time (latency) between devices.
  - Checks for packet loss.

Example: `ping google.com`

2. `traceroute` / `tracert` (Windows)

- **Use Case**: Trace the path that a packet takes from your device to a destination, showing all the routers (hops) in between.
- **What It Checks**:
  - Displays the route and measures the transit delays of packets across each hop.
  - Helps in identifying where delays or issues occur along the network path.

Example: `traceroute google.com`

3. `ifconfig` / `ip addr` (Linux) or `ipconfig` (Windows)

- **Use Case**: View and manage network interface configurations.
- **What It Checks**:
  - Displays information about all network interfaces, including IP address, MAC address, subnet mask, and more.
  - Use `ipconfig` on Windows to renew or release DHCP IP addresses.

Example (Linux): `ifconfig`

Example (Windows): `ipconfig`

4. `netstat`

- **Use Case**: Display active network connections, routing tables, and listening services.

- **What It Checks**:
    - Shows active TCP/UDP connections and their status.
    - Displays ports that are being listened on by services.
    - Can show routing tables and interface statistics.

Example: `netstat -an`

5. `nslookup`

- **Use Case**: Query DNS servers to find the IP address associated with a domain name or vice versa.
- **What It Checks**:
    - Helps diagnose DNS-related issues by querying the DNS servers directly.
    - Can be used to troubleshoot domain resolution problems.

Example: `nslookup google.com`

6. `dig` (Linux)

- **Use Case**: A more advanced DNS lookup tool that provides detailed query information.
- **What It Checks**:
    - Displays the DNS server response, including record types, TTL, and additional details.
    - Useful for in-depth DNS troubleshooting.

Example: `dig google.com`

7. `arp`

- **Use Case**: Display or modify the ARP (Address Resolution Protocol) table, which maps IP addresses to MAC addresses.
- **What It Checks**:
    - Lists the current ARP table, showing IP-to-MAC address mappings.
    - Can be used to troubleshoot connectivity between devices on a local network.

Example: `arp -a`

8. `route` / `ip route`

- **Use Case**: View and manipulate the IP routing table on a device.
- **What It Checks**:
  - Displays the current routing table, showing how packets are routed through different networks.
  - Useful for diagnosing routing issues or setting static routes.

Example: `route -n`

9. `curl`

- **Use Case**: Transfer data from or to a server using various protocols (HTTP, FTP, etc.).
- **What It Checks**:
  - Can be used to test API endpoints or web services.
  - Fetches HTTP headers or content from a web page for troubleshooting.

Example: `curl http://example.com`

10. `telnet`

- **Use Case**: Test connectivity to remote hosts on specific ports (for example, to see if a service like HTTP or SSH is running on a remote machine).
- **What It Checks**:
  - Verifies if a specific port is open and accessible.
  - Useful for troubleshooting application-layer connectivity issues.

Example: `telnet google.com 80`

Virtualization Vs Containerization :

Virtualization:

- **Definition**: Virtualization refers to creating multiple **virtual machines (VMs)** on a single physical machine. Each VM runs its own operating system (OS) and behaves like a separate physical computer.

- **How it works**: A hypervisor (software layer) runs on the physical hardware and manages multiple virtual machines. Each VM has its own OS and applications, and the hypervisor allocates CPU, memory, and storage to each VM.
- **Example**: VMware, Hyper-V, and KVM are popular hypervisors used for virtualization.
- **Benefits**:
    - Complete isolation of each VM with its own OS.
    - VMs can run different OS versions (Windows, Linux, etc.) on the same hardware.
    - Strong security because VMs are isolated from one another.
- **Drawbacks**:
    - VMs are **resource-heavy** because each VM requires its own OS, leading to more CPU, memory, and disk usage.
    - Slower startup times because an entire OS needs to boot up.

Containerization:

- **Definition**: Containerization packages applications and their dependencies into **containers**. Containers share the host machine's operating system kernel but remain isolated from each other.
- **How it works**: Unlike VMs, containers use the host OS, and the container engine (e.g., Docker, Kubernetes) manages containers. Each container includes the application and its dependencies (libraries, binaries) but does not need a separate OS instance.
- **Example**: Docker and Kubernetes are the most popular containerization platforms.
- **Benefits**:
    - **Lightweight** compared to VMs, since containers share the same OS kernel.
    - Faster startup times as containers don't need to boot an entire OS.
    - Easier to deploy, manage, and scale applications.
- **Drawbacks**:
    - Less isolation than VMs because containers share the same OS kernel.
    - Slightly less secure, as a vulnerability in the host OS could affect multiple containers.

Key Differences:

Aspect	Virtualization (VMs)	Containerization (Containers)
OS Requirement	Each VM has its own OS	Containers share the host OS
Resource Usage	Heavy (more memory and CPU for each VM)	Lightweight (less overhead, uses shared resources)
Boot Time	Slow (boots entire OS)	Fast (just starts the application)
Isolation Level	Strong isolation (full OS per VM)	Lightweight isolation (shares OS kernel)

Security	More secure (stronger isolation)	Less secure (kernel-level vulnerabilities)
Management	More complex (managing full OS per VM)	Easier (managing containers only)
Performance	Slower due to OS overhead	Near-native performance

Example Analogy:

- **Virtualization**: It's like running multiple **computers** (VMs) on one physical server, each with its own OS, apps, and resources.
- **Containerization**: It's like running multiple **apps** (containers) on the same computer, all sharing the same OS but isolated from each other.

Firewall :

A **firewall** is a network security device or software that monitors and controls incoming and outgoing network traffic based on predetermined security rules. Firewalls act as a barrier between a trusted internal network and untrusted external networks (like the internet), preventing unauthorized access while permitting legitimate communications.

Types of Firewalls:

1. **Packet-Filtering Firewall**:
    - **How it works**: Examines the headers of incoming and outgoing packets (e.g., source and destination IP addresses, ports) and applies rules based on this information.
    - **Advantages**: Fast and simple; works at the network layer.
    - **Disadvantages**: Limited to inspecting packet headers, can't detect complex attacks or stateful threats.
    - **Example use case**: Small networks with basic security requirements.

2. **Stateful Inspection Firewall**:
    - **How it works**: Tracks the **state** of active connections (TCP/UDP) and makes decisions based on the context of traffic (i.e., it understands whether a packet is part of an existing connection or not).
    - **Advantages**: More secure than packet-filtering firewalls because it analyzes packet context.
    - **Disadvantages**: More resource-intensive.
    - **Example use case**: Larger enterprises requiring enhanced security.

3. **Proxy Firewall (Application-Level Gateway)**:
    - **How it works**: Acts as an intermediary between internal and external networks. It receives incoming traffic, inspects it at the application layer (Layer 7 of OSI), and then forwards it to the destination if deemed safe.
    - **Advantages**: Provides deep packet inspection, can filter traffic based on content, and hides internal network addresses (proxying).
    - **Disadvantages**: Slower because it needs to process all application-layer data, complex configuration.
    - **Example use case**: Secure environments where content-level filtering is required.

4. **Next-Generation Firewall (NGFW)**:
    - **How it works**: Combines traditional firewall features with advanced security functionalities like **deep packet inspection (DPI)**, **intrusion prevention systems (IPS)**, **application awareness**, and the ability to detect modern threats (malware, ransomware).
    - **Advantages**: Highly effective against advanced threats, can analyze encrypted traffic, and offers greater visibility into applications.
    - **Disadvantages**: Expensive and resource-heavy.
    - **Example use case**: Enterprises needing to combat sophisticated attacks like zero-day exploits and Advanced Persistent Threats (APTs).

5. **Unified Threat Management (UTM) Firewall**:
    - **How it works**: Integrates multiple security features such as firewall, intrusion detection, antivirus, VPN, and content filtering into one device.
    - **Advantages**: Easy management of multiple security features in one appliance.
    - **Disadvantages**: May have performance issues if multiple security features are enabled simultaneously.
    - **Example use case**: Small to medium-sized businesses looking for an all-in-one security solution.

6. **Cloud Firewall (Firewall as a Service)**:
    - **How it works**: A firewall service that is hosted in the cloud rather than on-premises. It protects cloud infrastructure and can provide firewall services to virtual networks.
    - **Advantages**: Scalable, cost-effective, and easy to deploy across distributed environments.
    - **Disadvantages**: Reliant on internet connectivity.
    - **Example use case**: Organizations with cloud infrastructure or hybrid networks.

Network attacks :

Network attacks are attempts to disrupt, compromise, or gain unauthorized access to a network or its resources. They target vulnerabilities in the infrastructure, protocols, and security mechanisms to steal data, cause disruption, or manipulate traffic.

1. Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks

- **How it works**:
  - In a **DoS attack**, attackers flood a network, server, or device with overwhelming amounts of traffic or requests, exhausting its resources and making it unavailable to legitimate users.
  - In a **DDoS attack**, the traffic originates from multiple distributed sources (often compromised machines called **botnets**) to make detection and defense more difficult.
- **Impact**:
  - Services become slow or entirely unavailable, leading to business disruption, loss of customer trust, and financial damage.
- **Example**:
  - An HTTP flood attack that overwhelms a web server with HTTP requests until it becomes unresponsive.

2. Man-in-the-Middle (MITM) Attack

- **How it works**:
  - The attacker intercepts communication between two parties, allowing them to eavesdrop, alter, or inject malicious content into the communication without either party knowing.
- **Impact**:
  - The attacker can steal sensitive information (e.g., login credentials, financial details), alter messages, or launch further attacks.
- **Example**:
  - **HTTPS spoofing**, where the attacker intercepts and decrypts supposedly secure traffic between a browser and a server.

3. Phishing and Spear Phishing

- **How it works**:

    - **Phishing** involves sending fraudulent messages (usually via email) that appear to come from a legitimate source to trick users into providing sensitive information or clicking on malicious links.
    - **Spear phishing** is a targeted version where specific individuals or organizations are targeted with personalized messages.
- **Impact**:
    - Users may unknowingly provide credentials, download malware, or reveal financial information.
- **Example**:
    - An email that appears to be from a trusted company asking the user to reset their password, but the link directs them to a malicious site.

4. Packet Sniffing (Eavesdropping Attack)

- **How it works**:
    - Attackers use software tools (like Wireshark) to capture and inspect unencrypted data as it travels across a network.
- **Impact**:
    - Sensitive data, such as usernames, passwords, credit card numbers, or any unencrypted communication, can be intercepted.
- **Example**:
    - Sniffing on public Wi-Fi networks to capture login credentials transmitted over insecure HTTP websites.

5. SQL Injection

- **How it works**:
    - Attackers insert or manipulate **SQL queries** in input fields of web applications, tricking the system into executing unintended commands. This can lead to unauthorized access to databases.
- **Impact**:
    - Attackers can view, modify, or delete sensitive database information, including customer data and login credentials.
- **Example**:
    - Inserting malicious SQL commands into a login form to bypass authentication (e.g., `admin' OR '1'='1`).

6. DNS Spoofing (DNS Cache Poisoning)

- **How it works**:
  - The attacker corrupts the **DNS cache** so that when a user tries to access a legitimate website (e.g., bank.com), they are instead redirected to a malicious website without their knowledge.
- **Impact**:
  - Users unknowingly provide credentials or personal information to malicious websites that appear legitimate.
- **Example**:
  - Poisoning the DNS cache to redirect users attempting to visit "www.bank.com" to an attacker's fake site that looks identical to the bank's real website.

7. IP Spoofing

- **How it works**:
  - The attacker sends malicious packets that appear to come from a trusted IP address, bypassing security mechanisms.
- **Impact**:
  - Attackers can bypass IP-based security restrictions or launch further attacks like **DDoS** using spoofed IPs.
- **Example**:
  - An attacker sends a malicious packet with a forged source IP address, tricking the firewall into allowing the packet through.

8. Cross-Site Scripting (XSS)

- **How it works**:
  - Attackers inject malicious scripts (usually JavaScript) into web pages viewed by users. When users visit the page, their browsers execute the malicious script.
- **Impact**:
  - The attacker can steal session cookies, deface websites, or launch phishing attacks.
- **Example**:
  - An attacker injects a malicious script into a comment section of a web page, which, when viewed by other users, executes and steals their session tokens.

9. Social Engineering Attacks

- **How it works**:
  - Attackers manipulate people into breaking normal security practices to gain unauthorized access to systems or data. Common tactics include impersonation, baiting, and pretexting.
- **Impact**:
  - Attackers can obtain sensitive information like passwords or access critical systems without using technical methods.
- **Example**:
  - An attacker calls an employee, pretending to be from the IT department, and tricks them into revealing their login credentials.

10. Brute Force Attack

- **How it works**:
  - Attackers use automated tools to systematically try different combinations of passwords or encryption keys until they find the correct one.
- **Impact**:
  - If successful, the attacker gains unauthorized access to systems, databases, or encrypted files.
- **Example**:
  - A brute-force attack on an admin login page, trying thousands of possible password combinations until the correct one is found.

11. Ransomware

- **How it works**:
  - Malicious software that encrypts the victim's data and demands a ransom for the decryption key.
- **Impact**:
  - Victims lose access to critical data, and attackers demand payment (usually in cryptocurrency) for restoration.
- **Example**:
  - The **WannaCry** ransomware attack that affected over 230,000 computers worldwide, demanding ransom payments for data decryption.

12. Session Hijacking

- **How it works**:
  - Attackers steal or predict session tokens (unique identifiers for user sessions) to take over an active user session without needing credentials.
- **Impact**:
  - The attacker can impersonate a legitimate user and perform unauthorized actions on their behalf.
- **Example**:
  - Hijacking a user's session on a web application by stealing their session cookie and accessing their account.

13. Malware-Based Attacks

- **How it works**:
  - Malware (viruses, worms, trojans, etc.) is introduced into a network to disrupt, steal, or damage data and systems. Malware can spread through email attachments, infected websites, or malicious downloads.
- **Impact**:
  - Malware can cause data loss, steal information, disrupt network operations, or provide a backdoor for attackers.
- **Example**:
  - **Trojans** that masquerade as legitimate software but provide a backdoor for attackers to control the infected system remotely.

Encryption:

Encryption is the process of converting plain, readable data (called **plaintext**) into an unreadable, scrambled format (called **ciphertext**) to protect its confidentiality. Only authorized parties who possess the decryption key can convert the ciphertext back into plaintext. It is one of the most

important techniques used in modern cybersecurity to ensure data privacy and secure communications.

Types of Encryption:

1. **Symmetric Encryption**:
    - **How it works**: The same key is used to both encrypt and decrypt the data.
    - **Key Management**: The key must be shared securely between the sender and the recipient.
    - **Common Algorithms**:
        - AES (Advanced Encryption Standard)
        - DES (Data Encryption Standard)
        - 3DES (Triple DES)
    - **Use Cases**: Disk encryption, secure communications, and file encryption.

    Example:

    Alice and Bob both use the same secret key to encrypt and decrypt a message. If an attacker intercepts the message, they cannot read it without the key.

2. **Asymmetric Encryption** (Public-Key Cryptography):
    - **How it works**: Uses two different keys—a **public key** for encryption and a **private key** for decryption.
    - **Key Management**: The public key is shared openly, but the private key is kept secret by the owner.
    - **Common Algorithms**:
        - RSA
        - Elliptic Curve Cryptography (ECC)
    - **Use Cases**: Secure web communication (SSL/TLS), digital signatures, email encryption, and secure key exchange.

    Example:

    Alice encrypts a message using Bob’s public key. Only Bob, with his private key, can decrypt it. Even if someone intercepts the encrypted message, they cannot decrypt it without Bob's private key.

Hashing :

Hashing is the process of converting data (such as a file or a message) into a fixed-size string of characters, typically a sequence of numbers and letters, using a mathematical algorithm. The resulting output, called a **hash** or **digest**, is unique to the original data. Even a small change in the input drastically changes the hash.

Unlike encryption, hashing is **one-way**—you cannot reverse the hash to get the original data. It is primarily used for data integrity verification, password storage, and digital signatures.

Common Hashing Algorithms:

- **MD5** (Message Digest Algorithm 5)
- **SHA-256** (Secure Hash Algorithm 256-bit)

Uses:

- **Password storage**: Hashing passwords before storing them ensures that even if the database is compromised, the actual passwords remain hidden.
- **Data integrity**: Hashes verify that files or messages haven't been altered during transmission or storage.

Digital Certificate

A **digital certificate** is an electronic document used to prove the ownership of a public key. It is a key component of the **Public Key Infrastructure (PKI)**, enabling secure communications and authentication over the internet. The certificate binds the **public key** to the identity of an individual, organization, or device, ensuring that the entity owning the key is legitimate.

How a Digital Certificate Works:

1. **Key Pair Generation**:

The certificate owner generates a pair of cryptographic keys: a **public key** (shared with everyone) and a **private key** (kept secret).

2. **Certificate Authority (CA)**:

   The certificate is issued by a **Certificate Authority (CA)**, a trusted third party that verifies the identity of the certificate requester and signs the certificate to establish its authenticity.

3. **Certificate Structure**:

   A digital certificate typically contains:
   i. **Public key** of the owner.
   ii. **Owner's information** (e.g., organization name, domain name).
   iii. **Issuer's information** (CA's name, digital signature).
   iv. **Validity period** (start and expiration date).
   v. **Serial number** (unique identifier).
   vi. **Signature algorithm** used by the CA to sign the certificate.
4. **Verification Process**:

   When someone receives a digital certificate, they verify it by checking the CA's digital signature on the certificate. If the signature is valid, the public key is trusted, and secure communication can proceed.

Common Uses of Digital Certificates:

- **SSL/TLS for websites**: Ensures secure communication (HTTPS) between web browsers and servers.
- **Email encryption**: Ensures that emails are securely transmitted.
- **Code signing**: Verifies the authenticity of software and that it hasn't been tampered with.

Digital Certificate

A **digital certificate** is an electronic document used to prove the ownership of a public key. It is a key component of the **Public Key Infrastructure (PKI)**, enabling secure communications and

authentication over the internet. The certificate binds the **public key** to the identity of an individual, organization, or device, ensuring that the entity owning the key is legitimate.

How a Digital Certificate Works:

1. **Key Pair Generation**:
    a. The certificate owner generates a pair of cryptographic keys: a **public key** (shared with everyone) and a **private key** (kept secret).
2. **Certificate Authority (CA)**:
    a. The certificate is issued by a **Certificate Authority (CA)**, a trusted third party that verifies the identity of the certificate requester and signs the certificate to establish its authenticity.
3. **Certificate Structure**:
    a. A digital certificate typically contains:
        i. **Public key** of the owner.
        ii. **Owner's information** (e.g., organization name, domain name).
        iii. **Issuer's information** (CA's name, digital signature).
        iv. **Validity period** (start and expiration date).
        v. **Serial number** (unique identifier).
        vi. **Signature algorithm** used by the CA to sign the certificate.
4. **Verification Process**:
    a. When someone receives a digital certificate, they verify it by checking the CA's digital signature on the certificate. If the signature is valid, the public key is trusted, and secure communication can proceed.

Common Uses of Digital Certificates:

- **SSL/TLS for websites**: Ensures secure communication (HTTPS) between web browsers and servers.
- **Email encryption**: Ensures that emails are securely transmitted.
- **Code signing**: Verifies the authenticity of software and that it hasn't been tampered with.

Digital Signature

A **digital signature** is a cryptographic technique used to verify the authenticity and integrity of a message, document, or piece of data. It assures the recipient that the message was created and sent by the legitimate sender (authentication) and that the message was not altered in transit (integrity).

How Digital Signature Works :

1. **Sender's Actions**:
    a. Hash the message to get a digest.
    b. Encrypt the digest with the private key to create the digital signature.
    c. Send the message and the signature.
2. **Recipient's Actions**:
    a. Hash the received message.
    b. Decrypt the signature using the sender's public key to get the original digest.
    c. Compare the two digests. If they match, the message is authentic and unaltered.

Common Uses of Digital Signatures:

- **Document signing**: Used to sign contracts, legal documents, etc., ensuring that the document hasn't been altered.
- **Code signing**: Ensures that software or application updates are from a trusted source and haven't been tampered with.
- **Email signing**: Ensures that the email content and sender are authentic.

Additional Resources :

1) https://youtu.be/IPvYjXCsTg8?si=GkCr_J0B2UiBYtli

2) Top Networking Interview Questions (2024) - InterviewBit

3) Top 25 Networking Interview Questions and Answers - InterviewPrep