The Institute of Electrical and Electronics Engineers (IEEE): is an international professional organization that develops and maintains standards for a wide range of industries, particularly in technology, telecommunications, and computing. Founded in 1963, IEEE is one of the largest organizations of its kind, with a global membership focused on advancing technological innovation and setting standards that help ensure compatibility, interoperability, and safety in tech industries.

IEEE 802.3 standard: defines the physical and data link layers for Ethernet, including the frame structure, MAC addressing, physical connections, media types, and the speed of data transmission. It is split into multiple sections covering various Ethernet specifications, such as frame structure, speeds, and operational methods. Key parts of the standard include specifications for copper, fiber-optic, and wireless media, as well as variations that handle different transmission speeds (from 10 Mbps to 400 Gbps or more).

Ethernet: is a **networking technology and standard (IEEE 802.3)** used to connect devices within a local area network (LAN). It defines how devices format and transmit data over cables or wireless connections, enabling them to communicate efficiently. Ethernet specifies the structure of data packets, physical cabling, transmission speeds, and protocols that control data flow, allowing devices to share resources and communicate on the same network. Here's what that means:

1. **Standard**: Ethernet is defined by the **IEEE 802.3 standard**, which sets out how data should be formatted and transmitted across a LAN. This standard governs the rules for data transmission, such as speed, cabling types, frame structure, and how devices access the network.
2. **Protocol**: Within the Ethernet standard, there are rules or **protocols** for communication at the data link layer of the OSI model, allowing devices to format, send, and receive data.
3. **Physical Medium**: Ethernet often uses **cables** (like twisted-pair or fiber optic cables) that connect devices in a LAN. These cables are often called "Ethernet cables" because they're compatible with Ethernet technology.

4. **Network Type**: An **Ethernet network** refers to a LAN that follows the IEEE 802.3 standard, allowing devices like computers, printers, and routers to connect and communicate over a shared medium.

The Internet Engineering Task Force (IETF) is a global, open community of network designers, engineers, researchers, and other experts who develop and promote voluntary Internet standards. The IETF is responsible for creating and maintaining protocols that enable the internet to function, including foundational technologies like TCP/IP, HTTP, and DNS. Founded in 1986, the IETF operates as an independent body under the Internet Society (ISOC) and holds regular meetings where members collaborate to address technical challenges in networking and set new standards.

When a new protocol, such as HTTP/2, is proposed to improve web performance, it's reviewed by IETF working groups. After extensive collaboration, testing, and review, the proposal can be finalized as an RFC. This allows web servers, browsers, and other devices to implement the new protocol in a consistent way, enhancing compatibility across the internet.

Examples of IETF Standards: HTTP/HTTPS, IPv4 and IPv6 SMTP (The protocol used for email transmission), TLS

Key Functions:

- **Developing Internet Standards:** The IETF is responsible for creating technical standards that define how data moves across the internet. These standards help ensure interoperability between different networks and devices. Notably, the IETF develops and maintains the TCP/IP protocol suite, which is the foundation of the internet.
- **Publishing RFCs**: The IETF publishes its standards and other documents as **RFCs (Requests for Comments)**, which serve as guidelines or technical documents. RFCs cover everything from core networking protocols to experimental protocols and policies. Each RFC is a proposal or specification that has been reviewed by the IETF community.
- **Problem-Solving and Improvement**: The IETF addresses issues and updates standards to improve the reliability, security, and

scalability of internet protocols. The organization regularly evaluates and improves upon standards as technology evolves.

- **Collaborative and Open Membership**: Anyone can participate in the IETF, and decisions are made through a collaborative, consensus-based process rather than a formal membership structure. This open approach allows for a wide range of ideas and expertise, promoting the development of robust and adaptable standards.

Request for Comments (RFC): are a series of technical documents and standards published by the Internet Engineering Task Force (IETF). These documents are used to define and develop the protocols and technologies that form the foundation of the internet. For example, **RFC 1918** is the official document that defines private IP address ranges for use in local networks. Or **RFC 2616**, which defines the **HTTP/1.1 protocol**—the standard for how web browsers and servers communicate on the internet. Or **RFC 826** Address Resolution Protocol

IEEE vs IETF:

1. **Scope and Focus:**
    a. **IEEE**: IEEE primarily develops standards for the **physical and data link layers** of network communication, focusing on hardware and data transmission. IEEE standards cover everything from Ethernet (IEEE 802.3) and Wi-Fi (IEEE 802.11) to electrical engineering, telecommunications, and various industrial standards.IEEE operates through working groups under its **Standards Association (IEEE-SA)**, which brings together engineers and companies to draft and review standards. IEEE has a formal voting process and publishes final standards through a formal, regulated approach.
    b. **IETF**: The IETF, in contrast, focuses on **higher-level protocols and applications** for the internet. This includes standards for the **network layer and above**, such as IP (Internet Protocol), TCP (Transmission Control Protocol), HTTP (Hypertext Transfer Protocol), and DNS (Domain Name System). These protocols enable global network interoperability, security, and scalability.

2. **Organizational Structure and Process:**
    a. **IEEE**: operates through working groups under its **Standards Association (IEEE-SA)**, which brings together engineers and companies to draft and review standards. IEEE has a formal voting process and publishes final standards through a formal, regulated approach.
    b. **IETF**: The IETF is less formal and operates through a consensus-driven, open participation process. Any interested individual can join and contribute to IETF discussions, and its standards are developed in **working groups** and published as **RFCs (Requests for Comments)**. The IETF does not rely on formal voting but instead reaches decisions based on rough consensus.

RFC 826. Address Resolution Protocol (ARP): is used to map **IP addresses** to **MAC addresses**, enabling devices to find each other within the LAN. Since the Data Link Layer (Layer 2) relies on MAC addresses to direct data within a network, ARP helps bridge the gap between Layer 3's IP-based routing and Layer 2's MAC-based delivery.

Command:
arp -h
cat /proc/net/arp

ARP Request Structure:

- **Hardware Type:** Specifies the type of hardware used (1 for Ethernet).
- **Protocol Type:** Specifies the protocol used for addressing (0x0800 (IPv4)).
- **Hardware Address Length:** Length of the MAC address (6 bytes for Ethernet).
- **Protocol Address Length:** Length of the IP address (4 bytes for IPv4).
- **Operation Code:** Indicates whether the packet is a request (1) or a response (2).

- **Sender MAC Address:** The MAC address of the sender. (00:1A:2B:3C:4D:5E)
- **Sender IP Address:** The IP address of the sender. (192.168.1.5)
- **Target MAC Address:** The MAC address of the target (set to 00:00:00:00:00:00 in a request).
- **Target IP Address:** The IP address of the target. (192.168.1.10)

How it works:
When a device needs to send a packet to another device on the same network, it first checks if it knows the destination device's MAC address. If it doesn't, it uses ARP to find it. Here's a step-by-step example of how ARP works:

1. **ARP Request**: Suppose Device A with IP address **192.168.1.10** wants to communicate with Device B, which has an IP address of **192.168.1.20**. However, Device A only knows Device B's IP address, not its MAC address. Device A will send out an **ARP request**, essentially asking, "Who has IP 192.168.1.20? Tell me your MAC address." This ARP request is broadcast to all devices on the local network.
2. **ARP Response**: When Device B receives the ARP request, it recognizes its own IP address in the request and replies with an **ARP response**. This response includes Device B's MAC address, allowing Device A to store it in its **ARP cache** for future use. The ARP cache is a table that maps IP addresses to MAC addresses to reduce the need for repeated ARP requests.
3. **Packet Delivery**: Now that Device A has both the IP and MAC addresses of Device B, it can use these addresses to construct a frame and send the data packet directly to Device B on the local network.

Internet Assigned Numbers Authority (IANA): is a critical organization that maintains and allocates several essential components for global internet functionality. Established originally as part of the U.S. government's ARPANET project, IANA now operates under the Internet Corporation for Assigned Names and Numbers (ICANN).

Here are the primary functions of IANA in more technical detail:

1. **IP Address Allocation**: IANA administers the global allocation of IP addresses. This includes the distribution of large IP address blocks to Regional Internet Registries (RIRs) like ARIN, RIPE, and APNIC, which then distribute addresses to ISPs and organizations within their regions. This process ensures that no IP addresses are duplicated and that allocations follow a structured hierarchy.
2. **DNS Root Zone Management**: IANA is responsible for the top layer of the Domain Name System (DNS) hierarchy, often called the DNS root zone. This includes maintaining the list of top-level domains (TLDs) such as .com, .org, and country-code TLDs like .uk or .jp. IANA works with registries and registrars worldwide to ensure TLDs remain globally unique, reliable, and up-to-date.
3. **Protocol Parameter Assignment**: For internet protocols like TCP/IP to operate consistently, specific numbers and codes (known as protocol parameters) must be standardized. IANA manages these protocol parameters, such as port numbers, protocol numbers, and MIME types. This oversight ensures seamless interoperability across devices and networks by maintaining a centralized reference for these parameters.

Through these responsibilities, IANA serves as a backbone for internet governance and operational standards, enabling efficient and consistent network communication across the globe.

RFC 7042:

Node: is a physical electronic device hooked up to a network, for example a computer, printer, router, and so on. If set up properly, a node is capable of sending and/or receiving information over a network.

Host refers to any device on a network that has an IP address and can communicate with other devices. Hosts can be computers, servers, smartphones, or any other device that sends or receives data within a network. All hosts are nodes, but not all nodes are hosts - switches for example are just nodes of LAN.

Packets: In networking, a packet is a small segment of a larger message. Each packet contains both data and information about that data. The information about the packet's contents is known as the "header," and it goes at the front of the packet so that the receiving machine knows what to do with the packet. To understand the purpose of a packet header, think of how some consumer products come with assembly instructions. When data gets sent over the Internet, it is first broken up into smaller packets, which are then translated into bits. The packets get routed to their destination by various networking devices such as routers and switches. When the packets arrive at their destination, the receiving device reassembles the packets in order and can then use or display the data. Compare this process to the way the United States' Statue of Liberty was constructed.

Protocol: In networking, a protocol is a standardized set of rules for formatting and processing data. Network protocols are like a common language for computers. The computers within a network may use vastly different software and hardware; however, the use of protocols enables them to communicate with each other regardless.
There are protocols for sending packets between devices on the same network **(Ethernet)**, for sending packets from network to network **(IP),** for ensuring those packets successfully arrive in order **(TCP),** and for formatting data for websites and applications **(HTTP).** In addition to these foundational protocols, there are also protocols for routing, testing, and encryption.

Network: a collection of devices (like computers, phones, or printers) connected together to share information and resources. Networks can be small, like a home Wi-Fi network, or large, like the internet, and they allow devices to communicate, share files, and access shared resources like printers or servers.

Internet: a global network of computers and other devices connected together to share information and communicate. In fact, the word "Internet" could be said to come from this concept: **interconnected networks.**

Layer: in the OSI model is a structured level within the seven-layer stack, where each layer has designated functions and interacts with the layers directly above and below it. The purpose of the layered structure is to divide complex networking tasks into manageable sections, ensuring modularity and interoperability between different devices and systems.

How Layers Work Together:

- **Encapsulation**: When data is sent from one device to another, each layer on the sending side adds its own headers (and sometimes footers) to the data. These headers contain critical information relevant to each layer's function.
- **Decapsulation**: On the receiving end, each layer reads and removes the headers it recognizes, ensuring the data moves smoothly up the stack to the application layer.
- **Inter-Layer Communication**: Layers interact with their adjacent layers using standardized protocols, allowing data to flow in a structured way through both local and remote systems.

The Seven OSI Layers:

1. **Please | Physical Layer (Layer 1)** - Manages the physical connection, dealing with raw data transmission through cables, radio signals, etc.
2. **Do | Data Link Layer (Layer 2)** - Ensures node-to-node data transfer and error handling.
3. **Not | Network Layer (Layer 3)** - Handles logical addressing (IP addresses) and routing to ensure data reaches the correct destination across networks.
4. **Tell (The) | Transport Layer (Layer 4)** - Manages end-to-end data delivery and error-checking between devices.
5. **Secret | Session Layer (Layer 5)** - Establishes, manages, and terminates sessions (communication exchanges) between applications.
6. **Password (to) | Presentation Layer (Layer 6)** - Formats data for the application layer, handling encryption, compression, and data translation.

7. **Anyone | Application Layer (Layer 7)** - Interfaces with end-user applications, supporting protocols like HTTP, FTP, and SMTP.

Open System Interconnection (OSI) model: Is a way of understanding how different parts of a computer network work together to send data between devices. It's like a blueprint for communication, split into seven layers. Each layer has a specific role in handling data, from the physical connection (like cables or Wi-Fi) all the way up to the applications you use (like a web browser or email). By dividing network communication into layers, the OSI model makes it easier for engineers to design and troubleshoot network systems, as each layer has its own set of rules and focuses on a specific part of the communication process.

OSI Layer 1:

The Physical Layer is the foundational level of the OSI model and is primarily responsible for establishing and maintaining the physical connection between devices on a network. As the first and lowest layer in the OSI stack, it defines the means by which raw, unstructured data *bits* (binary 0s and 1s) are physically transmitted from one device to another, regardless of the type of network or data content. In this layer, data is treated solely as electrical, optical, or electromagnetic signals rather than meaningful information, with each bit representing a single element in a continuous data stream.

OSI Layer 2:

The **Data Link Layer**, or Layer 2 of the OSI model, allows nodes to communicate with each other within a LAN. It serves as the link between the physical hardware of the network and the higher layers, which deal more with data structure and routing. The Data Link Layer plays a crucial role in ensuring that data packets from the Network Layer (Layer 3) are formatted correctly for transmission across the physical medium and that they reach their next destination intact and error-free.

The Data Link Layer is further divided into two sublayers:

- **Logical Link Control (LLC):** handles communication between the Network Layer and the Data Link Layer, managing flow control and error-checking. It provides error detection to ensure frames are correctly received and ready for the next step in their journey through the network.
- **Media Access Control (MAC):** on the other hand, directly interacts with the Physical Layer to control how devices share access to the physical medium, preventing collisions and managing access for multiple devices.

The data unit on Layer 2 is an *Ethernet frame:* is the fundamental data unit in Ethernet networks. It encapsulates data and provides crucial information that allows devices on the same network to identify, interpret, and process it correctly. It`s format:

1. **Preamble** (7 bytes):
    a. **Purpose:** The preamble is a series of alternating `1` and `0` bits that synchronizes the clocks of the sending and receiving devices.
    b. **Structure**: `10101010...` repeated 7 times (7 bytes).
    c. **Explanation**: The preamble prepares the receiving device to recognize the start of the frame, ensuring both devices are “in sync” for reading the bits that follow.
2. **Start Frame Delimiter (SFD)** (1 byte):
    a. **Purpose**: The SFD marks the exact start of the Ethernet frame.
    b. **Structure**: `10101011`
    c. **Explanation**: This byte signals to the receiving device that the frame is about to begin, ending the synchronization phase started by the preamble.
3. **Destination MAC Address** (6 bytes):
    a. **Purpose**: Identifies the device(s) to which the frame is intended.
    b. **Explanation**: This field holds the MAC address of the destination device, ensuring that only the intended device (or devices, in the case of broadcast or multicast) processes the data in the frame.

4. **Source MAC Address** (6 bytes):
    a. **Purpose**: Identifies the sending device's MAC address.
    b. **Explanation**: This field helps the recipient know who sent the frame, which is useful for acknowledging receipt, tracking data flow, and for potential error reporting.
5. **Ethertype / Length** (2 bytes):
    a. **Purpose**: Indicates the type of protocol encapsulated in the frame (e.g., IPv4, IPv6, ARP) or the length of the payload data.
    b. **Explanation:**
        i. If the value in this field is `0x0600` (1536 in decimal) or greater, it represents the Ethertype, identifying the protocol (e.g., IPv4 is `0x0800`, ARP is `0x0806`).
        ii. If the value is less than `0x0600`, it indicates the length of the data field in bytes, as seen in older IEEE 802.3 standards.
    c. **Examples of Ethertypes**:
        i. `0x0800` = IPv4
        ii. `0x0806` = ARP
        iii. `0x86DD` = IPv6
6. **Payload / Data** (46–1500 bytes):
    a. **Purpose**: This field carries the actual data being transmitted, typically an IP packet or other protocol data.
    b. **Size**: Must be between 46 and 1500 bytes.
    c. **Explanation:**
        i. If the data is shorter than 46 bytes, padding bytes are added to meet the minimum frame size of 64 bytes (including header and FCS).
        ii. The maximum Ethernet payload size is 1500 bytes, limiting the frame's total size to 1518 bytes.
    d. **Note**: Jumbo frames allow larger payloads, but they are specific to certain high-performance network setups.
7. **Frame Check Sequence (FCS)** (4 bytes):
    a. **Purpose**: This field contains a checksum (specifically, a 32-bit CRC) that allows error-checking of the entire frame.
    b. **Explanation**:

i. The sender calculates a CRC value based on the frame's content and includes it in the FCS.
ii. The receiver recalculates the CRC upon receipt and compares it to the FCS value in the frame. If they match, the frame is accepted as error-free; if not, the frame is discarded as corrupted.

Maximum Transmission Unit (MTU): The maximum size of frames is called. When a network device gets a frame that is larger than its MTU, the data is either fragmented into smaller frames, or dropped. Historically, Ethernet has a maximum frame size of 1500 bytes. An Ethernet packet larger than 1500 bytes is called a *jumbo frame.*

Media Access Control (MAC) address: is a unique hardware identifier assigned to a **Network Interface Card (NIC)** in a device, like your laptop's Wi-Fi card or an Ethernet adapter. This identifier is "burned in" by the manufacturer, making it unique to each device. It's usually expressed in six groups of two hexadecimal digits separated by colons or hyphens (e.g., 00:1A:2B:3C:4D:5E). MAC addresses work at the **Data Link layer (Layer 2)** of the OSI model, within local networks (e.g., your home Wi-Fi network). Used to deliver packets within a local network (e.g., sending data from a laptop to a printer on the same Wi-Fi network).Unlike IP addresses, MAC addresses do not change. They are permanent identifiers tied to hardware, so devices retain the same MAC address regardless of the network they connect to:

- The **second least significant bit** of the first byte:
    - `A0:B1:C2:D3:E4:F5.` `A0 - 10010000.`
      A **globally unique** MAC address is assigned by a central authority, such as the IEEE. The purpose is to ensure that each network interface card (NIC) in the world has a unique identifier
    - `A2:B3:C4:D5:E6:F7.` `A2 - 10010010.`
      A **locally assigned** MAC address is one that is set by a user or a local administrator rather than a global authority. This means the device's MAC address may

not be unique globally, but it is managed within a specific network. Sometimes, administrators need to manually assign MAC addresses, like in a virtual machine or a custom network setup. This address is unique within the context of the local network, but not globally unique.

- The **least significant bit** of the first byte:
  - `A4:B1:C2:D3:E4:F5` `A4 - 10100100`. **Unicast** is a one-to-one communication method where data is sent from a single sender to a single specific receiver.
  - `01:00:5E:7F:AA:BB` `01 - 00000001`. **Multicast** is a one-to-many communication method where data is sent from one sender to multiple specific receivers simultaneously. In multicast transmission, data packets are sent to a specific group address. Only devices that have "subscribed" to that group receive the packets, which reduces network load. Streaming a live video conference or an online seminar where multiple participants watch the same stream.

A MAC address is a 48-bit address usually displayed in six groups of two hexadecimal digits, like `A4:B1:C2:D3:E4:F5`. The MAC address is divided into two main parts:

- **Organizationally Unique Identifiers (OUI)**: The first 24 bits (or first three bytes) of the MAC address, which indicate the manufacturer or organization that owns the NIC or device. `A4:B1:C2:D3:E4:F5`.
  - **Example of OUIs**:
    - Apple: 28:CF
    - Cisco: 00:0A
    - Samsung: 00:16:6C
- **Device Identifier (NIC-specific)**: The last 24 bits (or last three bytes), which are unique to each device from the same manufacturer. `A4:B1:C2:D3:E4:F5`

OSI Layer 3:

The **Network Layer** is responsible for routing data between devices across different networks, allowing communication beyond local network boundaries. It handles **logical addressing**, typically using IP addresses, to identify both the source and destination of each data packet. Layer 3 routes these packets from their source to their destination by determining the best path across interconnected networks. This layer also handles **packet forwarding**, **error handling**, **congestion control**, and **fragmentation** to ensure efficient delivery even over complex, multi-hop paths.

One of the most essential components of the Network Layer is the **IP**, which enables devices to be identified across multiple networks using unique IP addresses. As data moves through the Network Layer, it is encapsulated into *packets*, which contain not only the data itself - frame, but also the source and destination IP addresses. Routers, which operate at Layer 3, use this addressing information to direct packets to the appropriate networks and devices.

OSI Layer 4:

The **Transport Layer** (Layer 4) of the OSI model is responsible for providing end-to-end communication services for applications. It ensures that data is transferred between devices in a reliable or unreliable manner, depending on the protocol being used. The Transport Layer takes data from the **Application Layer (Layer 7)** and passes it down to the **Network Layer (Layer 3)** for routing.

Key Functions:

1. **Segmentation and Reassembly:** The Transport Layer breaks down large data from applications into smaller, manageable chunks called **segments** (in TCP) or **datagrams** (in UDP). When the data reaches the destination, it is reassembled in the correct order.

2. **Flow Control:** It controls the rate of data transmission to prevent congestion and ensure that the sender does not overwhelm the receiver. This is more prominent in protocols like TCP.
3. **Error Detection and Correction:** The Transport Layer provides mechanisms to detect errors in data during transmission. For example, TCP uses checksums to ensure data integrity. If errors are detected, it may request retransmission.
4. **Reliable Data Transfer:** The Transport Layer ensures reliable data transfer for connection-oriented protocols like TCP, which guarantees that data arrives correctly and in order, and offers retransmissions if necessary.
5. **Multiplexing:** The Transport Layer allows multiple applications or services on the same device to use the network simultaneously by differentiating traffic based on port numbers (e.g., HTTP uses port 80, FTP uses port 21)

Public IP: is a globally unique IP address assigned to a device that connects directly to the internet. These addresses are routable on the internet, meaning they can be accessed by any device across the globe. Public IPs are assigned by **ISPs (Internet Service Providers)** and are necessary for online communication outside a local network. They allow devices, such as web servers, email servers, or home routers, to communicate with any other device on the internet. Example of public IP addresses: `8.8.8.8` (Google DNS), `142.251.36.78` (Google's server).

Private IP: A private IP address is an IP address assigned to a device within a local network (like a home or corporate LAN) and is not routable on the internet. Devices with private IPs can only communicate within the same local network. These addresses are reserved by the **IANA (Internet Assigned Numbers Authority)** for private network use, meaning they can be used by anyone in any private network but are isolated from the global internet. Private IP addresses help conserve public IPs and add a layer of security to local networks. Private IP addresses are defined in **RFC 1918** and fall into the following ranges:

- 10.0.0.0 – 10.255.255.255 (10.0.0.0/8) - supports up to ~16 million addresses and is often used in large organizations.
- 172.16.0.0 – 172.31.255.255 (172.16.0.0/12) - supports about one million addresses, suitable for medium-sized networks.
- 192.168.0.0 – 192.168.255.255 (192.168.0.0/16) - supports ~65,000 addresses and is common in home and small office networks.

Internet Protocol (IP): is a Network Layer protocol (Layer 3) responsible for moving packets across interconnected networks, such as the internet, to reach a specific destination. IP handles **addressing** (each device has a unique IP address) and **routing** (determining the best path for data packets to travel across networks). IP on its own is **connectionless** (means IP does not establish a connection between the sender and receiver; it simply sends packets.) and **unreliable** (means IP doesn't guarantee that packets will arrive in the right order or even at all).Despite these limitations, IP's job is to ensure that each packet has an IP address and knows the route it needs to take. When paired with protocols like **TCP**, reliability and data integrity are added.

How it works:

1. **Packet Formation:** When data is ready to be sent from an application (e.g., loading a webpage), the data is first processed by the transport layer protocol (TCP or UDP) and broken down into segments.
2. **Encapsulation with an IP Header**:
    a. At the network layer, IP takes each segment and **adds an IP header**. This header contains critical information, including:
        i. **Source IP Address**: The IP address of the device sending the packet.
        ii. **Destination IP Address**: The IP address of the target device.
        iii. **Time-to-Live (TTL)**: A counter that decreases with each router hop, preventing packets from looping endlessly.

iv. **Protocol Field**: Indicates whether the packet is using TCP, UDP, or another transport layer protocol.
v. **Checksum**: A basic error-checking mechanism.
vi. **Total Length**: Indicates the packet's total size.

b. The IP header, combined with the data, forms an **IP packet**.

3. **Routing and Forwarding**:
   a. The IP packet is forwarded to a router, which reads the destination IP address in the header.
   b. The router determines the best path for the packet, sending it closer to the destination based on its routing table.
   c. Each router on the path reads the IP header, updates the TTL, and forwards the packet along its journey.
4. **Delivery**:
   a. When the IP packet reaches its destination network, the final router forwards it to the device with the destination IP address.
   b. The device's network layer processes the packet, removes the IP header, and passes the data up to the transport layer (TCP or UDP) for reassembly and error checking.

Transmission Control Protocol (TCP): is a transport layer protocol (Layer 4) that provides reliable, ordered, and error-checked delivery of data between applications. TCP is known as a connection-oriented protocol because it establishes a connection before sending data, ensuring that all data packets reach their destination correctly.

How it works:

1. **Connection Establishment (TCP Handshake):**
   a. TCP uses a **three-way handshake** to set up a connection between the sender and the receiver:
      i. **SYN:** The client sends a "synchronize" (SYN) packet to the server, requesting to establish a connection.
      ii. **SYN-ACK**: The server responds with a "synchronize-acknowledgment" (SYN-ACK) packet to confirm the request.
      iii. **ACK**: The client replies with an acknowledgment (ACK) packet, and the connection is established.

b. This handshake ensures that both devices are ready for data transfer, reducing the risk of dropped packets or connection issues.

2. **Data Transmission:**
    a. Once the connection is established, TCP **divides the data** into small, manageable units called **segments**.
    b. Each segment is assigned a **sequence number** so the receiving device can reorder segments if they arrive out of order.
    c. TCP attaches a **TCP header** to each segment, which includes:
        i. **Source and Destination Ports**: Used to identify the sending and receiving applications.
        ii. **Sequence Number**: Indicates the segment's position in the data stream.
        iii. **Acknowledgment Number**: Shows the next expected byte, allowing TCP to keep track of received data.
        iv. **Flags**: Includes control bits like SYN, ACK, and FIN, which help with connection management.
        v. **Window Size**: Used for flow control, indicating how much data the sender can send before waiting for an acknowledgment.
        vi. **Checksum**: Checks for errors in the segment’s data.
    d. The TCP header, combined with the segment, creates a **TCP packet**.
3. **Reliability Features**:
    a. **Acknowledgments**: TCP requires the receiver to send an acknowledgment for each segment received. If no acknowledgment is received within a certain time, TCP retransmits the segment.
    b. **Error Detection**: TCP checks for errors in each segment using the checksum, discarding any segment that doesn’t match the expected checksum.
    c. **Flow Control**: Using the window size, TCP prevents the sender from overwhelming the receiver with too much data.

d. **Congestion Control**: TCP adjusts the sending rate based on network congestion, slowing down when traffic is high and speeding up when traffic is low.

4. **Connection Termination**:
   a. Once all data has been transmitted, TCP closes the connection in a controlled manner:
      i. The client or server sends a **FIN** packet to indicate that no more data will be sent.
      ii. The other side responds with an ACK and sends its own FIN packet to confirm.
      iii. The initial side responds with a final ACK, and the connection is closed.

Example:

1. **Initial Data (Application Layer):** Suppose you're loading a webpage, and your computer sends an HTTP GET request. Here's a simplified version of what the initial data might look like:

```
GET /index.html HTTP/1.1
Host: example.com
```

2. **TCP Segment (Transport Layer):** When this data reaches the transport layer, TCP takes over. It divides the data into segments (if it's large) and attaches a TCP header to each segment. For simplicity, let's assume this entire request fits into one segment. Here's what the TCP header might include:
   a. **Source Port**: 49152 (random port on your device)
   b. **Destination Port**: 80 (standard HTTP port on the server)
   c. **Sequence Number**: 1 (the starting sequence for this connection)
   d. **Acknowledgment Number**: 0 (not used at the start)
   e. **Flags**: SYN (for establishing the connection in this example)
   f. **Window Size**: 1024 bytes
   g. **Checksum**: Used for error-checking (simplified here)
   h. **Data**: HTTP GET request

```
[ TCP Header ]
Source Port: 49152
```

Destination Port: 80
Sequence Number: 1
Acknowledgment Number: 0
Flags: SYN
Window Size: 1024
Checksum: (checksum data)
Data: GET /index.html HTTP/1.1
Host: example.com

3. **IP Packet (Network Layer):** Now, the TCP segment is passed to the network layer, where IP adds its own header, forming an IP packet. Here's what the IP header might look like:
    a. **Source IP Address**: 192.168.1.10 (your computer's private IP)
    b. **Destination IP Address**: 93.184.216.34 (IP address of example.com server)
    c. **Version**: IPv4
    d. **Header Length**: 20 bytes
    e. **Time-to-Live (TTL)**: 64 (each router reduces this by 1)
    f. **Protocol**: TCP (to indicate the packet contains TCP data)
    g. **Checksum**: Used for error-checking of the IP header
    h. **Total Length**: Size of the entire packet, including both headers and data

[IP Header]
Source IP: 192.168.1.10
Destination IP: 93.184.216.34
Version: IPv4
Header Length: 20 bytes
TTL: 64
Protocol: TCP
Checksum: (checksum data)
Total Length: (total size of IP packet)

[TCP Header]
Source Port: 49152
Destination Port: 80
Sequence Number: 1

Acknowledgment Number: 0
Flags: SYN
Window Size: 1024
Checksum: (checksum data)

[Data]
GET /index.html HTTP/1.1
Host: example.com

User Datagram Protocol (UDP): is a transport layer protocol in the OSI model, similar to TCP, but with key differences. It is designed for fast, connectionless communication and does not guarantee reliable delivery, ordering, or error checking of the transmitted data. UDP is commonly used for applications where speed is more important than reliability, such as video streaming, online gaming, and DNS queries.

UDP Header:

- **Source Port**: Identifies the sender's port.
- **Destination Port**: Identifies the receiver's port.
- **Length**: The total length of the UDP header and the data.
- **Checksum**: Used for error-checking the header and data

Network Address Translation (NAT): is a technique used in networking to modify the IP addresses of packets as they pass through a router or firewall. Its primary purpose is to allow multiple devices on a local network (LAN) to share a single public IP address when accessing external networks, such as the internet.

How it works: When a device on a local network (say, your computer) wants to access a resource on the internet (like browsing a website), the router (or NAT device) intercepts the request and changes the source IP address in the packet from the local private IP address to the router's public IP address. When the response comes back from the website, the router will know which local device initiated the request (based on a translation table) and will send the data back to that device.

DHCP

OSI Layer 5:

The **Session Layer** (Layer 5) of the OSI model is responsible for managing and controlling the dialogue between two devices or applications. Its primary function is to establish, maintain, and terminate communication sessions between applications on different devices. This layer ensures that data exchange between systems is properly synchronized, organized, and secure.

Key Functions of the Session Layer:

1. **Session Establishment, Maintenance, and Termination**: The Session Layer is responsible for establishing, maintaining, and gracefully terminating sessions between two communicating devices. It manages the dialogue to ensure that communication is continuous and properly synchronized.
2. **Dialog Control**: The Session Layer controls the flow of data in the form of **half-duplex** (one-way communication at a time) or **full-duplex** (two-way communication simultaneously). It ensures that both sides of the conversation follow a coordinated sequence of communication.
3. **Synchronization**: The Session Layer uses checkpoints to allow the communication to resume at the point it was interrupted in case of a failure. For example, if a large file transfer is interrupted, the session layer can resume the transfer from the last successful checkpoint, instead of starting over.
4. **Session Recovery**: In case of network disruptions, the Session Layer can help recover data from the point of interruption using recovery methods, which improves the reliability of the communication.
5. **Token Management**: In some protocols, the Session Layer manages the use of tokens to ensure that only one session at a time can perform an action, preventing conflicts in shared resources.

Examples of Session Layer Protocols:

1. **NetBIOS (Network Basic Input/Output System)**: Used for session management on Windows-based networks. NetBIOS allows different applications on different systems to communicate with each other and manage their sessions.
2. **RPC (Remote Procedure Call)**: Used for communication between a client and a server over a network. RPC manages the session, ensuring the remote function call and response are completed.
3. **SMB (Server Message Block)**: A protocol that provides session management for file and printer sharing in Windows networks.

OSI Layer 6:

The **Presentation Layer** (Layer 6) of the OSI model is responsible for translating, encrypting, and compressing data for the application layer. It acts as a translator between the application layer (Layer 7) and the session layer (Layer 5), ensuring that the data sent and received by applications is in a format that is understandable and usable on both ends.

Key Functions:

1. **Data Translation**: The Presentation Layer is responsible for translating data between different formats. It ensures that the data sent by the application layer is in a standard format that can be understood by the receiving application. For example, it can translate data from EBCDIC (Extended Binary Coded Decimal Interchange Code) to ASCII (American Standard Code for Information Interchange), or from one character encoding to another.
2. **Data Compression**: The Presentation Layer can compress data to reduce its size for more efficient transmission over the network. Compression helps save bandwidth and reduces the time it takes to send data.
3. **Data Encryption/Decryption**: The Presentation Layer can encrypt data before it is transmitted, ensuring that the data is secure during transmission. Once the encrypted data reaches the destination, the Presentation Layer is also responsible for decrypting it to be used

by the application. This is particularly important in secure communications like online banking or email.

4. **Syntax and Semantics**: The Presentation Layer also deals with the syntax and semantics of the data, ensuring that the data format used in communication is understood correctly by both parties. For instance, it handles converting data into an appropriate format for different systems (e.g., converting a 32-bit integer from one system's endianness to another).

Data Formatting Methods:

1. **Character Encoding**:
    a. **ASCII**: A common character encoding for English text, which encodes characters as numbers (e.g., 'A' = 65).
    b. **UTF-8/UTF-16**: Unicode formats used to encode a wider variety of characters from different languages, including symbols, special characters, and emojis.
2. **Data Serialization**:
    a. **XML (eXtensible Markup Language)**: A flexible format used for storing and transmitting structured data. It is used in many web services and applications.
    b. **JSON (JavaScript Object Notation)**: A lightweight format for data exchange, often used in APIs for web applications.
3. **Data Compression**:
    a. **ZIP**: A file compression format used to reduce the size of files and folders for transmission.
    b. **GZIP**: Another file compression standard used in web communications, particularly for compressing HTTP responses.
4. **Data Encryption**:
    a. **SSL/TLS (Secure Sockets Layer / Transport Layer Security)**: Protocols used to secure data transmitted over the network, such as during HTTPS communication.

Famous Protocols:

- **SSL/TLS (Secure Sockets Layer / Transport Layer Security)**: SSL and its successor, TLS, provide encryption for secure communication over networks, especially for websites (HTTPS).

They ensure that data transmitted between the client and the server is secure and private.

- **MIME (Multipurpose Internet Mail Extensions)**: MIME is used to format and encode email messages so they can contain non-text elements like images, attachments, and audio. It's especially useful in email communications and extends the capabilities of traditional text-based email.
- **XDR (External Data Representation)**: XDR is a standard for data representation that ensures data can be transmitted between different systems regardless of architecture or endianness. It is used for communication between systems with different data representations.
- **JPEG (Joint Photographic Experts Group)**: JPEG is a widely used compression standard for images. The Presentation Layer may compress image data using JPEG before transmitting it across the network.
- **GIF (Graphics Interchange Format)**: GIF is another image format that includes data compression. The Presentation Layer can use GIF to handle image files for web communication or other multimedia data.

OSI Layer 7:

Application Layer, is the topmost layer in the OSI model. This layer is closest to the end user and directly interfaces with software applications. It serves as the doorway for network services, where end-user software communicates with the network.

At this layer, data is presented in a format that can be understood by the receiving system or application. It is responsible for providing the essential services and protocols that allow applications to communicate over a network. For example, when you use a web browser, email client, or FTP program, the Application Layer ensures the data gets formatted and transmitted in a way that the receiving end can understand.

Key Functions:

1. **Providing Network Services to Applications**: The Application Layer enables software applications to access the network. It

allows for communication and data exchange between applications across different systems and networks.

2. **Data Representation and Encoding**: It ensures that data is properly formatted for the receiving application. For example, it can handle text encoding (like converting data into ASCII or Unicode) to make sure that both sender and receiver interpret the data the same way.
3. **Facilitating User Interaction**: This layer is where users interact with networked services. Whether it's browsing the web, sending emails, or transferring files, all of these functions happen through the Application Layer.
4. **Protocol Management**: The Application Layer uses a variety of protocols to manage and deliver services to applications. These protocols define the rules for how data is transmitted and received.

Common Protocols :

1. **HTTP (Hypertext Transfer Protocol)**: Used for accessing web pages. When you visit a website, your browser uses HTTP to send a request for the webpage to the server, and the server sends back the webpage content.
2. **FTP (File Transfer Protocol)**: Used for transferring files over a network. FTP allows users to upload and download files between computers or servers.
3. **SMTP (Simple Mail Transfer Protocol)**: Used for sending emails from one server to another. SMTP is responsible for the delivery of outgoing mail.
4. **DNS (Domain Name System)**: Translates human-readable domain names (like "www.example.com") into IP addresses so computers can locate each other on the network.
5. **IMAP (Internet Message Access Protocol) and POP3 (Post Office Protocol)**: These protocols allow email clients to retrieve emails from a server.
6. **SSH (Secure Shell)**: Used for secure remote login and administration of remote machines. SSH encrypts the communication between the client and server.