Introduction to Network Topologies

Contents

[Introduction to Computer Networks 2](#_Toc185193437)

[Protocols and RFCs 3](#_Toc185193438)

[Static vs Dynamic IP Addresses 7](#_Toc185193439)

[Layered Architecture and Network Communicati0on 10](#_Toc185193440)

[Some Points 12](#_Toc185193441)

[Sending a letter through a courier 13](#_Toc185193442)

[Internet Layered Architecture (OSI Model) 15](#_Toc185193443)

[1. Application Layer 18](#_Toc185193444)

[2. Presentation Layer 19](#_Toc185193445)

[3. Session Layer: 20](#_Toc185193446)

[4. Transport Layer: 20](#_Toc185193447)

[5. Network Layer: 22](#_Toc185193448)

[6. Data Link Layer: 23](#_Toc185193449)

[7. Physical Layer: 24](#_Toc185193450)

[OSI and TCP/IP Models 25](#_Toc185193451)

[Transition to the TCP/IP Model: 26](#_Toc185193452)

## Introduction to Computer Networks

1. **What is a Network?**
   * A network is a group of interconnected items.
     + *Examples*: Social network (people), Highway network (roads), Railway network (tracks).
   * In a network, the *items* and the *medium* connecting them are essential.
2. **What is a Computer Network?**
   * A computer network is a group of interconnected computers.
   * **Why do we create computer networks?**
     + **Communication**: Share information or data (emails, HTTP, etc.).
     + **Resource Sharing**: Access shared resources such as CPU, RAM, disk space, or cloud computing platforms (e.g., AWS, Google Cloud).
3. **Internet: The Largest Computer Network**
   * Internet = A network of interconnected computers using the *Internet Protocol (IP)* for communication.
4. **The Need for Protocols**
   * **Scenario**: Imagine 4 people in a locked room speaking different languages (e.g., Chinese, Hindi, Russian, Korean).
   * Problem: Without a common language, they cannot communicate effectively.
   * **Solution**: A **protocol** acts as a *common language* that standardizes communication between systems.
   * **Protocol**: A set of rules that enables devices to communicate efficiently in a network.

## Protocols and RFCs

**Introduction to Protocols**

* **Protocols** are rules that define how communication happens between devices on a network.
* Examples of protocols:
  + **HTTP**: Hypertext Transfer Protocol
  + **TCP**: Transmission Control Protocol
  + **UDP**: User Datagram Protocol
  + **IP**: Internet Protocol

**Who Defines Protocols?**

* **Protocols** are publicly available and **not defined by one person or authority**.
* Anyone can create a protocol, but it must be accepted and adopted for usage by others.
* **Protocols** are published as **RFCs**.

**IETF: Internet Engineering Task Force**

* IETF is a **nonprofit, volunteer-driven organization** that manages and maintains the Internet.
* Companies like **Google**, **Microsoft**, etc., contribute as members.
* Protocols are stored and published on the **IETF website**.

**What is an RFC?**

* **RFC** stands for **Request For Comments**.
* An RFC is a detailed proposal/document that:
  + Defines a protocol.
  + Describes the structure, benefits, and limitations of the protocol.
  + Explains how it works and how others can use/implement it.
* **Process** of creating a protocol:
  1. A person or group writes an RFC.
  2. The RFC is published on the IETF website.
  3. It is reviewed, debated, and improved by other experts.
  4. Once widely accepted, the RFC becomes a **standard protocol**.

**Key Examples of Protocols**

1. **HTTP**: Hypertext Transfer Protocol
   * Developed by **Tim Berners-Lee**, the Father of the World Wide Web.
   * Multiple versions exist:
     + **HTTP 1.1**: Most widely used version today.
     + **HTTP 2**: Standard but less common.
     + **HTTP 3**: Still in development (proposed state).
   * The protocol is defined in RFC documents:
     + Example: HTTP 1.1 has a **60-page** detailed RFC explaining all aspects.
       - **GET, HEAD, POST** methods.
       - HTTP **response codes** (e.g., **200 OK**, **404 Not Found**, etc.).
2. **IMAP**: Internet Message Access Protocol
   * Used for managing email.
3. **POP3**: Post Office Protocol (version 3)
   * Also used for email management.
4. **LDAP**: Lightweight Directory Access Protocol
   * Used for authentication and directory services.

**Conclusion**

* **Protocols** are the foundation of Internet communication.
* They are defined, reviewed, and standardized through the **IETF** process.
* RFCs ensure transparency, participation, and proper documentation for protocol development.

**Key Terms**

1. **Protocol**: A set of rules for communication.
2. **IETF**: Internet Engineering Task Force.
3. **RFC**: Request For Comments – a detailed document proposing a protocol.
4. **HTTP**: Hypertext Transfer Protocol.
5. **IMAP**: Internet Message Access Protocol.
6. **POP3**: Post Office Protocol version 3.
7. **LDAP**: Lightweight Directory Access Protocol.

**Computer Networks Analogy**

1. **Humans Communication Analogy**
   * Person 1 wants to communicate with Person 2 → They need an **address** for sending letters.
   * The **postman** delivers the letter using the **address**.
2. **Internet Analogy**
   * Your computer (Person 1) wants to send/receive data to/from scaler.com server (Person 2).
   * The **Internet service provider** (ISP) and mediums like **wires** facilitate communication.
   * Computers also need an **address** to locate and deliver data effectively.

**IP Address**

1. **What is an IP Address?**
   * IP = **Internet Protocol**.
   * It is a unique **address** assigned to every computer or device connected to the Internet.
   * **Computers understand numbers**, so IP addresses are represented in numbers.
2. **Versions of IP Address**
   * **IPv4** (Internet Protocol Version 4):
     + A **32-bit** address written in the form: A.B.C.D
     + Each A, B, C, and D ranges from 0 to 255.
     + Each part requires **8 bits** → Total: **32 bits**.
     + **Example**: 192.168.0.1
     + Total unique addresses = **2³² =**
   * **IPv6** (Internet Protocol Version 6):
     + A **128-bit** address designed to solve IPv4 limitations.
     + Much larger address space → Almost infinite.

**Why IPv4 Is Still Used?**

* **Problem**: There are **more devices** today than IPv4 can support (4 billion addresses).
* **Solution**: IPv4 is still used with **NAT (Network Address Translator)**.

**What is NAT?**

* NAT allows **multiple devices** in a private network to share a single public IP address.
* This makes IPv4 addresses reusable and extends their usability despite the shortage.

**Devices Connected to the Internet**

* **Computers** are not just PCs.
* Examples of devices connected to the Internet:
  + Laptops
  + Mobile phones
  + Smartwatches
  + Smart TVs
  + Smart bulbs
  + Routers
* **Every household** may have **40–50 devices** connected to the Internet.

**Why IPv6 Is Necessary?**

* IPv4 has address limitations → **4 billion addresses** are insufficient.
* IPv6 provides a much larger address space (128 bits) to support the growing number of devices.

**Summary**

1. **IP Address** is the unique address of every device on the Internet.
2. **IPv4** is a 32-bit address but has limitations due to a shortage of addresses.
3. **NAT** helps IPv4 still function despite these limitations.
4. **IPv6** solves the issue by offering a massive address space.

## Static vs Dynamic IP Addresses

**1. What is an IP Address?**

* IP Address = **Internet Protocol Address**.
* It is a unique identifier for devices on the Internet (like a house address for computers).
* Assigned by **Internet Service Providers (ISPs)**.

**2. Types of IP Addresses**

* **Static IP Address**
  + An IP address that **does not change**.
  + Useful for hosting servers (e.g., google.com, personal home servers).
  + Costs **money** as ISPs or cloud providers (e.g., AWS) must pay for these addresses.
  + Example: AWS charges for static IPs.
* **Dynamic IP Address**
  + An IP address that **changes each time** you connect to the Internet.
  + Provided by ISPs **for free**.
  + Example: Today’s IP = A, next connection = B, and so on.

**Why Static IP Costs Money**

1. Static IPs are **purchased** from global governing bodies like **ICANN** (Internet Corporation for Assigned Names and Numbers).
2. ISPs (e.g., Jio, Airtel, AWS) buy IP addresses from ICANN.
3. ISPs charge users extra for static IPs since they **paid to obtain them**.
4. In India, getting a static IP address costs around **₹4000–5000 per month** (as of 3 years ago).

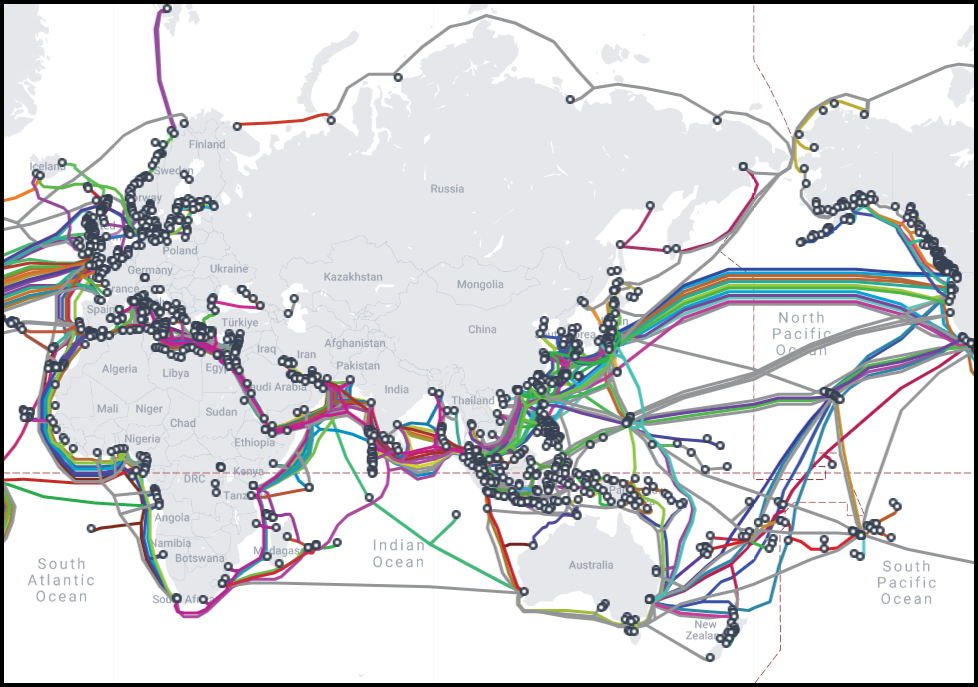
**Hosting a Server at Home**

1. A server needs a **fixed (static) IP address** so that clients can access it consistently.
2. With a **dynamic IP**, the address keeps changing → making it **impossible** for users to access the server reliably.

**Internet Infrastructure**

**1. How Data Travels Across the Internet**

* When you access a website (e.g., google.com), your request travels from **India to the US** (or wherever the Google server is).
* The data travels through **undersea cables** that connect continents.



A close-up of a cable

Description automatically generated**2. Undersea Cables**

* **Massive cables** lie under oceans to facilitate Internet communication.
* Examples of Indian undersea cable points:
  + **Mundra, Mumbai, Cochin, Chennai, Trivandrum, Vizag, etc.**
* Cables are owned and maintained by private companies (e.g., **Submarine Cable Networks**).

**Why Internet Costs Money**

1. ISPs like **Jio** have to pay owners of undersea cables to send data internationally.
2. **Satellites** are slow and cannot replace undersea cables for high-speed Internet.

**Geopolitical Challenges**

* **Internet towers or cables** cannot rely on other countries (e.g., Pakistan, China) because:
  + Cables can be **cut off** during conflicts.
  + Countries could control or disrupt the flow of Internet data.

**Reliance and Internet Costs**

1. Companies like **Reliance Jio** are building their own undersea cables.
2. Owning cables reduces dependency and costs, making Internet **cheaper** for users.

**Key Takeaways**

1. Static IPs are **fixed addresses** needed for servers and cost money.
2. Dynamic IPs are **free** but change periodically.
3. Undersea cables are critical for global Internet connectivity.
4. ISPs pay for infrastructure, leading to Internet costs for users.
5. Geopolitical concerns prevent Internet reliance on neighbouring countries.

## Layered Architecture and Network Communicati0on

**Key Concepts Discussed**

1. **Need for Addressing and Protocols**
   * Every computer connected to the internet must:
     + Follow a protocol for communication.
     + Have a unique address (IP Address) to send and receive data.
2. **Analogy of Addressing: Hostel/Flat Example**
   * **Flat/Hostel Address**: Similar to an **IP Address**.
   * **Room Number**: Similar to a **Port Number**.
     + The IP address is for identifying a computer (host).
     + The port number identifies which **application** or **process** on the computer should receive the data.
   * Gatekeeper analogy: The operating system acts as a "gatekeeper," determining where data goes within the computer.
3. **Applications and Network Communication**
   * Multiple applications (e.g., Chrome, Zoom, WhatsApp) can communicate with the network simultaneously.
   * **Challenge**: How does a computer know which application should receive the incoming data?
     + **Solution**: Port numbers are used as unique identifiers for applications.

**IP Address and Port Numbers**

* **IP Address**:
  + Identifies a computer on the network.
  + Example: 192.168.1.1.
* **Port Number**:
  + A **16-bit identifier** used to identify an application.
  + Range: **0 to 65,535** (65,536 ports).
  + Example: 192.168.1.1:3000 means the data is sent to:
    - **IP Address**: 192.168.1.1.
    - **Port**: 3000 (application bound to this port).

**How Data is Handled by the Operating System**

1. Data first reaches the **operating system** (OS), which acts as the "security guard."
2. The OS checks the **port number** of the incoming data.
3. The OS maintains an **internal mapping table** of port numbers to applications/processes.
   * Example:
     + Port 82: Application XYZ.
     + Port 1234: Google Chrome.
     + Port 5678: Another Google Chrome tab.
4. OS forwards the data to the correct application based on the port number.

**Port Number Usage**

* A single process/application can have **multiple ports**.
* **Default Ports**:
  + **HTTP**: Port 80.
  + **HTTPS**: Port 443.
  + **FTP**: Port 21.
  + **Express.js**: Port 3000 (common default).
  + **Spring Boot**: Port 8000 (configurable).

**Note**: Default ports are **protocol-specific** but can be changed by the developer.

**Summary of Learning**

1. **Data Communication**:
   * Requires both an **IP Address** (computer identification) and a **Port Number** (application identification).
   * Format: IP Address: Port Number.
2. **Role of Operating System**:
   * Acts as the "gatekeeper" to forward network data to the correct application/process using port mapping.
3. **Port Numbers**:
   * 16-bit identifiers (range: 0–65,535).
   * Allow multiple applications to communicate over the network.
4. **Socket Programming**:
   * Detailed handling of ports and connections will be covered in a future **Socket Programming** class.

**Key Clarifications**

* There is no universal "default" port like 8080. Default ports depend on the protocol:
  + Example: HTTP (80), HTTPS (443).
* Developers can configure ports for applications like Express.js and Spring Boot.

## Some Points

1. **What is the Internet?**
   * A network of computers that exists for communication and resource sharing.
2. **IP Addresses, Ports, and Protocols:**
   * **IP Address:** Identifies a device on a network.
   * **Port:** Allows multiple applications on a single device to communicate simultaneously.
   * **Protocols (e.g., HTTP/HTTPS):** Define the rules of communication.
3. **Default Ports:**
   * **HTTP:** Default port 80.
   * **HTTPS:** Default port 443.
   * If no port is specified, the browser assumes these defaults for the respective protocol.
4. **Multiple Applications on a Server:**
   * Each application on a server must run on a **unique port**.
   * If an application doesn’t use the default port (e.g., 443 for HTTPS), the port must be explicitly mentioned (e.g., example.com:8080).
5. **Domain Names and Ports:**
   * Domain registration doesn’t include port numbers.
   * The operating system binds applications to ports when they run.
6. **How Does the Browser Know the Port?**
   * If the protocol is **HTTPS**, the browser defaults to port 443.
   * If an application runs on a non-default port, the port number must be explicitly specified in the URL.
7. **Key Takeaways:**
   * One server can host multiple applications, but each must have a different port.
   * A single port (e.g., 443) cannot serve multiple applications simultaneously.
   * Websites like google.com or facebook.com appear without a port because they use **default ports** for HTTP/HTTPS.

## Sending a letter through a courier

1. **Layered steps** — Postman → Post Office → Transport (like an airplane) → Intermediate stops → Final delivery.

A drawing of a plane and a truck

Description automatically generated

1. **Adding/removing information at each layer** — Envelopes, stamps, reference numbers are added, and the corresponding layer at the receiver’s side processes/removes this information.
2. **Reversal of steps** — The exact layers that added information at the sender side *remove* it at the receiver side.
3. **Independent tasks for each layer** — A layer only communicates with the corresponding layer at the destination.

Now you’re transitioning to:

* How **applications** use the **Internet** via layered protocols.
* How **each layer adds information** for its counterpart layer on the receiving side.
* Key factors for choosing services, such as **price, cost, time**, relate to **quality of service (QoS)** in networking.

To continue and link it back to **Internet architecture**:

1. **What is the Internet Layering Model?**
   * Introduce the **OSI Model** and the **TCP/IP Model**.
2. **Break it down into layers**:
   * Physical Layer → Data Link Layer → Network Layer → Transport Layer → Application Layer.
3. Relate each OSI Layer to your postal analogy:
   * Text = Data (Application Layer)
   * Envelope = Header (Transport Layer)
   * Stamp = IP Address (Network Layer)
   * Airplane = Physical Network Transmission.

## Internet Layered Architecture (OSI Model)

**Overview:**

* **Internet works on a layering architecture**.
* Even before the Internet came into existence, researchers predicted that the Internet could be structured into multiple **layers**.
* Each layer would have specific **responsibilities** and work independently.
* Layers interact by **passing work** to the next layer.

**Advantages of Layering Architecture:**

1. **Clarity of Responsibilities**:
   * Each layer has a **well-defined role**.
   * Example (postal analogy):
     + Sender → Postman → Post Office → Transport (airplane) → Receiver.
     + Each step does its part and passes the work to the next.
2. **Independence of Layers**:
   * A layer does its work and passes control to the next layer **without worrying** about how the next layer operates.
   * Example: Postman delivers to the post office and doesn’t care about how the post office sends the letter.
3. **Multiple Choices at Each Layer**:
   * Each layer can have **multiple options** for completing its work.
   * Example: A post office can choose from multiple airlines depending on requirements (cost, speed, etc.).
   * Similarly, each layer in the Internet chooses a service based on its needs.

**OSI Model (Open Systems Interconnection):**

* **Proposed before the Internet** was invented.
* The OSI model defines **7 layers** for structuring the Internet.
* **Purpose**: Simplifies understanding of how data is transmitted across the Internet.
* Layers go from **top to bottom**.

**The 7 Layers of the OSI Model:**

1. **Application Layer**:
   * **Topmost Layer**.
   * This is the layer **users interact** with directly.
   * Examples: Web browsers, email clients, etc.
   * **Gateway to the Internet** for users.
2. **Presentation Layer**:
   * Handles **data formatting, encryption, and compression**.
   * Ensures data is presented in a readable and usable form.
3. **Session Layer**:
   * Manages **sessions** between applications.
   * Responsible for opening, maintaining, and closing communication sessions.
4. **Transport Layer**:
   * Ensures **end-to-end communication**.
   * Provides reliable data transfer, error correction, and flow control.
   * Example: TCP (Transmission Control Protocol).
5. **Network Layer**:
   * Handles **routing and addressing**.
   * Determines the best path for data to travel across the network.
   * Example: IP (Internet Protocol).
6. **Data Link Layer**:
   * Manages communication **between two directly connected nodes**.
   * Handles error detection and correction for data transfer.
   * Example: Ethernet protocol.
7. **Physical Layer**:
   * **Lowest Layer**.
   * Deals with the **physical transmission** of data through wires, Wi-Fi, or other mediums.
   * Example: Cables, fibre optics, radio signals.

**Layer Interaction:**

* **Top Layer (Application Layer)**: The user’s data starts here.
* **Bottom Layer (Physical Layer)**: Data is physically transferred via wires, Wi-Fi, or wireless mediums.
* Each layer adds its own **information** (headers, metadata) and passes it down to the next layer.
* At the receiver’s end, the layers **reverse the process**, removing the information added by each corresponding layer.

**Key Points:**

* **Application Layer**: User-facing, data input/output.
* **Physical Layer**: Actual transmission of data.
* Layers operate **independently**, but work together to ensure smooth communication.
* **Advantages**: Clarity, modularity, and flexibility in choosing services at each layer.

**Summary:**

The OSI model is a layered framework that defines how Internet communication works:

* **7 layers**, each with clear responsibilities.
* Layers work **independently** but interact by passing work to the next.
* Example of postal service helps simplify understanding:
  + Each step/layer has its own responsibility and does not interfere with others.

# Application Layer

**Definition:**

The **Application Layer** is the **topmost layer** of the OSI model. It is the layer that **directly interacts with users** or user applications. It is responsible for:

1. **Creating data** to send.
2. **Receiving data** sent to the user.

**Key Points:**

1. **Communication Start/End**:
   * The **communication starts** at the application layer and **ends** at the application layer on the receiver's side.
   * Both sender and receiver are interacting through applications.
2. **Types of Applications**:
   * **Front-end Applications**: User-facing apps like browsers or chat apps.
   * **Back-end Applications**: Server-side apps like Spring Boot, Django, or .NET applications. Even though the user does not directly interact with these, they are still part of the application layer.
3. **Examples**:
   * When you visit google.com:
     + Your browser (application) sends a **request**.
     + Google’s server (another application) **responds** to that request.
4. **Protocols at Application Layer**:
   * Protocols used by the user or applications for communication.
   * Examples:
     + **HTTP/HTTPS**: Used for web browsing.
     + **FTP**: Used for file transfers.
     + **SMTP**: Used for sending emails.
     + **IMAP/POP**: Used for receiving emails.
     + **DNS**: Resolves domain names to IP addresses.
5. **Summary**:
   * **Creates or receives data**.
   * **Interacts directly with the user or application**.
   * **Provides services through protocols like HTTP, FTP, SMTP, etc.**

# Presentation Layer

**Definition:**

The **Presentation Layer** is responsible for **transforming the data** from the application layer into a form suitable for transmission, and vice versa.

**Key Points:**

1. **Purpose**:
   * Transforms data **into a format** suitable for transmission.
   * Ensures data is **understandable** by the receiving application.
2. **Tasks Performed**:
   * **Compression**: Reduces data size before sending it.
   * **Encryption**: Secures the data for transmission (e.g., password encryption).
   * **Encoding/Decoding**: Ensures compatibility with applications.
3. **Receiver Side**:
   * Opposite operations are performed:
     + **Decompression**
     + **Decryption**
     + **Decoding**
4. **Example**:
   * If a 1GB file is sent, it might be **compressed** before transmission and **decompressed** upon receiving.
   * If a password is **encrypted** before transmission, it is **decrypted** at the receiver end.
5. **Summary**:
   * Transforms data for **efficient transmission** (e.g., compresses, encrypts).
   * Converts received data into a format the application can understand.

# Session Layer:

* **Definition**: Manages user sessions, ensuring consistent and reliable communication between devices.
* **Purpose**:
  + Establishes, maintains, and terminates communication sessions.
  + Tracks and manages data exchange in long-running interactions.
* **Key Functions**:
  + **Session Establishment**: Initiates a session between the communicating devices.
  + **Session Maintenance**: Keeps track of the session state and user authentication (e.g., login persistence).
  + **Session Termination**: Ends the session when communication is complete.
* **Example Scenarios**:
  + Maintaining user login status on a website.
  + Ensuring smooth navigation without requiring repeated authentication.

**Note**: In modern implementations, the session layer’s responsibilities are often integrated into the application layer.

# Transport Layer:

* **Definition**: Acts as the middleman, ensuring reliable data transfer between systems.
* **Purpose**:
  + Breaks down data into manageable chunks for transmission.
  + Provides guarantees on data delivery, order, and integrity.
* **Key Functions**:
  1. **Segmentation**:
     + Divides data into smaller units (chunks) for transmission.
     + Chunks are called "segments" (in TCP) or "datagrams" (in UDP).
  2. **Error Control**:
     + Detects and corrects errors in transmitted data.
     + Resends data if errors occur.
  3. **Delivery Guarantees**:
     + Ensures data is delivered correctly and in order.
     + Provides reliability through retransmissions if necessary.
  4. **Flow Control**:
     + Prevents overwhelming the receiver by controlling data flow.
* **Transport Protocols**:
  + **TCP (Transmission Control Protocol)**:
    - Reliable, connection-oriented protocol.
    - Ensures ordered and error-free data delivery.
  + **UDP (User Datagram Protocol)**:
    - Unreliable, connectionless protocol.
    - Faster but does not guarantee delivery or order.

**Example Scenarios**:

* Sending a large file (split into smaller chunks) over the internet.
* Guaranteeing correct order of video frames in a live stream.

**Key Analogies:**

* **Transport Layer as a Mover and Packer**:
  + Breaks the “house” (data) into smaller packages (chunks).
  + Manages transportation, ensures delivery, and handles errors.

# Network Layer:

* **Definition**: Responsible for ensuring data delivery from source to destination across multiple networks.
* **Purpose**:
  + Handles end-to-end communication across diverse network paths.
  + Finds the optimal route for data packets to travel.
* **Key Functions**:
  1. **Routing**:
     + Determines the best path for data to travel through various network nodes.
     + Manages routers and paths (e.g., Delhi → Ahmedabad → Pune → Mumbai).
  2. **Packet Handling**:
     + Divides data into packets for transmission.
     + Ensures packets are sent to the correct destination.
  3. **Protocol Support**:
     + Uses the Internet Protocol (IP) as its primary protocol.
* **Characteristics**:
  + Does not guarantee data integrity or order.
  + Relies on the transport layer for error detection and correction.

**Key Term**:

* **Packets**: Data chunks at the network layer, distinct from transport layer segments/datagrams.

**Example Scenarios**:

* Sending data through multiple routers to reach a destination.
* Managing IP addresses for communication.

# Data Link Layer:

* **Definition**: Handles communication between directly connected devices, ensuring reliable transfer within a single "hop" in the network.
* **Purpose**:
  + Manages data transfer across a single link (e.g., from a computer to a router).
  + Detects and corrects errors at the hop level.
* **Key Functions**:
  1. **Error Detection and Correction**:
     + Detects errors during data transmission.
     + Resends data to ensure integrity within the hop.
  2. **Frame Handling**:
     + Divides data into frames for transmission across the physical medium.
     + Reassembles frames upon receipt.
  3. **Medium Access Control (MAC)**:
     + Manages how devices share access to the physical medium.
  4. **Hop-by-Hop Communication**:
     + Ensures data reliability for each hop before passing it to the next.
* **Characteristics**:
  + Operates between adjacent devices in a network.
  + Handles communication within a single hop rather than end-to-end.

**Key Term**:

* **Frames**: Data chunks at the data link layer, distinct from packets at the network layer.

**Example Scenarios**:

* Transmitting data from a laptop to a router via Wi-Fi.
* Error correction during the transfer between two directly connected devices.

# Physical Layer:

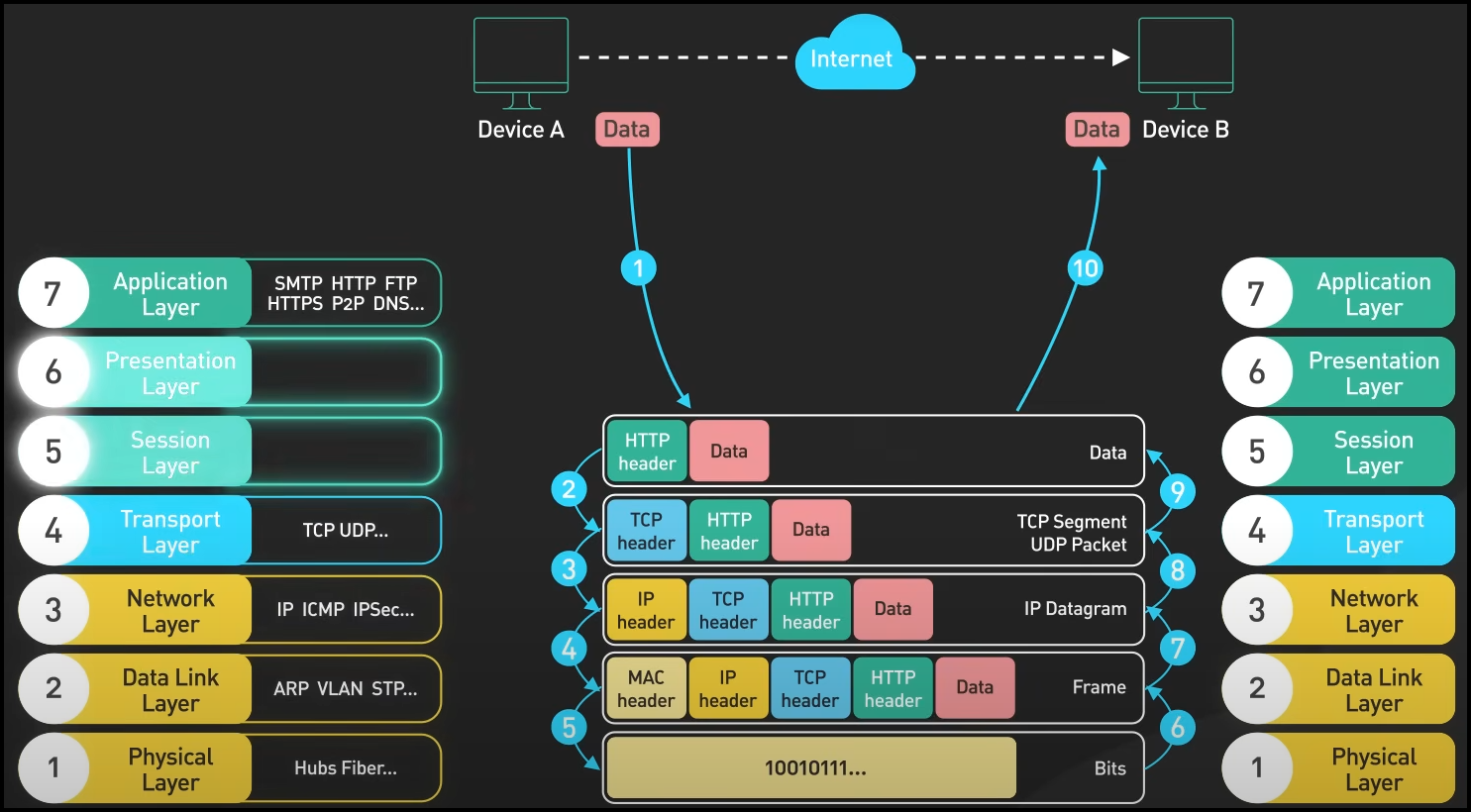
* **Definition**: The hardware-oriented, lowest layer of the OSI model, responsible for the physical transmission of data.
* **Purpose**:
  + Acts as the medium through which data is transmitted as raw bits (0s and 1s).
  + Converts data into signals (electrical, optical, or radio) for transmission.
* **Key Functions**:
  1. **Bit Transmission**:
     + Converts data into a stream of bits for transmission over the physical medium.
     + Sends and receives these bits at the hardware level.
  2. **Physical Medium Management**:
     + Defines the specifications for cables, connectors, and wireless technologies.
  3. **Signal Encoding**:
     + Determines how bits are encoded into physical signals.
  4. **Synchronization**:
     + Ensures sender and receiver are synchronized for accurate data transmission.
* **Characteristics**:
  + Deals with the actual hardware and medium.
  + Includes devices like Ethernet cables, Wi-Fi transmitters, fiber optics, etc.

**Example Scenarios**:

* Transmitting data through optical fiber cables.
* Sending bits wirelessly using radio waves (e.g., Wi-Fi, Bluetooth).

**Summary:**

* **Application Layer**: Interacts directly with the user, generates/receives data.
* **Presentation Layer**: Formats data for transmission or interpretation.
* **Session Layer**: Manages and tracks communication sessions.
* **Transport Layer**: Ensures reliable data delivery, segmentation, and integrity.
* **Network Layer**: Handles routing, end-to-end delivery, and packet management for source-to-destination communication.
* **Data Link Layer**: Ensures reliable communication and error correction within a single hop in the network.
* **Physical Layer**: Handles the actual transmission of bits through physical hardware.



**Note: The OSI model is a theoretical framework. Many of these layers are merged or implemented differently in the practical TCP/IP model used in modern networking.**

## OSI and TCP/IP Models

**The OSI Model:**

* **Definition**: A theoretical framework proposed before the Internet's existence, designed to conceptualize a seven-layered architecture for network communication.
* **Purpose**: Describes a layered approach to data exchange, helping developers understand networking fundamentals.
* **Layers**:
  1. **Application**: Interfaces with the user, handles application-level data.
  2. **Presentation**: Manages data formatting, encryption, compression.
  3. **Session**: Tracks, maintains, and terminates communication sessions.
  4. **Transport**: Ensures reliable data transfer between devices.
  5. **Network**: Handles routing and packet transmission between networks.
  6. **Data Link**: Ensures hop-to-hop reliability and frame delivery.
  7. **Physical**: Transmits raw bits over hardware media.

**Key Points**:

* Upper three layers (Application, Presentation, Session) typically involve developer-written logic.
* Lower layers (Transport, Network, Data Link, Physical) are managed by the operating system.

## Transition to the TCP/IP Model:

* **Definition**: A practical, simplified model representing how the Internet operates today.
* **Changes**:
  + Upper three OSI layers collapsed into a single **Application Layer** in TCP/IP.
  + Responsibilities like compression, encoding, encryption, and authentication shifted to developers within the Application Layer.
* **Structure**:
  + **Application Layer**: Handles all user-facing logic and data processing.
  + **Transport Layer**: Ensures reliable/unreliable communication (TCP/UDP).
  + **Internet Layer**: Manages IP addressing and routing.
  + **Network Interface Layer**: Combines Data Link and Physical layers for hardware-level transmission.

**Rationale**:

* Theoretical OSI models served as guidance; practical Internet architecture adapted for efficiency and developer control.

**Why Understanding OSI/TCP-IP Matters:**

1. **For Developers**:
   * Critical for tasks like socket programming and protocol selection (TCP vs. UDP).
   * Application layer now bears responsibilities like encryption, session management, and data compression.
2. **For System Design**:
   * Helps evaluate trade-offs in reliability, speed, and resource utilization.
   * Supports scalability by grounding applications in networking principles.

**Developer Responsibilities in Modern Networking:**

* **Compression & Encoding**: Developers choose or implement methods for data compression and encoding (e.g., gzip for HTTP).
* **Authentication**: Application logic (e.g., JWT tokens, cookie-based sessions) ensures secure access.
* **Protocol Decisions**: Developers specify transport protocols (TCP/UDP) for their application's needs.

**Conclusion:**

* The OSI model provides a foundational understanding of networking layers.
* The TCP/IP model is the current, practical implementation.
* Developers must adapt to the shift of certain OSI-layer responsibilities into the Application Layer, requiring deeper knowledge of networking intricacies.