**CN**

**Unit 2**

### **Application Layer: Principles and Architectures of Network Applications**

The **Application Layer** is the topmost layer in the **OSI (Open Systems Interconnection) model** and the **TCP/IP model**, responsible for enabling communication between networked applications. It provides interfaces and protocols that allow software applications to communicate over a network.

## **Principles of Network Applications**

Network applications follow a client-server or peer-to-peer architecture and utilize various protocols to facilitate communication. These applications rely on **transport services** provided by the underlying layers.

Key principles include:

1. **Reliability:** Ensuring data integrity and correct order of delivery.
2. **Scalability:** Handling an increasing number of users or devices.
3. **Security:** Protecting data through encryption, authentication, and authorization.
4. **Interoperability:** Ensuring compatibility across different devices and platforms.
5. **Efficiency:** Minimizing bandwidth and computational overhead.

## **Architectures of Network Applications**

### **1. Client-Server Architecture**

* A **client** sends requests, and a **server** responds.
* Servers are often **always on**, while clients connect on demand.
* Examples: **Web browsing (HTTP), Email (SMTP, IMAP, POP3), File transfer (FTP).**

### **2. Peer-to-Peer (P2P) Architecture**

* Peers act as both clients and servers.
* Scales well without a central authority.
* Examples: **BitTorrent, Skype (older versions), Blockchain networks.**

### **3. Hybrid Architecture**

* A mix of **client-server** and **P2P**.
* Examples: **Instant messaging apps (WhatsApp, Telegram), CDN networks.**

## **Client and Server Processes**

A **process** is a running instance of a program. In network communication:

* **Client process**: Initiates communication (e.g., web browser, email client).
* **Server process**: Listens for and processes client requests (e.g., web server, email server).

**Inter-process Communication (IPC):**

* Processes communicate over a network using **sockets**.
* Data exchange follows application layer protocols (HTTP, FTP, DNS, etc.).

## **The Idea of a Socket**

A **socket** is an endpoint for sending or receiving data across a network. It acts as an interface between the application and the network.

* **A socket is defined by:  
  (IP Address, Port Number, Transport Protocol)**
* Common transport protocols:
  + **TCP sockets** (connection-oriented, reliable)
  + **UDP sockets** (connectionless, fast, best-effort)

Example:

import socket

# Creating a TCP socket

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect(("example.com", 80)) # Connects to a web server on port 80

## **Transport Services Available to the Application Layer**

The **Transport Layer** provides communication services for applications. The two most common transport protocols in the internet are:

### **1. Transmission Control Protocol (TCP)**

* **Connection-oriented:** Establishes a connection before data transfer.
* **Reliable:** Ensures ordered, error-free delivery.
* **Flow control & Congestion control:** Prevents overwhelming the receiver.
* **Used for:** HTTP (web browsing), FTP (file transfer), SMTP/IMAP/POP3 (email).

### **2. User Datagram Protocol (UDP)**

* **Connectionless:** No setup phase.
* **Unreliable:** No guarantee of order or delivery.
* **Low overhead, fast transmission.**
* **Used for:** DNS, VoIP, video streaming, online gaming.

| **Feature** | **TCP** | **UDP** |
| --- | --- | --- |
| Connection | Yes | No |
| Reliability | Yes | No |
| Speed | Slower | Faster |
| Ordered Data | Yes | No |
| Use Cases | Web browsing, Email, File transfer | Streaming, VoIP, Gaming, DNS |

## **Conclusion**

The **Application Layer** enables communication between networked applications using different architectures (client-server, P2P). It relies on **transport protocols (TCP, UDP)** for data transmission and uses **sockets** to facilitate communication between processes. Understanding these concepts is fundamental for designing efficient and scalable network applications.

# **Application Layer Protocols: The Web and HTTP**

The **HyperText Transfer Protocol (HTTP)** is the foundation of communication on the **World Wide Web**. It is an **application-layer protocol** that enables web browsers (clients) and web servers to exchange information.

## **Persistent and Non-Persistent Connections in HTTP**

HTTP can operate in two modes: **non-persistent** and **persistent** connections.

### **1. Non-Persistent HTTP**

* A **new TCP connection** is established for each HTTP request/response pair.
* After the server sends the requested object, the connection **closes**.
* **Disadvantages**:
  + **Increased latency** due to repeated TCP handshakes.
  + **Higher overhead** as multiple connections are opened and closed.

### **2. Persistent HTTP**

* The TCP connection **remains open** for multiple requests and responses.
* Reduces overhead and improves performance.
* Can operate in two modes:
  + **Without pipelining**: The client sends a new request only after receiving a response.
  + **With pipelining**: Multiple requests can be sent without waiting for responses.

**Comparison:**

| **Feature** | **Non-Persistent HTTP** | **Persistent HTTP** |
| --- | --- | --- |
| TCP Connection | Closed after each request | Reused for multiple requests |
| Latency | Higher | Lower |
| Efficiency | Less efficient | More efficient |

## **HTTP Message Format**

HTTP communication involves **request** and **response** messages.

### **1. HTTP Request Message**

A request message is sent by the client to the server and has the following format:

GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0

Accept: text/html

* **Request Line**: Specifies the HTTP method (GET, POST, PUT, etc.), the requested resource (/index.html), and the HTTP version (HTTP/1.1).
* **Header Fields**: Contain metadata about the request (e.g., Host, User-Agent, Accept).
* **Optional Body**: Included in POST and PUT requests.

### **2. HTTP Response Message**

A response message is sent by the server and has the following format:

HTTP/1.1 200 OK

Date: Sat, 01 Mar 2025 12:00:00 GMT

Server: Apache/2.4.1

Content-Type: text/html

Content-Length: 1234

<html>

<body>Welcome to my website!</body>

</html>

* **Status Line**: Includes the HTTP version, status code (e.g., 200 OK, 404 Not Found), and a reason phrase.
* **Header Fields**: Provide metadata about the response (Date, Server, Content-Type).
* **Message Body**: Contains the actual web content (HTML, JSON, etc.).

## **Cookies**

**Cookies** are small pieces of data stored on a client’s browser, used to maintain state between HTTP requests.

### **How Cookies Work**

1. **First Request (No Cookie)**
   * The client sends an HTTP request.
   * The server responds with a Set-Cookie header.

Set-Cookie: session\_id=12345; Expires=Wed, 10 Mar 2025 12:00:00 GMT

1. **Subsequent Requests (With Cookie)**
   * The client includes the cookie in the request.

Cookie: session\_id=12345

### **Uses of Cookies**

* **User Authentication**: Keeps users logged in.
* **Session Tracking**: Stores user preferences.
* **Personalization**: Customizes content based on user activity.

## **Proxy Server**

A **proxy server** is an intermediary between the client and the web server. It handles requests on behalf of the client.

### **Types of Proxies**

1. **Forward Proxy** (Used by Clients)
   * Clients send requests to the proxy, which forwards them to the destination server.
   * Used for content filtering, caching, and anonymity.
2. **Reverse Proxy** (Used by Servers)
   * Sits in front of web servers and distributes client requests.
   * Used for load balancing, security, and caching.

### **Benefits of Proxy Servers**

* **Caching**: Speeds up web access by storing frequently requested content.
* **Anonymity**: Hides the client's IP address.
* **Access Control**: Restricts access to certain websites.
* **Load Balancing**: Distributes traffic to multiple servers.

## **Conditional GET**

**Conditional GET** is a mechanism that reduces unnecessary data transfer by requesting data only if it has been modified.

### **How It Works**

The client includes an If-Modified-Since or If-None-Match header in the request.  
  
GET /index.html HTTP/1.1

Host: www.example.com

If-Modified-Since: Wed, 10 Mar 2025 12:00:00 GMT

1. The server checks if the resource has changed:
   * **If unchanged**, it responds with 304 Not Modified (no need to resend the file).
   * **If modified**, it sends a fresh copy with 200 OK.

### **Benefits of Conditional GET**

* **Reduces bandwidth usage** by avoiding redundant data transfer.
* **Improves performance** by minimizing unnecessary downloads.
* **Enhances efficiency** for web caching.

## **Conclusion**

HTTP is a fundamental protocol for the web, supporting **persistent and non-persistent connections**. It follows a structured **message format** and relies on **cookies** for session management. **Proxy servers** optimize performance and security, while **conditional GET** improves efficiency by reducing redundant data transfer.

* + **Passive Mode:** Client opens the data connection (used in firewalls/NAT).
* Supports authentication with username and password.

### **Commands and Responses**

* USER <username> → Log in with a username.
* PASS <password> → Authenticate user.
* LIST → List files in the directory.
* RETR <filename> → Download a file.
* STOR <filename> → Upload a file.
* QUIT → Close connection.

## **2. Email**

Email services use a combination of protocols for sending, retrieving, and formatting messages.

### **2.1 Simple Mail Transfer Protocol (SMTP)**

* Used for **sending email** from client to server and between mail servers.
* **Works over TCP (Port 25 or 587 for secure connections).**
* **Push protocol** (sends messages to recipient mail servers).
* Requires an SMTP server (e.g., smtp.gmail.com).

### **SMTP Example**

HELO mail.example.com

MAIL FROM:<alice@example.com>

RCPT TO:<bob@example.com>

DATA

Subject: Hello

This is a test email.

.

QUIT

### **2.2 Email Message Formats**

Emails follow **RFC 5322 format**, which includes:

* **Header**: Contains sender, receiver, subject, and other metadata.
* **Body**: Contains the main content.

**Example Email Format:**

From: Alice <alice@example.com>

To: Bob <bob@example.com>

Subject: Meeting Update

Let's meet at 3 PM today.

### **2.3 Mail Access Protocols (POP3 & IMAP)**

After an email is sent using **SMTP**, it is retrieved using **POP3 or IMAP**.

| **Feature** | **POP3** | **IMAP** |
| --- | --- | --- |
| Full Form | Post Office Protocol v3 | Internet Message Access Protocol |
| Port | 110 (unencrypted), 995 (SSL) | 143 (unencrypted), 993 (SSL) |
| Message Storage | Downloads emails to the client and deletes from the server (unless configured otherwise) | Emails remain on the server |
| Access from multiple devices | No | Yes |
| Synchronization | No | Yes |

* **POP3**: Simple, used for downloading emails to a single device.
* **IMAP**: More advanced, allows access from multiple devices.

### **2.4 Multipurpose Internet Mail Extensions (MIME)**

* Extends **email message format** to support:
  + Text in multiple character sets.
  + Attachments (images, audio, video, etc.).
  + HTML formatting.

**Example MIME Header for an Attachment:**

MIME-Version: 1.0

Content-Type: multipart/mixed; boundary="boundary123"

--boundary123

Content-Type: text/plain

Hello, this is an email with an attachment.

--boundary123

Content-Type: image/jpeg

Content-Disposition: attachment; filename="image.jpg"

## **3. Domain Name System (DNS)**

DNS is a **distributed database system** that resolves domain names (e.g., www.google.com) into IP addresses (e.g., 142.250.190.78).

### **3.1 DNS Services**

* **Name resolution:** Converts domain names to IP addresses.
* **Load balancing:** Distributes traffic among multiple servers.
* **Email routing:** Maps email addresses to mail servers.

### **3.2 How DNS Works**

1. A user enters www.example.com in a browser.
2. The browser checks the **local DNS cache**.
3. If not found, it queries a **recursive DNS resolver** (ISP's DNS).
4. The resolver queries the **Root DNS server**.
5. The Root DNS server directs it to a **Top-Level Domain (TLD) DNS server** (e.g., .com).
6. The TLD server directs it to the **Authoritative DNS server** for example.com.
7. The authoritative server returns the IP address.
8. The browser connects to the IP address.

### **3.3 Types of DNS Servers**

1. **Root DNS Servers**: 13 global servers that direct requests to TLD servers.
2. **TLD DNS Servers**: Store records for top-level domains (.com, .org, .net).
3. **Authoritative DNS Servers**: Store domain-specific records (example.com).

### **3.4 DNS Resource Records**

DNS stores information in **Resource Records (RRs)**.

| **Record Type** | **Purpose** |
| --- | --- |
| A | Maps domain to IPv4 address |
| AAAA | Maps domain to IPv6 address |
| CNAME | Alias for another domain |
| MX | Mail exchange record (for email servers) |
| NS | Name server for the domain |
| PTR | Reverse lookup (IP → Domain) |
| TXT | Miscellaneous text information |

### **3.5 DNS Message Format**

DNS messages are exchanged using **UDP (Port 53)**.

* **DNS Query Message**
  + Contains a domain name request.
* **DNS Response Message**
  + Contains the requested IP address.

## **4. Peer-to-Peer (P2P) File Distribution: BitTorrent**

Unlike **client-server**, P2P systems distribute data among peers (computers acting as both clients and servers).

### **4.1 How BitTorrent Works**

1. A user downloads a **.torrent file** containing metadata about the files.
2. A **tracker** provides a list of peers sharing the file.
3. The file is divided into small pieces.
4. Peers download pieces from multiple sources.
5. Once a peer has a piece, it **uploads it to others** (seeding).

### **4.2 Advantages of BitTorrent**

* **Efficient bandwidth usage:** No reliance on a central server.
* **Faster downloads:** Multiple sources provide file parts.
* **Redundancy:** File availability even if some peers leave.

### **4.3 Key BitTorrent Terms**

* **Seeder:** A peer that has the full file and uploads it.
* **Leecher:** A peer that is downloading the file.
* **Tracker:** A server that helps peers find each other.
* **Swarm:** The group of peers sharing the same file.

## **Conclusion**

* **FTP** enables file transfers between computers.
* **Email protocols** (SMTP, POP3, IMAP) manage email communication, while **MIME** allows attachments.
* **DNS** resolves domain names into IP addresses through a hierarchical structure.
* **P2P file distribution (BitTorrent)** efficiently shares files without a central server.