UNIT – 2.1

🌐 Introduction to Application Layer

- It is the **topmost layer of the OSI model** — the one **closest to the user**.
- It provides **network services to applications** like web browsers, email, and chat apps.
- **Common protocols:** HTTP, FTP, SMTP, DNS, DHCP, POP3, Telnet, etc.
- It **doesn't send data itself**, but **prepares** it to be sent using lower layers.
- **Example:** Like a **waiter in a restaurant** – the user tells what they want, the waiter (Application Layer) communicates it to the kitchen (Network) and brings back the result.

💡 Importance of Application Layer Protocols

- **Enable user services:** Helps in web browsing (HTTP), email (SMTP, POP3), file transfer (FTP), etc.
- **Standardizes communication:** Defines how data should be formatted and sent so all systems understand each other.
- **Ensures interoperability:** Devices and apps from different companies can work together.

🧠 Principles of Network Applications

- A **network application** runs on one device and communicates with another device over a network.
- Example: A **web browser** (client) on your laptop talks to a **web server** (server) on the internet.
- Examples: Email, Web, Remote Login, P2P Sharing, Online Games, Video Streaming, Voice Chat, Social Media.

📐 Network Application Architectures

1. Client-Server Architecture

- **Server:** Always on, fixed IP, stores data, runs continuously.
- **Client:** Requests service, may have changing IP, not always connected.
- Example: Web Server (server) and Browser (client).

2. Peer-to-Peer (P2P) Architecture

- Devices (peers) **communicate directly** with each other.
- Each peer can **request and provide services**.
- **Self-scalable:** More peers = more capacity.
- **Example:** Torrent file sharing.

⚙️ Process and Socket

- **Process:** A running program.
- In the same computer → use **Inter-process communication (IPC)**.
- On different computers → communicate by **sending messages**.
- **Client process** starts communication; **Server process** waits.
- **Socket:** A software "door" between the process and the network.

Example:

- Each socket has:
    - **IP Address** – device location
    - **Port Number** – specific application
    - **Transport Protocol** – TCP or UDP

🚚 Transport Services for Applications

When building apps, we choose transport protocol based on the needs:

Service	Description
Reliable Data Transfer	Ensures no data loss (used in Email, Banking, FTP)
Throughput	Some apps need high data rate (video streaming)
Timing	Some apps need low delay (video call, games)
Security	Data is encrypted while sending and decrypted when received

---

🔗 Internet Transport Protocols

TCP (Transmission Control Protocol)

- **Connection-oriented** (setup first).

- **Reliable** and ordered data delivery.
- **Has congestion control**.
- Used in: Web, Email, File Transfer.

UDP (User Datagram Protocol)

- **Connectionless**, no setup.
- **Unreliable**, no guarantee of delivery.
- **No congestion control or security.**
- Used in: Video calls, Games.

🌍 Web & HTTP

- The **World Wide Web (WWW)** was invented by **Tim Berners-Lee** in 1994.
- It connects documents using **URLs** and **hyperlinks**.
- **Web page** = Base HTML file + other objects (images, audio, etc).
- Each object has its own **URL**.
- Example: www.someschool.edu/someDept/pic.gif

💬 HTTP (HyperText Transfer Protocol)

- It's an **Application Layer protocol** for web communication.
- Two programs use it:
    - **Client:** Browser (Chrome, Edge)
    - **Server:** Web server (Apache)
- Uses **TCP port 80** for communication.
- HTTP is **stateless** (server doesn't remember past requests).
- Two types:
    - **Non-persistent HTTP (HTTP/1.0):** New connection for each object.
    - **Persistent HTTP (HTTP/1.1):** One connection for multiple objects (faster).

🕒 HTTP Response Time

- **RTT (Round Trip Time):** Time for a message to go to the server and come back.

- Non-persistent HTTP = 2 × RTT + file transmission time
- Persistent HTTP saves time by keeping the connection open.

✉️ HTTP Message Format

Request Message

- Sent by client to ask for a file.
- Has:
    - **Request line** (GET, POST)
    - **Header lines**
    - **Carriage return (end of headers)**

Example:

GET /index.html HTTP/1.1

Host: www.net.cs.umass.edu

User-Agent: Firefox

Response Message

- Sent by server to reply.
- Has:
    - **Status line** (protocol + status code)
    - **Header lines**
    - **Data (HTML file)**

Example:

HTTP/1.1 200 OK

Date: Sun, 26 Sep 2010

Server: Apache

Content-Type: text/html

Common Status Codes

Code	Meaning
200	OK

Code	Meaning
301	Moved Permanently
400	Bad Request
404	Not Found
505	HTTP Version Not Supported

🍪 Cookies

- A **small text file** stored on user's computer by websites.
- Helps website **remember user preferences and login info.**
- Cookie parts:
    1. Header line in HTTP request
    2. Header line in HTTP response
    3. Cookie file on user's system
    4. Database on website's server

🧰 Web Cache (Proxy Server)

- A **middle server** that stores copies of web pages.
- When a user requests a page:
    - If it's in the cache → sent directly (faster).
    - If not → fetched from the main server, stored in cache for next time.
- **Benefits:**
    - Reduces client response time.
    - Reduces Internet traffic.

✅ Summary

- Application Layer helps apps communicate using protocols.
- Common ones: **HTTP, FTP, SMTP, DNS, DHCP, POP3**.
- It uses **client-server or P2P architecture**.

- **HTTP** enables web communication using request/response messages.
- **Cookies** remember users; **Cache** makes web faster.