The **World Wide Web (WWW)** is a system of interlinked hypertext documents that are accessed via the internet. It allows users to navigate between web pages using hyperlinks, making it one of the most important services that runs on the internet. The web uses browsers, such as Chrome or Firefox, to retrieve and display content, including text, images, videos, and more.

Key aspects of the World Wide Web:

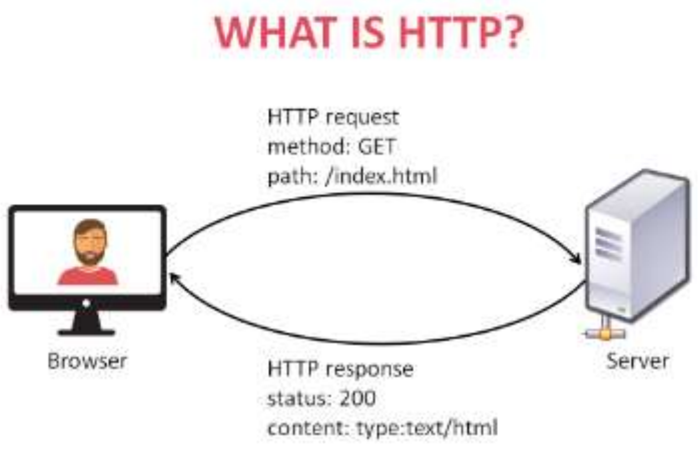
- **Hypertext**: Documents are connected through hyperlinks, allowing easy navigation.
- **Web Pages**: Written in HTML (HyperText Markup Language) and accessible using a browser.
- **HTTP/HTTPS**: Protocols used to transmit and display web pages securely.

The WWW is built on top of the internet and provides a user-friendly way to access vast amounts of information.

Here are some key facts about the **World Wide Web (WWW)**, presented point-wise:

1. **Created by Tim Berners-Lee** in 1989.
2. **Uses HTTP (Hypertext Transfer Protocol)** to communicate between browsers and servers.
3. **Web Pages** are written in HTML (HyperText Markup Language).
4. **Accessible via web browsers** like Chrome, Firefox, Safari, and Edge.
5. **Relies on DNS (Domain Name System)** to translate domain names into IP addresses.
6. **Runs on the internet**, which is the underlying network infrastructure.
7. **Links and hypertext** allow users to navigate between web pages.
8. **Evolved from static pages** to dynamic, interactive content.
9. **HTTPS** provides secure, encrypted communication on the web.
10. **Multimedia support** allows for images, videos, audio, and other types of media.

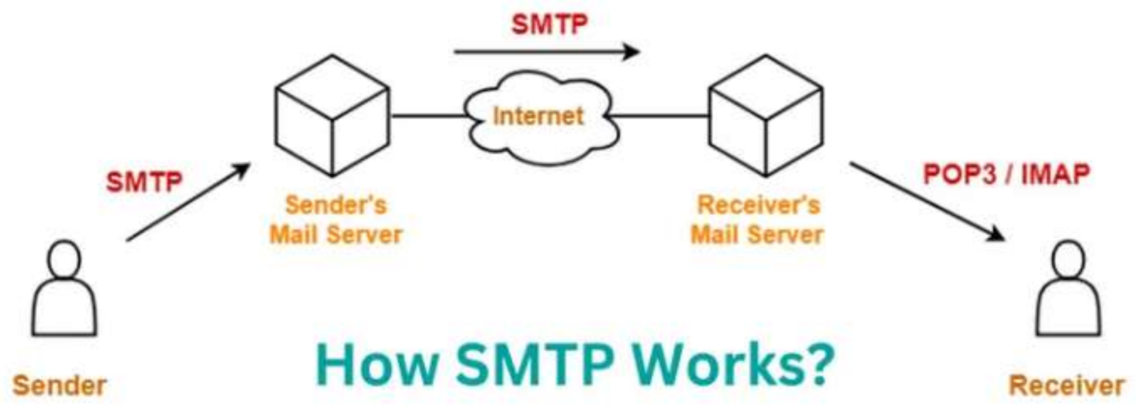
HTTP (Hypertext Transfer Protocol) is the foundational protocol used for transferring web pages and resources over the World Wide Web. It defines how messages are formatted and transmitted, and how web servers and browsers should respond to various commands.



Key Features of HTTP:

1. **Request-Response Model**: HTTP operates on a client-server model. The client (typically a web browser) sends an HTTP request to a server, and the server responds with the requested resource (e.g., an HTML page, image, or file).
    - **GET**: Requests data from the server.
    - **POST**: Submits data to be processed.
    - **PUT**: Updates an existing resource.
    - **DELETE**: Deletes a specified resource.
2. **Stateless Protocol**: Each request is independent and does not retain information between transactions. Servers treat each request as new, unrelated to previous ones.
3. **Ports**: HTTP typically uses **port 80**, while HTTPS (secure version) uses **port 443**.
4. **HTTPS (HTTP Secure)**: An encrypted version of HTTP using SSL/TLS protocols, which ensures that data transmitted between the client and server is secure and private.
5. **Headers and Body**: HTTP messages contain headers (metadata about the request/response) and, optionally, a body (the actual content like an HTML page or data

SMTP (Simple Mail Transfer Protocol) is a protocol used for sending and relaying email messages between email servers and from clients to servers. It is the standard protocol used for email transmission over the internet.



Key Features of SMTP:

1. **Sending Emails**: SMTP is responsible for the outgoing mail process, i.e., sending emails from a mail client (like Outlook or Gmail) to an email server, and from one server to another.
2. **Relaying Emails**: It also allows the transfer of email messages between different email servers to reach the recipient's mail server.
3. **Ports**:
   - **Port 25**: The default port for SMTP communication between servers.
   - **Port 587**: Often used for secure submission of email from clients to servers.
   - **Port 465**: Sometimes used for SMTP with SSL/TLS encryption.
4. **Commands**:
   - **HELO**: Introduces the client to the server.
   - **MAIL FROM**: Specifies the sender's email address.
   - **RCPT TO**: Specifies the recipient's email address.
   - **DATA**: Begins the transfer of the email content.
   - **QUIT**: Ends the session.

How SMTP Works:

1. **Email Client** (like Gmail, Outlook) sends the message to the SMTP server.
2. **SMTP Server** then determines whether the destination email address is on the same server.
    - If so, it delivers the email directly.
    - If not, it forwards the email to the recipient's email server.
3. The **recipient's email server** stores the email until the user retrieves it using a protocol like **POP** or **IMAP**.

POP (Post Office Protocol):

POP (Post Office Protocol) is a protocol used by email clients to retrieve messages from a mail server. It's designed to download emails to a local device and then, typically, remove them from the server.

Key Features of POP:

1. **Download and Delete**: Emails are usually downloaded from the server to the user's local device, and then removed from the server (though many clients can be configured to leave a copy on the server).
2. **Offline Access**: Once emails are downloaded, they can be accessed offline since they are stored locally.
3. **Single Device Usage**: POP is designed for use on a single device, as once emails are downloaded, they are not available on other devices unless configured otherwise.
4. **Port Numbers**:
    - **Port 110**: Standard port for unencrypted POP3.
    - **Port 995**: Used for POP3 over SSL (POP3S), which secures the connection.

How POP Works:

1. **Email client** connects to the POP server.
2. The client retrieves and downloads emails from the server.
3. The email is typically deleted from the server (unless configured otherwise).
4. The user reads and manages emails locally on their device.

Advantages of POP:

- Ideal for users who access their email from a single device.
- Emails are stored locally, allowing offline access.

Disadvantages of POP:

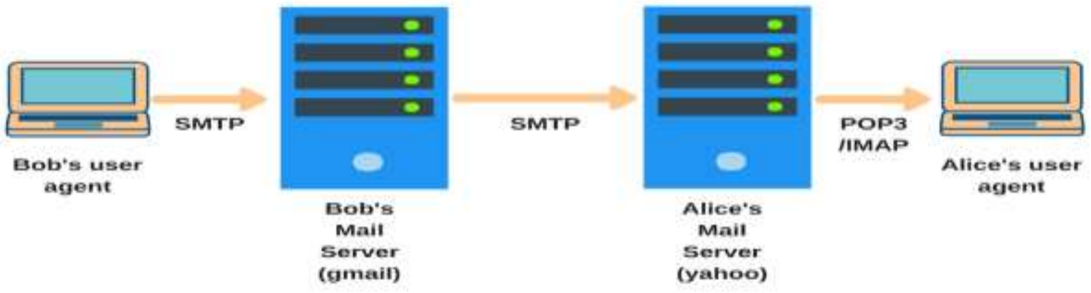
- Not suited for accessing the same email account on multiple devices (like a phone and a laptop), as emails are removed from the server.
- Less flexible for modern, multi-device usage.

IMAP (Internet Message Access Protocol):

IMAP (Internet Message Access Protocol) is a protocol used for retrieving email messages from a mail server, but unlike POP, it allows users to manage and organize emails on the server itself, making it more suitable for multi-device access.

Key Features of IMAP:

1. **Server-Side Management**: Emails remain on the server, and the client can view and manage them without downloading the messages. Folders and labels can be synchronized between multiple devices.
2. **Multiple Device Access**: IMAP is ideal for accessing email from multiple devices (like a laptop, smartphone, or tablet) since all changes are reflected on the server and visible across all devices.
3. **Partial Downloads**: IMAP can download only the headers or portions of messages, leaving the full message on the server until needed.
4. **Port Numbers**:
    - **Port 143**: Standard port for unencrypted IMAP.
    - **Port 993**: Used for IMAP over SSL (IMAPS), which encrypts the connection for security.



How IMAP Works:

1. **Email client** connects to the IMAP server.
2. The client views and manages emails directly on the server.
3. Emails are not necessarily downloaded to the device (or only partially downloaded), so they remain available on the server.
4. Any changes made (reading, deleting, moving messages to folders) are synchronized across all devices that access the email account.

Advantages of IMAP:

- Great for accessing the same email account from multiple devices.
- Email remains on the server, allowing real-time synchronization between devices.
- Full control over server-stored folders and organization.

Disadvantages of IMAP:

- Requires an internet connection to fully access and manage emails (though many clients offer offline functionality).
- Can consume more server space compared to POP, since emails are stored remotely unless manually deleted.

POP vs. IMAP:

- **POP**: Suitable for single-device users who prefer offline access and local storage of emails.
- **IMAP**: Better for users who access email from multiple devices and want to keep everything synchronized across them.

DNS (Domain Name System):

The **Domain Name System (DNS)** is a decentralized naming system for translating human-friendly domain names (like www.example.com) into IP addresses (such as 192.168.1.1), which are used by computers to locate and communicate with each other over the internet.

Key Features of DNS:

1. **Hierarchy**: DNS has a hierarchical structure with the following components:
   - **Root Level**: The top of the DNS hierarchy represented by a dot (.).
   - **Top-Level Domains (TLDs)**: Examples include .com, .org, .net, etc.
   - **Second-Level Domains**: The part directly before the TLD (e.g., example in example.com).
   - **Subdomains**: Domains under the second-level domain, like mail.example.com.
2. **DNS Records**: DNS stores various types of records:
   - **A Record**: Maps a domain name to an IPv4 address.
   - **AAAA Record**: Maps a domain name to an IPv6 address.
   - **MX Record**: Specifies the mail server responsible for receiving email.
   - **CNAME Record**: An alias for another domain.
   - **NS Record**: Points to the authoritative name servers for a domain.
3. **Resolvers and Recursive Lookup**: When a user types a domain name into a browser, a DNS resolver (usually provided by the user's ISP) queries multiple DNS servers in a recursive process to find the correct IP address.
4. **Ports**: DNS typically operates over **port 53**.

DNS Example:

- User enters www.example.com into their browser.
- The DNS resolver queries root servers, TLD servers, and authoritative servers to find the corresponding IP address, e.g., 192.0.2.1.
- The resolver returns the IP address to the browser, which then connects to the website.

MIME (Multipurpose Internet Mail Extensions):

MIME (Multipurpose Internet Mail Extensions) is a standard that extends the format of email to support:

- Text in character sets other than ASCII.
- Attachments such as images, audio, video, and application files.

- Multipart messages (i.e., an email that includes both plain text and HTML).

Key Features of MIME:

1. **Content Types**: MIME allows specifying the media type of the content in the email header. Examples include:
    - **text/plain**: For plain text.
    - **text/html**: For HTML-formatted text.
    - **image/jpeg**: For JPEG image files.
    - **application/pdf**: For PDF files.
2. **Multipart Messages**: MIME supports multipart emails, allowing a single email to contain multiple parts, such as text, images, or attachments. For example:
    - **multipart/mixed**: Used when the message includes both text and attachments.
    - **multipart/alternative**: Provides alternative versions of the same content (e.g., plain text and HTML).
3. **Base64 Encoding**: Binary data like images or documents are encoded into a text format using Base64 encoding to ensure safe transmission via email, which was originally designed to handle only plain text.
4. **MIME Headers**: These are added to the beginning of email messages and describe the content being sent. For example:
    - Content-Type: text/html; charset=UTF-8
    - Content-Disposition: attachment; filename="document.pdf"

FTP (File Transfer Protocol):

FTP (File Transfer Protocol) is a standard network protocol used for transferring files between a client and a server over the internet or a local network.

Key Features of FTP:

1. **File Transfer**: FTP allows users to upload, download, rename, delete, or move files between a client and server.
2. **Connection Modes**:
    - **Active Mode**: The client opens a random port and sends the port number to the server. The server then connects back to the client.

   - **Passive Mode**: The server opens a random port, and the client connects to it. This mode is used when the client is behind a firewall.
3. **Authentication**: FTP typically requires a username and password for access, but it can also be configured for anonymous access, where no authentication is needed.
4. **Unencrypted by Default**: Standard FTP is not secure, as it transmits data, including login credentials, in plain text. For secure file transfer, **FTPS (FTP Secure)** or **SFTP (SSH File Transfer Protocol)** is used.
5. **Ports**:
   - **Port 21**: Control connection (used for sending commands).
   - **Port 20**: Data connection (used for transferring files in active mode).

How FTP Works:

1. The **FTP client** connects to the **FTP server** over port 21.
2. The client can then send commands to upload, download, rename, or delete files on the server.
3. Files are transferred using a separate data connection.

Example FTP Session:

- **Client**: USER username
- **Server**: 331 Password required
- **Client**: PASS password
- **Server**: 230 User logged in
- **Client**: RETR file.txt (Download file)
- **Server**: 150 Opening data connection

Advantages of FTP:

- Widely used for file transfer between client and server.
- Supports large file transfers.

Disadvantages of FTP:

- **Security risk**: Data, including credentials, is sent unencrypted in standard FTP.
- Requires additional configuration for secure versions like FTPS or SFTP.

Telnet:

Telnet is a network protocol used for remote communication between two devices, typically a client and a server. It allows users to connect to and execute commands on a remote machine as if they were physically present at the machine.

Key Features of Telnet:

1. **Remote Access**: Telnet allows users to log into a remote computer or device to execute commands in a terminal session.
2. **Command-Line Interface (CLI)**: Once connected, the user is presented with a text-based interface to control the remote machine, which can be used for tasks such as system management, configuration, or file operations.
3. **Unencrypted Communication**: Telnet transmits all data, including login credentials, in plain text. This makes it insecure for use over the internet. **SSH (Secure Shell)** is a more secure alternative that encrypts all communication.
4. **Ports**:
    - **Port 23**: The default port used by Telnet.

How Telnet Works:

1. The **client** establishes a connection to the **Telnet server**.
2. The user logs in and is given command-line access to the remote system.
3. Commands are sent to the remote machine and executed, with the results displayed to the user.

Telnet
Telnet is a network protocol used to provide a bidirectional, interactive communication facility for a user to connect to remote systems. Developed in the early 1970s, it operates on port 23 by default and allows users to log into remote computers, primarily UNIX systems, over a network. Telnet offers a command-line interface that allows users to control systems remotely, often used for network diagnostics, remote server management, or accessing services.

Key Characteristics:

Protocol Type: Telnet is a client-server protocol and is part of the TCP/IP protocol suite.
Port Number: By default, it operates on TCP port 23.
Unencrypted Communication: Telnet sends data, including login credentials, in plaintext, making it insecure for modern use. It is highly susceptible to eavesdropping or interception.

Use Cases:

Remote administration and management of devices (routers, switches, servers).
Debugging network services.
Superseded by Secure Shell (SSH): Due to its lack of security, Telnet has largely been replaced by SSH (Secure Shell), which encrypts the data, including passwords and session information.

Basic Telnet Commands:

telnet [hostname] [port]: Connect to a remote server.
open [hostname]: Opens a connection to the server.
quit: Closes the connection.
status: Displays status information about the connection.

Introduction to Mobile Computing

Mobile computing refers to the use of portable computing devices, such as smartphones, tablets, and laptops, in conjunction with wireless networks to access data and applications from virtually any location.

It enables users to remain connected and access resources in real time, offering flexibility and mobility in various personal and professional settings.

Key Concepts:

1. **Mobile Devices**:
   - Portable computing devices such as smartphones, tablets, laptops, and wearables.
   - **These devices are equipped with wireless communication capabilities** and support a variety of operating systems (iOS, Android, Windows, etc.).
2. **Networks**: Mobile computing relies on various types of networks to enable connectivity and data access.

Wireline Networks: Traditional wired networks that use **physical cables** (Ethernet, fiber optics) for communication. These networks provide stable and fast connections but limit mobility.

Characteristics:

- Stable and fast data transmission.
- High bandwidth and low latency.
- Less prone to interference compared to wireless networks.

Use Case: Best suited for environments where high speed, reliability, and security are required, such as corporate offices, data centers, and homes.

Wireless Networks: Wireless technologies (Wi-Fi, LTE, 5G) allow devices **to connect to the internet without the need for physical cables**. Wireless networks offer flexibility and are integral to mobile computing.

Characteristics:

- Provides flexibility and mobility.
- Devices can connect from virtually anywhere within the network range.
- The speed and bandwidth depend on the wireless technology (Wi-Fi, 4G, 5G, etc.).

Limitations:

- Lower security compared to wired networks unless proper encryption and security protocols are used.

Use Case: Ideal for mobile computing, homes, and public hotspots where users need flexible access to the internet.

- **Ad-hoc Networks**: A decentralized type of wireless network where devices communicate directly with each other without relying on a central infrastructure (like a router or base station). **This is commonly used in scenarios such as sensor networks, peer-to-peer file sharing, or in remote areas where traditional networks aren't available.**

Characteristics:

- Dynamic and self-configuring; devices connect on the fly.
- No central point of failure, making the network resilient in certain environments.

- Typically used in specific scenarios where traditional infrastructure is unavailable or unnecessary.

Limitations:
- Limited range and scalability.
- Typically lower performance compared to structured wireless networks.
- More complex routing due to the decentralized nature.

Use Case: Used in temporary or emergency scenarios like disaster recovery, military communications, or remote areas, and in applications like sensor networks and peer-to-peer file sharing.