## Week 2

### 1. Network Layering & TCP/IP Model

* Layering: Breaking communication into separate components that handle different parts of a network message.
* TCP/IP Model:

1. Application Layer – Encodes and decodes messages for applications (e.g., web browsers, email clients).
2. Transport Layer – Breaks messages into smaller packets, and numbers them for reassembly.
3. Network Layer – Adds sender and recipient addresses (IP addresses).
4. Link Layer – Handles physical transfer of packets between nodes.

* Why Layering?
  + Standardizes communication.
  + Allows different applications to work over the same transport and network layers.
  + Enables software flexibility.

### 2. Application Layer Overview

The Application Layer includes:

* Web and HTTP
* Email protocols (SMTP, IMAP)
* DNS (Domain Name System)
* P2P (Peer-to-Peer) Applications
* Video Streaming & Content Delivery Networks (CDNs)
* Socket Programming (UDP/TCP)

### 3. Client-Server vs. Peer-to-Peer (P2P) Architectures

* Client-Server Model:
  + A server is always on and provides resources to clients.
  + Examples: Websites, email services.
* P2P Architecture:
  + No central server; all nodes act as both clients and servers.
  + Example: BitTorrent (file-sharing).

### 4. Communication Between Processes

* Process: A running program.
* Sockets: A communication interface that allows a process to send and receive messages.
* Addressing:
  + Uses IP address to identify a device.
  + Uses port numbers to identify specific services (e.g., HTTP = Port 80).

### 5. Web and HTTP (Hypertext Transfer Protocol)

* HTTP is the protocol used for web communication.
* Works using the client-server model:
  + Client (Browser) requests a web page.
  + Server (Web Server) responds with the requested content.
* Types of HTTP Connections
* Non-Persistent HTTP:
  + A separate connection is created for each object (e.g., images, CSS).
  + Slower due to multiple connections.
* Persistent HTTP (HTTP/1.1 and HTTP/2):
  + Keeps the connection open for multiple requests.
  + Faster and more efficient.
* HTTP Request Methods
  + GET – Requests data from the server.
  + POST – Sends data to the server.
  + HEAD – Requests only metadata, not content.
  + PUT – Uploads a file to the server.
* HTTP Response Codes
  + 200 OK – Successful request.
  + 301 Moved Permanently – URL has changed.
  + 404 Not Found – Page does not exist.
  + 500 Internal Server Error – Server issue.

### 6. Cookies and Web Caching

Cookies: Store user data to maintain state across HTTP requests.

Web Caching:

* A proxy server stores copies of frequently requested web pages.
* Reduces response time and network traffic.

### 7. Email & Mail Transfer Protocols

Email Components:

1. User Agent (Mail Client) – Composes and reads emails (e.g., Outlook).
2. Mail Server – Stores and forwards emails.
3. SMTP (Simple Mail Transfer Protocol) – Sends emails between servers.

Mail Retrieval Protocols:

* IMAP (Internet Mail Access Protocol) – Stores emails on a remote server.
* POP3 (Post Office Protocol 3) – Downloads emails to a local device.

### 8. Domain Name System (DNS)

DNS translates human-readable domain names (e.g., google.com) into IP addresses.

Hierarchy of DNS Servers:

1. Root DNS Servers – Directs queries to Top-Level Domain (TLD) servers.
2. TLD Servers – Manage domains like .com, .org, .edu.
3. Authoritative DNS Servers – Store actual IP address mappings.

DNS Query Types

* Recursive Query: The DNS server finds the answer for the client.
* Iterative Query: The DNS server provides a referral to another DNS server.

### 9. Peer-to-Peer (P2P) Networks

* No central server; nodes communicate directly.
* Advantages:
  + Self-scalable (new peers contribute resources).
  + Efficient for large-scale file distribution.
* Example: BitTorrent
  + Files are split into chunks.
  + Peers download chunks from multiple sources.

### 10. Video Streaming & Content Delivery Networks (CDNs)

* Challenges:
  + Variable bandwidth.
  + High demand for videos.
* Solution: DASH (Dynamic Adaptive Streaming over HTTP)
  + Client requests video chunks based on available bandwidth.
* CDNs (Content Delivery Networks):
  + Store copies of videos in multiple locations.
  + Reduce latency and improve performance.
  + Example: Netflix uses CDNs for streaming.

### 11. Socket Programming (UDP & TCP)

* UDP (User Datagram Protocol):
  + Fast but unreliable (used for live streaming).
* TCP (Transmission Control Protocol):
  + Reliable, ensures data is delivered correctly (used for emails, file transfers).
* Example TCP Server & Client
* TCP Server:

1. Listens for client connections.
2. Accepts connection.
3. Receives request.
4. Sends response.

* TCP Client:

1. Connects to the server.
2. Sends request.
3. Receives response.
4. Closes connection.

### 12. Summary

* Application Layer provides services like web browsing, email, DNS, and video streaming.
* Protocols like HTTP, SMTP, and DNS define communication rules.
* Client-server and P2P architectures support different network applications.
* Web caching and CDNs improve performance.