**Computer Networking**

**Ch 02**

**○ Principles of Network Applications**

* The **application layer** enables network applications to communicate using protocols.
* **Client-server architecture:** A server provides resources, and clients request them.
* **P2P architecture:** Peers communicate directly, distributing resources among users.
* **Processes and sockets:** Applications communicate over the network using sockets, which act as endpoints for data transfer.

**○ Web and HTTP (HyperText Transfer Protocol)**

* **HTTP** is a stateless, request-response protocol used for web browsing.
* **HTTP methods:** GET, POST, HEAD, PUT, DELETE.
* **Cookies** enable session tracking.
* Web caching (proxy servers) reduces latency by storing copies of frequently accessed content.

**○ Email and SMTP (Simple Mail Transfer Protocol)**

* Email uses three main protocols:
	+ **SMTP** (sending mail)
	+ **POP3** and **IMAP** (retrieving mail)
* **SMTP** follows a "push" model, while **POP3/IMAP** allow users to fetch emails.

**○ DNS (Domain Name System)**

* Maps human-readable domain names (e.g., google.com) to IP addresses.
* Hierarchical structure:
	+ Root DNS servers
	+ Top-Level Domain (TLD) servers (.com, .org, etc.)
	+ Authoritative DNS servers
* **Recursive vs. Iterative queries:** Clients request resolution from local DNS servers, which may forward queries.

**○ Peer-to-Peer (P2P) Applications**

* Examples: File sharing (BitTorrent), VoIP (Skype).
* No central server; peers share resources directly.
* Challenges include scalability, security, and maintaining network topology.

**○ Video Streaming and Content Distribution**

* **Adaptive streaming:** Adjusts video quality based on network conditions.
* **CDNs (Content Delivery Networks):** Distribute content across multiple locations to improve access speed.
* **Dash Protocol (Dynamic Adaptive Streaming over HTTP):** Segments video for efficient streaming.

**○ Socket Programming**

* Network applications use **sockets** for communication.
* **TCP sockets** (reliable, connection-oriented)
* **UDP sockets** (faster, connectionless).
* Simple client-server interaction using Python/Java.