UNIT 2 APPLICATION LAYER

Introduction; Application Architectures (Host-Based Architectures, Client-Based Architectures, Client-Server Architectures, Cloud Computing Architectures, Peer-to-Peer Architectures, Choosing Architectures); World Wide Web (Working of WWW, HTTP Request and Response); Electronic Mail (Working of Email, SMTP Packet, Multipurpose Internet Mail Extension); Other Applications (Telnet, Instant Messaging, Videoconferencing).

- **Introduction**
- The application layer is the top-most layer of OSI model.
- It provides services directly to user applications.
- It enables the to access the network.
- It provides user interfaces and support for services such as email, remote file access and transfer, shared database management and other types of distributed information services.
- The functions of the application layer are:

- It facilitates the user to use the services of the network.

- It is used to develop network-based applications.

- It provides user services like user login, naming network devices, formatting messages, and e-mails, transfer of files etc.

- It is also concerned with error handling and recovery of the message as a whole

Application layer protocols

• Hyper Text Transfer Protocol, HTTP: It is the underlying protocol for world wide web. It defines how hypermedia messages are formatted and transmitted.

• File Transfer Protocol, FTP: It is a client-server based protocol for transfer of files between client and server over the network.

• Simple Mail Transfer Protocol, SMTP: It lays down the rules and semantics for sending and receiving electronic mails (e-mails).

• Domain Name System, DNS: It is a naming system for devices in networks. It provides services for translating domain names to IP addresses.

• TELNET: It provides bi-directional text-oriented services for remote login to the hosts over the network.

• Simple Network Management Protocol, SNMP: It is for managing, monitoring the network and for

organizing information about the networked devices.

- **Functionalities of the Application layer**

Specific **functionalities of the Application layer are as follows:**

1. Network Virtual terminal

- The application layer is the software version of a physical terminal and this layer permitted to a user to log on to a remote host.
- For this, an application creates a software emulation of a terminal at the remote host. By this user's computer can communicate with the software terminal, which in turn, communicates with the host.
- It is shown that the remote host is communicating with one of its terminals, so it allows the user to log on.
- **2. File Transfer, Access, and Management (FTAM)**
- An application permits a user to access files in a remote computer, to retrieve files from a computer and to manage files on a remote computer.
- FTAM is concerned with a hierarchical virtual file in terms of file attributes, file structure and the types of operations performed on the files and their attributes.

3. Addressing

- To achieve communication between client and server system, there is a need for addressing.
- When a request is sent from the client side to the server side, this request contains the server address and its own address.
- The server answered to the client request, this request contains the destination address, i.e., client address. DNS is used to achieve this type of addressing.

4. Mail Services

- Email forwarding and storage of e-mails provided by an application layer.

5. Directory Services

- A distributed database is contained by an application that provides access for global information about various objects and services.

6. Authentication

- It provides authentication to occur between devices for an extra layer of security and it authenticates the sender or receiver's message or both.

- **Application Architecture**
- To use the Internet, two application programs need to communicate with each other. One program runs on a computer somewhere in the world, and the other runs on a different computer somewhere else. Application architecture defines how these programs interact and how the functions of the software are distributed across clients (user devices) and servers (remote computers).
- **Four Key Functions of Application Programs**
- Any application program performs four main functions, which are the basic building blocks of the application:

1. **Data Storage**: This involves storing and retrieving data. The data could be something small like a text file or something large like a company's entire database of accounting records.
2. **Data Access Logic**: This refers to the processing needed to access the stored data. It often involves running database queries, typically using SQL (Structured Query Language).
3. **Application Logic (Business Logic)**: This is the core functionality of the application. It involves the rules and operations that determine how data is processed and used. Depending on the application, this logic can be simple or complex.
4. **Presentation Logic**: This involves displaying information to the user and accepting user commands. It includes the design of user interfaces and how the user interacts with the application

- **Online Shopping Application Example**
- Imagine you're using an online shopping website like Daraz . Here's how the application architecture might work:

1. **Data Storage**
    1. **Server**: All product information, user data, order history, and inventory details are stored in databases on the server. These databases can be large and need to handle a lot of data efficiently.
    2. **Example**: The server stores a database with all the products, including their descriptions, prices, and images.
2. **Data Access Logic**
    1. **Server**: When you search for a product or view your order history, the server processes these requests using SQL queries to fetch the necessary data from the database.
    2. **Example**: When you search for "laptops," the server runs a SQL query to retrieve all products in the "laptops" category from the database and sends this data to your browser.
3. **Application Logic (Business Logic)**
    1. **Server and Client**: This logic can be on both the server and client, but complex operations are usually handled by the server. It includes rules for calculating prices, applying discounts, managing user sessions, and processing payments.
    2. **Example**: When you add a product to your cart, the server calculates the total price, including any applicable taxes and shipping fees. If there's a promotion, the server applies the discount.
4. **Presentation Logic**
    1. **Client (Browser or Mobile App)**: This is the user interface that you interact with. It includes the layout of the web page, buttons, forms, and how data is displayed to you. It also handles user input like clicks and form submissions.
    2. **Example**: When you visit the homepage, your browser displays the layout with product categories, featured products, and a search bar. If you click on a product, the client displays the product details page.

- There are many ways in which these four functions can be allocated between the client computers and the servers in a network. There are four fundamental application architectures in use today.
- In **host-based architectures, the server (or host computer) performs virtually all of the** work.
- In **client-based architectures, the client computers perform most of the work.**
- In **client-server architectures, the work is shared between the servers and clients.**
- In **peer-to-peer architectures, computers are both clients and servers and thus share the** work. The client-server architecture is the dominant application architecture.

- **Host-Based Architecture**: In this setup, the server (usually a large mainframe computer) performed all the computational tasks and data processing, while the clients (usually terminals) served as simple input and output devices.
- **Key Components**

1. **Server (Mainframe Computer)**

   Function: The mainframe computer was responsible for all processing tasks. It handled everything from data storage, data processing, and application logic.

   Role: It was the central powerhouse of the network, performing all functions necessary to run applications and manage data.

2. **Clients (Terminals)**

   1. **Function**: The terminals were very basic devices with limited capabilities. They primarily acted as interfaces for users to interact with the mainframe computer.
   2. **Role**: They captured keystrokes (user input), sent this input to the server, and displayed the output received from the server.

How It Worked

1. **User Input**: A user would type commands or data into the terminal. The terminal would capture these keystrokes.
2. **Data Transmission**: The captured keystrokes would be sent to the mainframe server over the network.
3. **Processing**: The mainframe server would receive the input, process it (which could involve complex computations, database queries, etc.), and generate the appropriate response or result.
4. **Output**: The server would send the results back to the terminal.
5. **Display**: The terminal would receive the server's instructions on what to display, and it would show the result to the user.

Host-Based Architectures

Server
(mainframe computer)

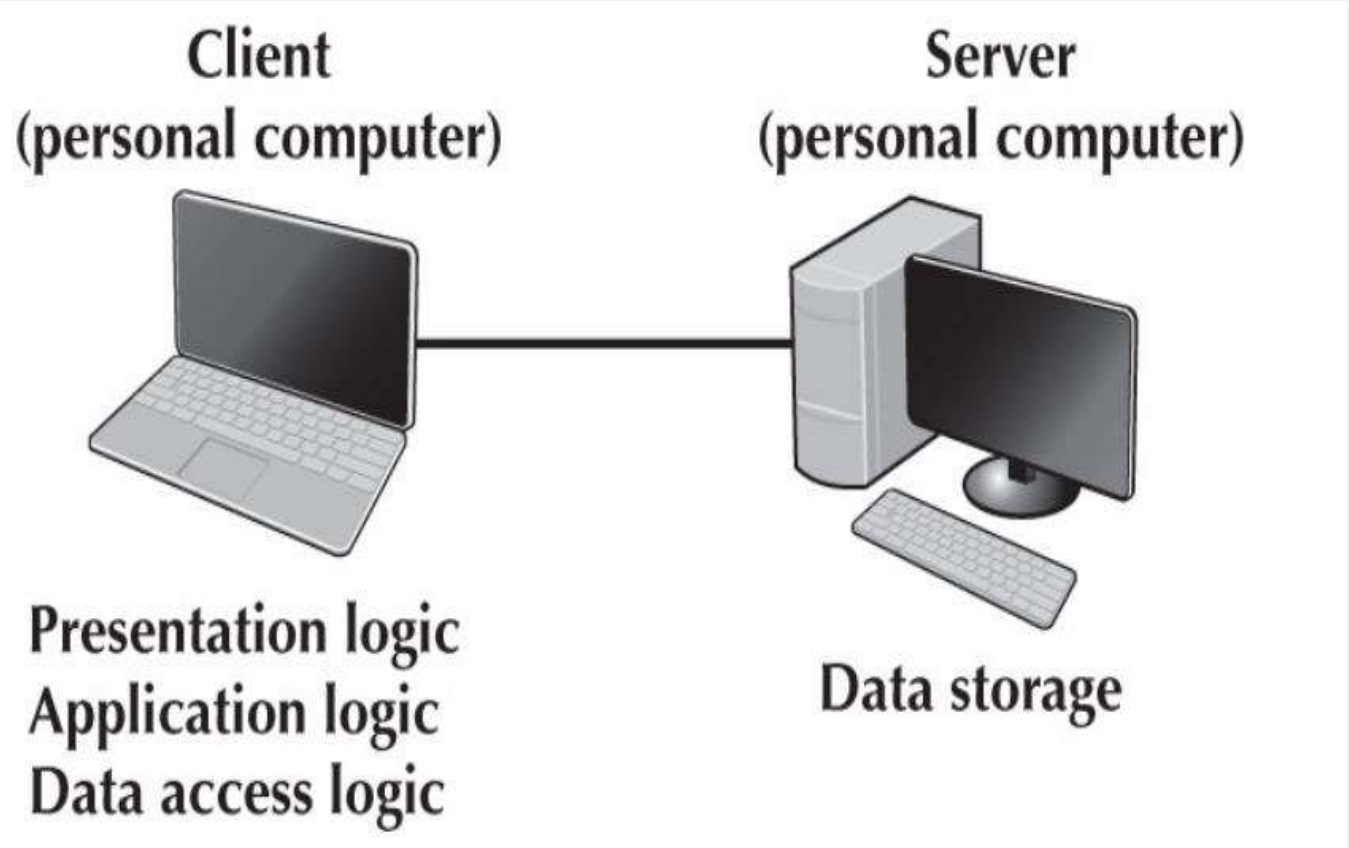
Client
(terminal)

or a PC running a terminal emulation program

Presentation logic
Application logic
Data access logic
Data storage

- captures key strokes then sends them to the mainframe
- displays information according to the server's instructions

- **Client Based Architectures:**
- In the late 1980s, there was an explosion in the use of personal computers. Today, more than 90% of most organizations' total computer processing power now resides on personal computers, not in centralized mainframe computers. Part of this expansion was fueled by a number of low-cost, highly popular applications such as word processors, spreadsheets, and presentation graphics programs. It was also fueled in part by managers' frustrations with application software on host mainframe computers. Most mainframe software is not as easy to use as personal computer software, is far more expensive, and can take years to develop.



- With client-based architectures, the clients are personal computers on a LAN, and the server is usually another personal computer on the same network. The application software on the client computers is responsible for the presentation logic, the application logic, and the data access logic; the server simply stores the data.

- This simple architecture often works very well. If you've ever used a word processor and stored your document file on a server (or written a program in Visual Basic or C that runs on your computer but stores data on a server), you've used a client-based architecture.

- The fundamental problem in client-based networks is that all data on the server must travel to the client for processing. For example, suppose the user wishes to display a list of all employees with company life insurance. All the data in the database (or all the indices) must travel from the server where the database is stored over the network circuit to the client, which then examines each record to see if it matches the data requested by the user. This can overload the network circuits because far more data are transmitted from the server to the client than the client actually need

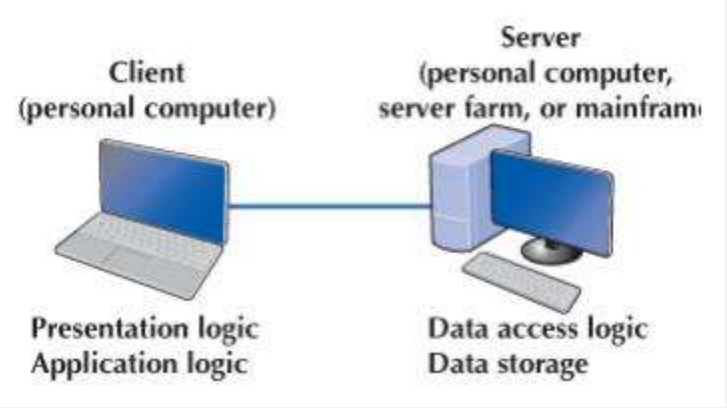
- **Client-Server Architecture**
- Client-server architecture is a way to design applications where tasks are split between two main parts: the client and the server
- **Basic Idea**
- **Client**: The part of the application that users interact with. It handles what users see and how they input data.
- **Server**: The part that manages data and handles requests from the client.
- **How It Works**

1. **Client-Side**:
    1. **Presentation Logic**: This is where the client displays information to the user and collects user input.
    2. **Application Logic**: Sometimes, the client also processes some of the data or performs certain tasks before sending a request to the server.
2. **Server-Side**:
    1. **Data Access Logic**: The server processes requests from the client, such as retrieving data from a database.
    2. **Data Storage**: The server stores and manages the data.

- **Simple Example**

1. **Using a Web Browser**:
    1. You open your browser (client) and request a web page.
    2. The browser sends this request to a web server.
    3. The server retrieves the necessary data and sends it back to your browser.
    4. Your browser displays the web page.

1. **Writing a Program with SQL**:
    1. Your program (client) needs some data from a database.
    2. The program sends an SQL request to the database server.
    3. The server processes the request and sends back the data.
    4. Your program uses this data and maybe displays it to the user.

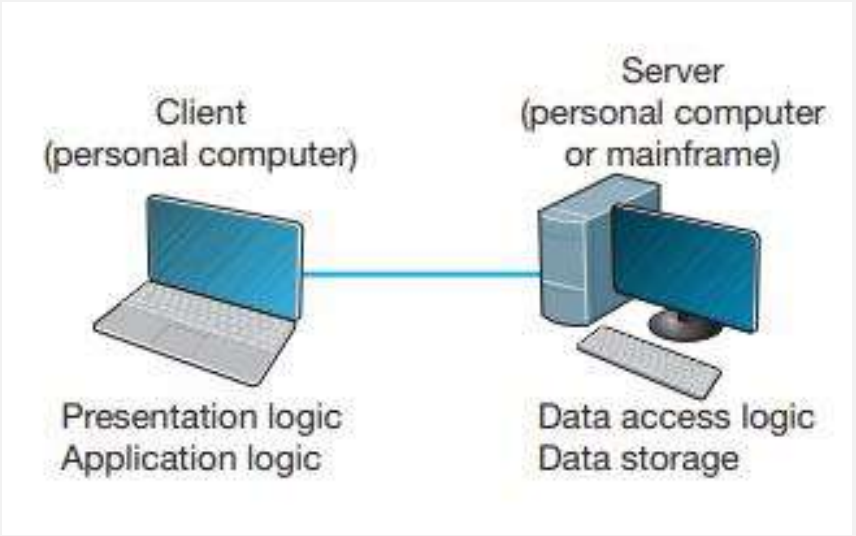


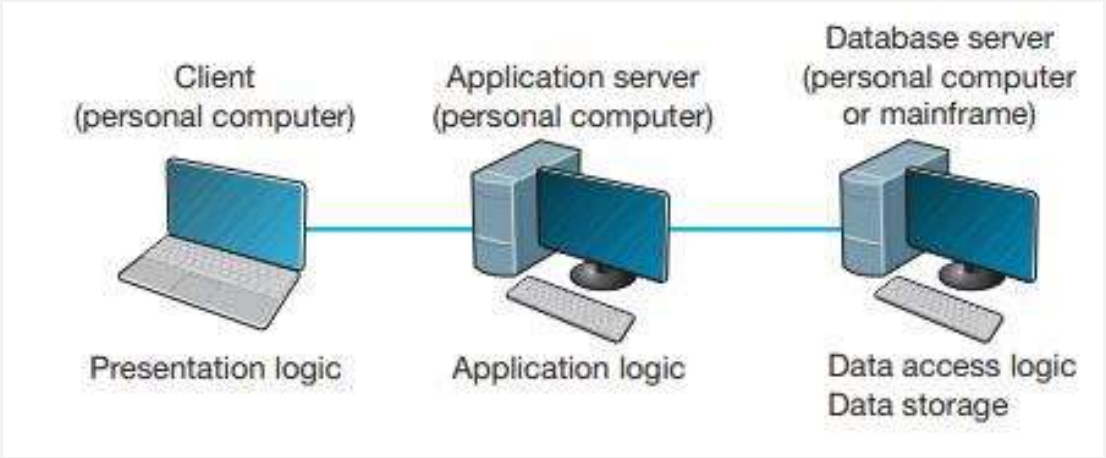
- **Key Points**
- **Client handles presentation**: Displays information and collects user input.
- **Server handles data**: Manages data requests and storage.
- **Balanced Processing**: Both client and server share the workload, making the system efficient

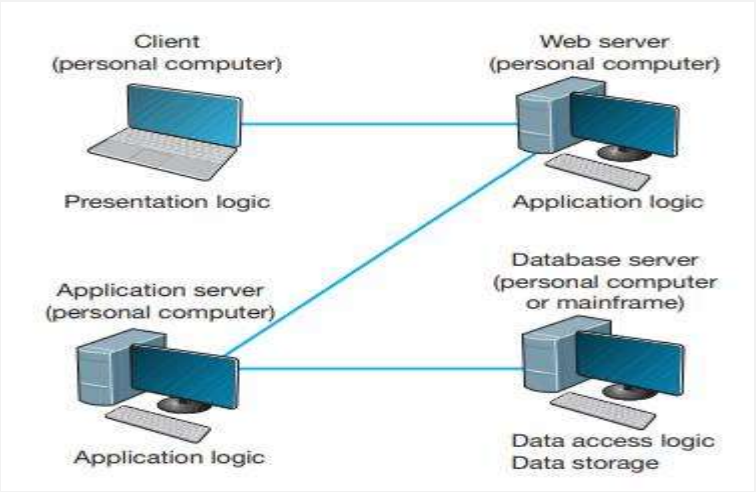
- **Types of Client-Server Architecture**
- **2-Tier Architecture**
- **Definition**: In a 2-tier architecture, the presentation and business logic layers are on the client, while the data layer is on a server.
- **How it Works**:
- **Client Side**: Contains both the presentation layer (what the user sees and interacts with) and the business logic layer (rules and processes of the application).
- **Server Side**: Houses the data layer, which is the database storing the application's data.
- **Example**:
- A desktop application where you log in to an online account.
- The application on your computer handles the user interface and the business logic.
- It sends requests to a remote database server to store or retrieve data.

- **3-Tier Architecture**
- **Definition**: In a 3-tier architecture, the presentation layer is on the client, the business logic layer is on one server, and the data layer is on another server.
- **How it Works**:
- **Client Side**: Contains the presentation layer.
- **Application Server**: Contains the business logic layer, processing the application's rules and processes.
- **Database Server**: Contains the data layer, storing and managing the data.
- **Example**:
- A smartphone app where you interact with the user interface.
- The app sends requests to an application server that processes the requests.
- The application server then interacts with a database server to retrieve or store data.

- **N-Tier Architecture**
- **Definition**: In N-tier architecture, the business logic layer is split into multiple layers to improve performance, management, and stability. The presentation and data layers remain as in the 3-tier architecture.
- **How it Works**:
- **Client Side**: Contains the presentation layer.
- **Multiple Business Logic Layers**: Different layers handle different parts of the business logic. This can include services like authentication, payment processing, and more.
- **Database Server**: Contains the data layer, managing the storage and retrieval of data.
- **Example**:
- A complex web application where you use a web interface.
- The interface communicates with multiple servers, each handling different parts of the business logic (e.g., one server for user authentication, another for payment processing).
- These servers interact with a central database server to manage data.


Client
(personal computer)
Server
(personal computer
or mainframe)
Presentation logic
Application logic
Data access logic
Data storage


Client
(personal computer)
Application server
(personal computer)
Database server
(personal computer
or mainframe)
Presentation logic
Application logic
Data access logic
Data storage


Client
(personal computer)
Web server
(personal computer)
Presentation logic
Application logic
Application server
(personal computer)
Database server
(personal computer
or mainframe)
Application logic
Data access logic
Data storage

Feature	2-Tier Architecture	3-Tier Architecture	N-Tier Architecture
Layers	- Presentation & Business Logic (Client)	- Presentation (Client)	- Presentation (Client)
	- Data (Server)	- Business Logic (Application Server)	- Multiple Business Logic Layers (Multiple Servers)
		- Data (Database Server)	- Data (Database Server)
Client Role	- Handles UI and Business Logic	- Handles UI	- Handles UI
Server Role	- Manages Data Storage	- Application Server processes Business Logic	- Multiple Servers process different aspects of Business Logic
		- Database Server manages Data Storage	- Database Server manages Data Storage
Complexity	- Simple, Direct Communication	- Moderately Complex	- High Complexity due to Multiple Layers
Scalability	- Limited Scalability	- Better Scalability	- High Scalability
Performance	- Dependent on Client Performance	- Better Load Distribution	- Optimal Load Distribution
Maintainability	- Easier to Maintain Client and Server Separately	- More Maintenance Effort for Application and Database Servers	- Highest Maintenance Effort due to Multiple Layers
Example	- Desktop App with Online Login	- Smartphone App with Separate App and Database Servers	- Complex Web App with Distributed Business Logic

- **Thin Client**
- **Definition**: A lightweight computer that relies on the resources of a remote server.
- **Characteristics**:
- Minimal Processing Power: Most computations are performed on the server.
- Minimal Local Storage: Data and applications are stored on the remote server.
- Network Dependency: Requires a reliable network connection to function effectively.
- Centralized Management: Easier to manage since updates and backups are done on the server.
- **Advantages**:
- Cost-Effective: Lower hardware costs due to reduced need for powerful client machines.
- Enhanced Security: Data is stored on a central server, reducing the risk of data theft or loss.
- Ease of Management: Simplified software deployment and updates, as changes are made on the server.
- Reduced Local Resource Usage: Lower power consumption and heat generation.
- **Disadvantages:**
- Network Dependency: Performance is heavily dependent on network reliability and speed.
- Limited Offline Capability: Has limited functionality when not connected to the server.
- **Examples:**
- Virtual Desktop Infrastructure (VDI): Thin clients connect to virtual desktops hosted on a central server.
- Web-Based Applications: Applications run in a web browser and rely on server-side processing

- **Thick Client**
- **Definition**: A computer that relies lightly on the server or computer and performs most data processing locally.
- **Characteristics**:
- Own computing power
- Local Storage and Processing: Stores data and runs applications locally.
- Less Dependent on Network: Can function independently of a network connection for most tasks.
- Independent Software: Has its own software applications and operating system.
- **Advantages**:
- Performance: Can perform tasks faster since processing is done locally.
- Offline Capability: Can operate without a network connection.
- Flexibility: Users can install and run various software applications independently of the server.
- **Disadvantages**:
- Higher Costs: More expensive hardware due to the need for powerful processors and ample storage.
- Security Risks: More vulnerable to security threats since data is stored locally.
- Maintenance: More complex to manage due to the need for individual updates and backups.
- **Examples:**
- Desktop Applications: Microsoft Office, Adobe Photoshop.
- Standalone Software: Video games, software development environments

- **Cloud Computing Architectures**
- The traditional client–server architecture can be complicated and expensive to deploy. Every application has to be hosted on a server so that it can fulfill requests from potentially thousands of clients. An organization has hundreds of applications, so running a successful client–server architecture requires a variety of software and hardware and the skilled personnel who can build and maintain this architecture.
- Cloud computing architectures are different because they outsource part or all of the infrastructure to other firms that specialize in managing that infrastructure. There are three common cloud-based architecture models.

- **Software as a Service (SaaS):**
- Saas is one of the three cloud computing models. With SaaS, an organization outsources the entire application to the cloud provider and uses it as any other application that is available via a browser (thin client). SaaS is based on multitenancy. This means that rather than having many copies of the same application, there is only one application that everybody shares, yet everybody can customize it for his or her specific needs. Popular SaaS offering the include Microsoft 365, Salesforce, and Google Workspace (also known as, G Suite).
- **Platform as a Service (PaaS):**
- PaaS is another of the three cloud computing models. What if there is an application you need but no cloud provider offers one you like? You can build your own application and manage your own data on the cloud infrastructure provided by your cloud supplier. This model is called Platform as a Service. The developers in your organization decide what programming language to use to develop the application of choice. The needed hardware and software infrastructure, called the platform, is rented from the cloud provider.

- **Infrastructure as a Service (laaS):**
- With laas, the cloud provider manages the hardware, including servers, storage, and networking components. The organization is responsible for all the software, including operating system (and virtualization software), database software, and its applications and data. IaaS is sometimes referred to also as Haas, or Hardware as a Service, because in this cloud model, only the hardware is provided; everything else is up to the organization This model allows a decrease in capital expenditures for hardware and maintaining the proper environment (e.g., cooling) and redundancy, and backups for data and applications. Providers of laas are Amazon Web Services, Microsoft Windows Azure, and so on.

- **Peer-to-peer (P2P)** architectures are very old, but their modern design became popular in the early 2000s with the rise of P2P file-sharing applications (e.g., Napster). With a P2Parchitecture, all computers act as both a client and a server. Therefore, all computers perform all four functions: presentation logic, application logic, data access logic, and data storage. With aP2P file-sharing application, a user uses the presentation, application, and data access logic installed on his or her computer to access the data stored on another computer in the network. With a P2P application-sharing network (e.g., grid computing such as seti.org), other users in the network can use others' computers to access application logic as well.

- **Advantages of Peer-to-Peer Architectures**

1. **Scalability**:
    1. Can handle a large number of peers without significant performance degradation.
    2. Resources grow as new peers join.
2. **Fault Tolerance**:
    1. No single point of failure.
    2. The network can adapt to changes and failures dynamically.
3. **Resource Utilization**:
    1. Efficient use of distributed resources.
    2. Peers can share their resources, reducing the need for dedicated infrastructure.
4. **Cost-Effectiveness**:
    1. Lower cost as there is no need for central servers.
    2. Peers contribute their resources voluntarily.

- **Disadvantages of Peer-to-Peer Architectures**

1. **Security**:
    1. More challenging to secure since each peer can be a potential point of vulnerability.
    2. Trust and authentication can be difficult to manage.
2. **Data Consistency**:
    1. Ensuring data consistency across distributed peers can be challenging.
    2. May require complex algorithms to manage synchronization.
3. **Search Efficiency**:
    1. Finding specific resources can be less efficient, especially in unstructured P2P networks.
    2. May require extensive searching or flooding the network.
4. **Network Management**:
    1. Harder to manage and control compared to centralized systems.
    2. Can lead to issues with coordination and updates.

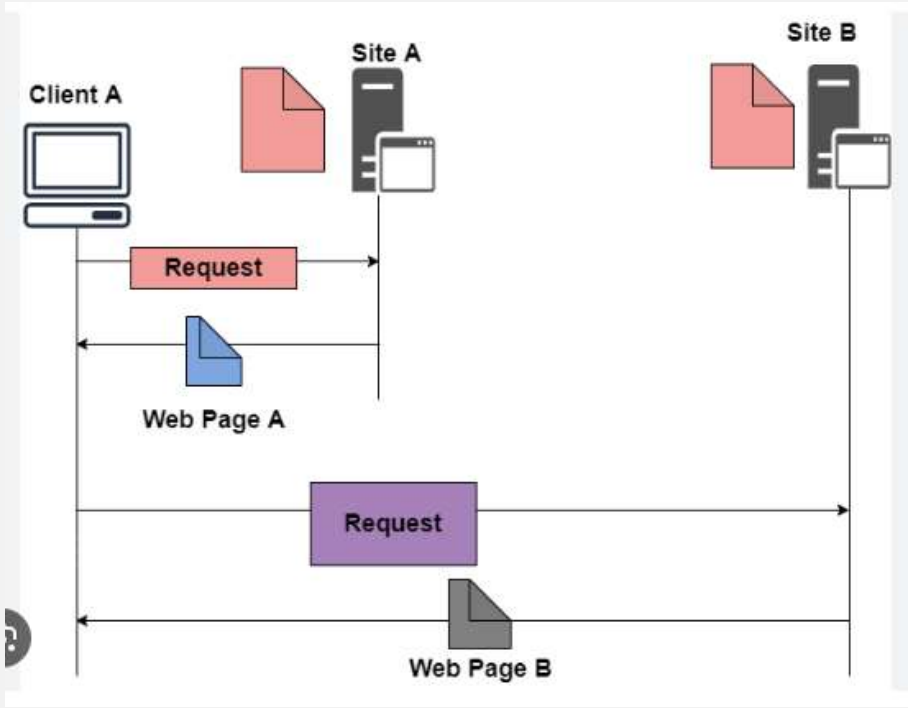
- **Choosing Architectures**
- Each of the preceding architectures has certain costs and benefits, so how do you choose the “right” architecture? In many cases, the architecture is simply a given; the organization has a certain architecture, and one simply has to use it. In other cases, the organization is acquiring new equipment and writing new software and has the opportunity to develop a new architecture, at least in some part of the organization.
- Almost all new applications today are client–server applications. Client– server architectures provide the best scalability, the ability to increase (or decrease) the capacity of the servers to meet changing needs. For example, we can easily add or remove application servers or database servers depending on whether we need more or less capacity for application software or database software and storage.
- •Client–server architectures are also the most reliable. We can use multiple servers to perform the same tasks, so that if one server fails, the remaining servers continue to operate and users don’t notice problems.

- Finally, client–server architectures are usually the cheapest because many tools exist to develop them. And lots of client–server software exists for specific parts of applications so we can more quickly buy parts of the application we need. For example, no one writes Shopping Carts anymore; it’s cheaper to buy a Shopping Carts software application and put it on an application server than it is to write your own.
- Client–server architectures also enable cloud computing. As we mentioned in cloud computing architectures, companies may choose to run a SaaS because of low price and high scalability compared to traditional client–server architecture hosted at home. One major issue that companies face when choosing SaaS is the security of the data. Each company has to evaluate the risk of its data being compromised and select its cloud provider carefully. However, SaaS is gaining popularity and companies are becoming more and more accustomed to this solution.

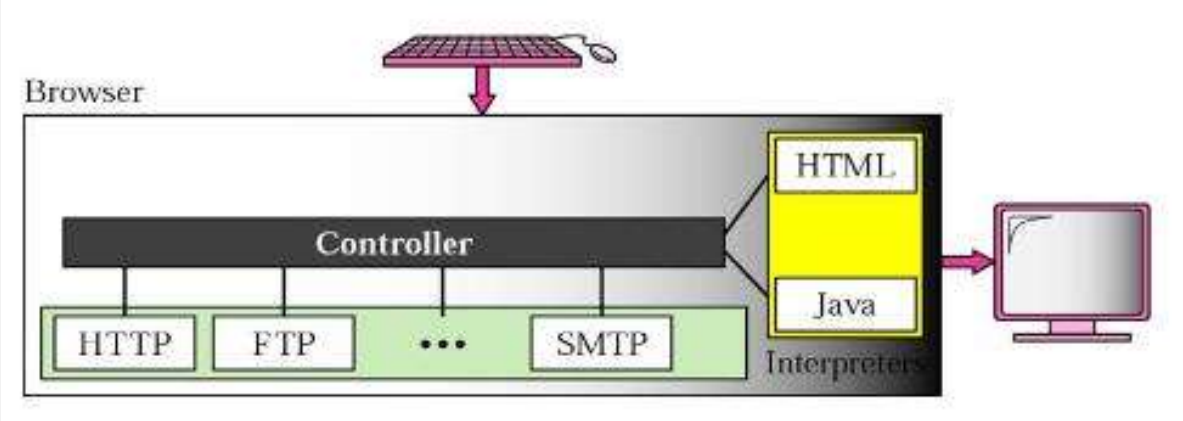
WORLD WIDE WEB

- The World Wide Web (WWW), or the Web, is a repository of information spread all over the world and linked together. The WWW has a unique combination of flexibility, portability, and user-friendly features that distinguish it from other services provided by the Internet. The WWW project was initiated by CERN (European Laboratory for Particle Physics) to create a system to handle distributed resources necessary for scientific research.
- Today it is a distributed client-server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called websites. Each website holds one or more documents, referred to as web pages. Each web page, however can contain some links to other web pages in the same or other websites. The pages can be retrieved and viewed by using browsers.

- Let us go through the scenario shown in figure. The client needs to see some information that it knows belongs to site A. It sends a request through its browser, a program that is designed to fetch Web documents. The request, among other information, includes the address of the site and the Web page, called the URL. The server at site A finds the document and sends it to the client. When the user views the document, s/he finds some references to other documents, including a Web page at site B. The reference has the URL for the new site. The user is also interested in seeing this document. The client sends another request to the new site, and the new page is retrieved.
- The above example shows the idea of hypertext and hypermedia. Hypertext means creating documents that refer to other documents. In a hypertext document, a part of text can be defined as a link to another document. When a hypertext is viewed with a browser, the link can be clicked to retrieve the other document. Hypermedia is a term applied to document that contains links to other textual document or documents containing graphics, video, or audio.



- **Browser**
- A variety of vendors offer commercial browsers that interpret and display a web document, and all of them use nearly the same architecture. Each browser usually consists of three parts a controller, client programs, and interpreters as shown in figure.



- **Controller:**
- Receives input from keyboard or mouse.
- Uses client programs to access documents.
- Uses interpreters to display documents.
- **Client Programs:**
- Access documents using protocols like HTTP, FTP, or TELNET.
- HTTP is the most commonly used protocol.
- **Interpreters:** Languages for displaying documents, such as HTML, Java, or JavaScript.
- **Workflow:**
- Controller gets input
- Client program accesses the document.
- Interpreter displays the document.

- **Web Server:**
- The server stores all pages belonging to the site. Each time a client request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory: memory is faster to access than disk. Some popular Web servers include Apache and Microsoft Internet Information Server.
- **Uniform Resource Locator (URL):**

A client that wants to access a Web page needs the file name and the address. To facilitate the access of documents distributed throughout the world, HTTP uses locators. The uniform resource locator (URL) is a standard locator for specifying any kind of information on the Internet. The URL defines four things: protocol, host computer, port, and path.

A URL has four main parts

1: Protocol: The application used to retrieve the document (e.g., HTTP, FTP).

2Host Computer: The domain name where the information is located.

3Port: The server port number (optional).

4Path: The pathname of the file with the information.

Example: https://cisco.com

Protocol: https

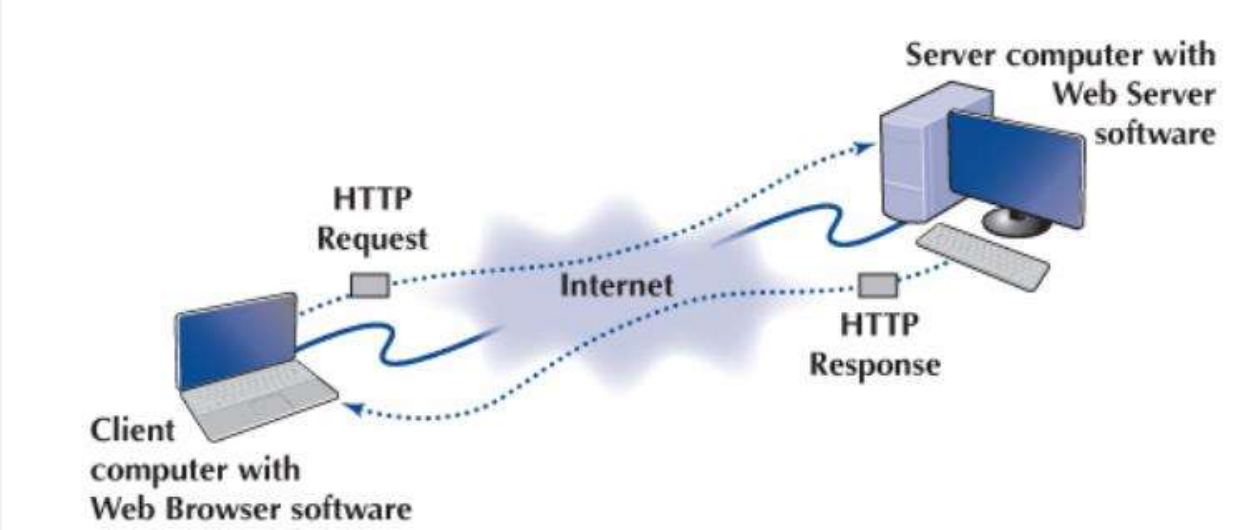
https://www.example.com/search?query=openai&lang=en

Host: cisco.com

Port: 443 (default for HTTPS)

Path: / (default root path)

- **How the web works**
- The Web is a good example of a two-tier client–server architecture. Each client computer needs an application layer software package called a Web browser. There are many different browsers, such as Microsoft's Internet Explorer. Each server on the network that will act as a Web server needs an application layer software package called a Web server. There are many different Webservers, such as those produced by Microsoft and Apache.
- To get a page from the Web, the user must type the Internet uniform resource locator (URL) for the page he or she wants (e.g., www.yahoo.com) or click on a link that provides the URL. The URL specifies the Internet address of the Web server and the directory and name of the specific page wanted. If no directory and page are specified, the Web server will provide whatever page has been defined as the site's home page



1.How It Works:

- When you type a URL (like yahoo.com)
- This request asks for the web page and includes the URL and other details.

2.Server Response:

- The web server receives the request, processes it, and sends back an HTTP response.
- This response includes the requested web page or an error message if something went wrong.

3.Request-Response for Multiple Files:

- Every file on a web page (like images and videos) requires its own request and response.
- For example, if a web page has two images, your browser will:
    1. Request the main web page.
    2. Receive and start displaying the page.
    3. Notice the two images.
    4. Send a request for each image.
    5. Receive each image in separate responses.

This back-and-forth communication ensures that all parts of the web page are loaded correctly.

- **HTTP and HTTPS**
- The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web. The protocol transfers data in the form of plain text, hypertext, audio, video, and so on. It is called the Hypertext Transfer Protocol because its efficiency allows its use in a hypertext environment where there are rapid jumps from one document to another.
- An HTTP request from a web browser to a web server consists of three parts. The first two parts are mandatory, while the third is optional:

1. **Request Line**:
    1. Begins with a command (e.g., GET).
    2. Specifies the requested web page.
    3. Ends with the HTTP version number that the browser supports.
    4. The version number ensures compatibility, preventing the web server from using an advanced HTTP version that the browser may not support.
2. **Request Header**:
    1. Contains optional information.
    2. Includes details such as the web browser being used (e.g., Internet Explorer) and the date.
3. **Request Body** (Optional):
    1. Contains information sent to the server.
    2. Typically includes data that the user has entered into a form.

- Example: https://www.example.com
- **When you type https://www.example.com into your browser and press Enter, your browser sends an HTTP GET request to the server hosting the example.com website. Here's what the request might look like:**
- **GET / HTTP/1.1**
- **Host: www.example.com**
- **User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36**
- **Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8**
- **Accept-Language: en-US,en;q=0.9**

- **Breakdown of the HTTP GET Request:**
- **Request Line:**
- **Command: GET**
- **Requested Resource: / (the root of the website, which typically loads the home page)**
- **HTTP Version: HTTP/1.1**

- **Request Header**:
- **Host**: Specifies the domain name of the server.
- **User-Agent**: Information about the browser making the request.
- **Accept**: Specifies the types of content that the client can process.
- **Accept-Language**: Specifies the preferred languages for the response

- Corresponding HTTP Response
- HTTP/1.1 200 OK
- Date: Mon, 21 Jun 2024 12:00:00 GMT
- Content-Type: text/html; charset=UTF-8
- Content-Length: 1256
- <!DOCTYPE html>
- <html> <head> <title>Example Domain</title> </head> <body> <h1>Example Domain</h1> This domain is for use in illustrative examples in documents. </body>
- </html>

- **Sending data to server using HTTP**
- Let's assume a user is filling out a registration form with fields for their username and password.
- POST /register HTTP/1.1
- Host: www.example.com
- Content-Type: application/x-www-form-urlencoded
- Content-Length: 39
- username=johndoe&password=securepassword

- **Server Response**
- HTTP/1.1 200 OK
- Date: Mon, 21 Jun 2024 12:15:00 GMT
- Content-Type: text/html; charset=UTF-8
- Content-Length: 58
- <html> <body> <h1>Registration Successful!</h1> </body> </html>

- **Electronic mail**

- Electronic mail, or e-mail, is the computerized version of writing a letter and mailing it at the local post office. Many people are so committed to using e-mail that if it were taken away tomorrow, some serious social and economic repercussions would be felt throughout the world.
- Many commercial e-mail programs are in existence, as well as a number of free ones that can be downloaded from the Internet. Although each e-mail program has its own unique foel and options, most offer the following services:

• Creating an e-mail message

• Sending an e-mail message to one recipient, multiple recipients, or a mailing list

• Receiving, storing, replying to, and forwarding e-mail messages

• Attaching a file such as a word processing document, a spreadsheet, an image, or a program to an outgoing e-mail message

- Most e-mail systems consist of two parts:
- (1) the user agent, which is the portion of the e-mail program that allows a user to create, edit, store, and forward e-mail messages; and
- (2) the message transfer agent, which is the portion of the e-mail program that prepares and transfers the e-mail message. Each transmitted e-mail also consists of two basic components: an envelope, which contains information describing the e-mail message, and the message, which is the contents of the envelope.
- **The Simple Mail Transfer Protocol (SMTP)** has always been the workhorse of the TCP/IP
- suite. However, SMTP has traditionally been limited to the delivery of simple text
- messages. In recent years, there has been a demand for the capability to deliver mail
- containing various types of data, including voice, images, and video clips. To satisfy this
- requirement, a new electronic mail standard, which builds on SMTP, has been defined: the
- Multi-Purpose Internet Mail Extension (MIME). In this section, we first examine SMTP,
- and then look at MIME.

•Simple mail transfer protocol

•**Definition**:
•SMTP stands for "Simple Mail Transfer Protocol."
•It is a connection-oriented, text-based network protocol from the internet protocol family.
•**OSI Model**:
•SMTP is located on the seventh layer of the OSI model: the application layer.
•**Purpose**:
•SMTP contains rules for correct communication between computers in a network.
•It is responsible for feeding and forwarding emails from sender to recipient.
•**Operation**:
•**Application Layer Protocol**: SMTP operates at the application layer.
•**Client-Side**:
    •The client who wants to send an email opens a TCP connection to the SMTP server.
    •The email is sent across this connection.
•**Server-Side**:
    •The SMTP server is always in listening mode.
    •It listens for TCP connections from clients on port 25.
    •Upon detecting a connection, the SMTP process initiates a connection on port 25.
•**Connection Establishment**:
    •After establishing the TCP connection, the client process sends the email instantly.

- SMTP has two types:

1. End-to-End Method:

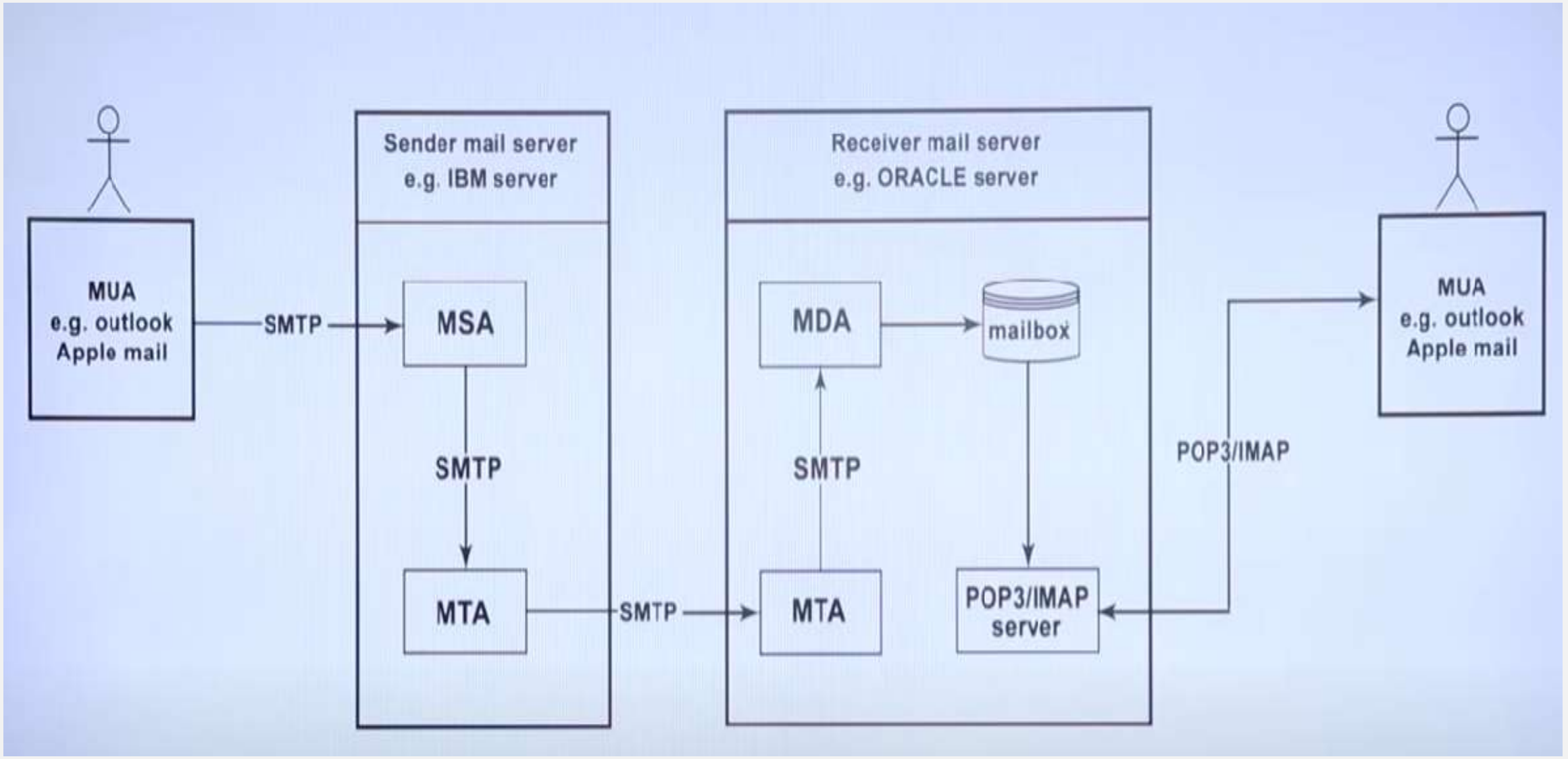
- Direct Communication: The email is sent directly from the sender's SMTP server to the recipient's SMTP server.
- Real-Time Delivery: Ensures immediate delivery of emails.
- Used For: Primarily used for sending emails over the Internet where the recipient's email server is directly reachable.
- Example: Personal Email Services, Business email

2. Store-and-Forward Method:

- Intermediate Servers: Emails are temporarily stored on intermediate SMTP servers before being forwarded to the final recipient's server.
- Delayed Delivery: Useful when the recipient's server is not immediately available.
- Used For: Situations where immediate delivery is not critical, such as when the recipient's server is offline or busy.
- Example: Corporate email, ISP email

- **Model of SMTP system**
- **User Interaction**
- **User Agents (UAs)**:
  - These are email clients like Microsoft Outlook, Netscape, and Mozilla Thunderbird.
  - Users interact with these programs to send and receive emails.
- **Behind the Scenes**
- **Message Transfer Agent (MTA)**:
  - This is the software that actually sends the email over the internet.
  - Users do not directly interact with the MTA; it's set up by the system administrator.
  - Examples of MTAs include Sendmail, Postfix, and Microsoft Exchange.
- **How It Works**

1. **Sending Emails**:
   1. When a user sends an email using their email client (UA), the email client communicates with the local MTA.
   2. The MTA then handles the job of sending the email to the recipient's MTA using TCP (Transmission Control Protocol).
2. **Handling Delivery**:
   1. If the recipient's email server is temporarily unavailable, the MTA keeps the email in a queue and retries sending it until successful.
   2. This ensures that emails are eventually delivered even if there are temporary network issues.
3. **Receiving Emails**:
   1. Once the recipient's MTA receives the email, it stores it in the recipient's mailbox on the server.
   2. The user can then use their email client to access and read the email.



MUA: Mail User Agent
MSA: Mail submission Agent
MTA: Mail Transfer Agent
MDA: Mail Delivery Agent

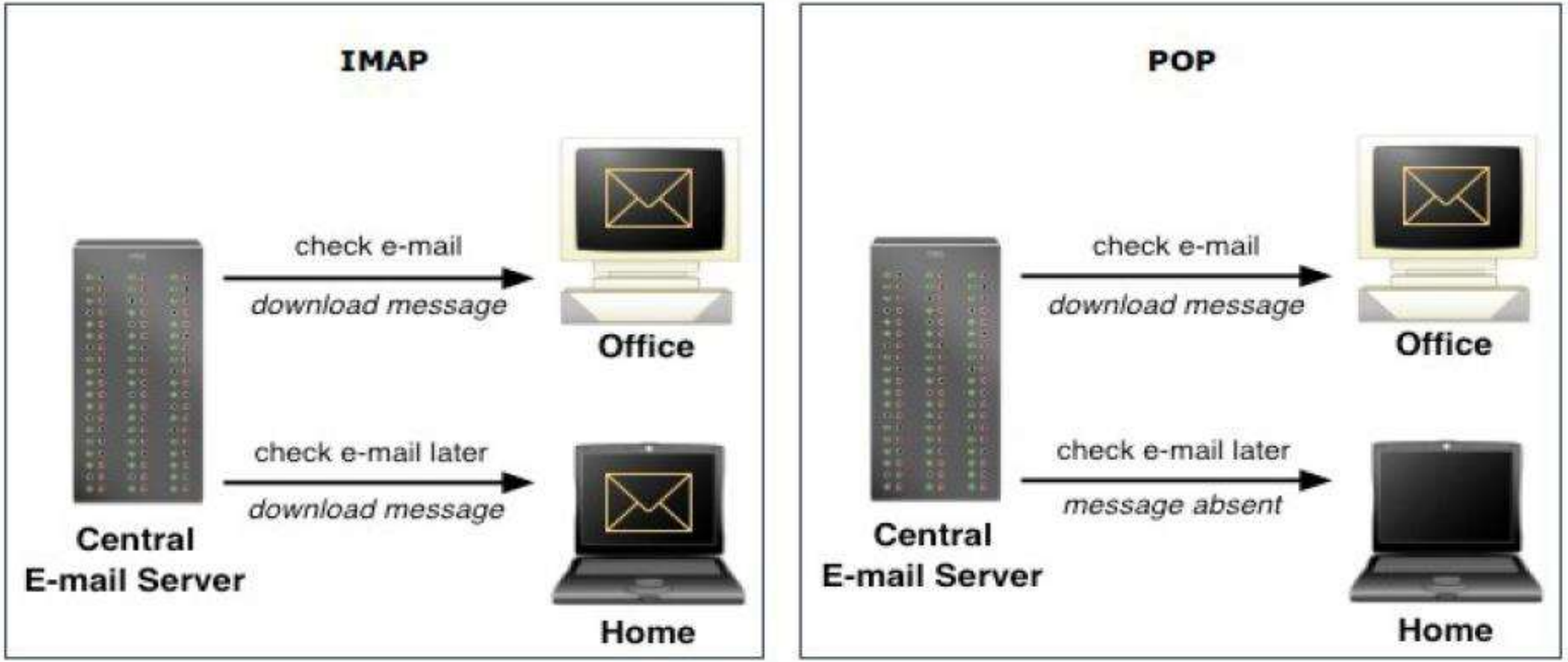
- **Internet mail access protocol (IMAP)**
- An Internet standard protocol for storing and retrieving messages from Simple Mail
- Transfer Protocol (SMTP) hosts.
- **How It Works**
- • SMTP provides the underlying message transport mechanism for sending e-mail over the Internet, but it does not provide any facility for storing and retrieving messages.
- SMTP hosts must be continuously connected to one another, but most users do not have a dedicated connection to the Internet.
- IMAP4 provides mechanisms for storing messages received by SMTP in a receptacle called a mailbox.
- IMAP4 server stores messages received by each user until the user connects to download and read them using an IMAP4 client such as Microsoft Outlook Express

- Specifically, IMAP4 allows users to
- Access multiple folders, including public folders
- Create hierarchies of folders for storing messages
- Leave messages on the server after reading them so that they can access the messages again from another location
- Search a mailbox for a specific message to download
- Flag messages as read
- Selectively download portions of messages or attachments only
- Review the headers of messages before downloading them

- **Post Office Protocol version 3 (POP3)**
- • An Internet standard protocol for storing and retrieving messages from Simple Mail Transfer Protocol (SMTP) hosts.
- **How It Works**
- SMTP provides the underlying transport mechanism for sending e-mail messages over the Internet, but it does not provide any facility for storing messages and retrieving them.
- SMTP hosts must be continuously connected to one another, but most users do not have a dedicated connection to the Internet.
- Post Office Protocol version 3 (POP3) provides mechanisms for storing messages sent to each user and received by SMTP in a receptacle called a mailbox.
- A POP3 server stores messages for each user until the user connects to download and read them using a POP3 client such as Microsoft Outlook 98, Microsoft Outlook Express, or Microsoft Mail and News.
- To retrieve a message from a POP3 server, a POP3 client establishes a Transmission Control Protocol (TCP) session using TCP port 110, identifies itself to the server, and then issues a series of POP3 commands

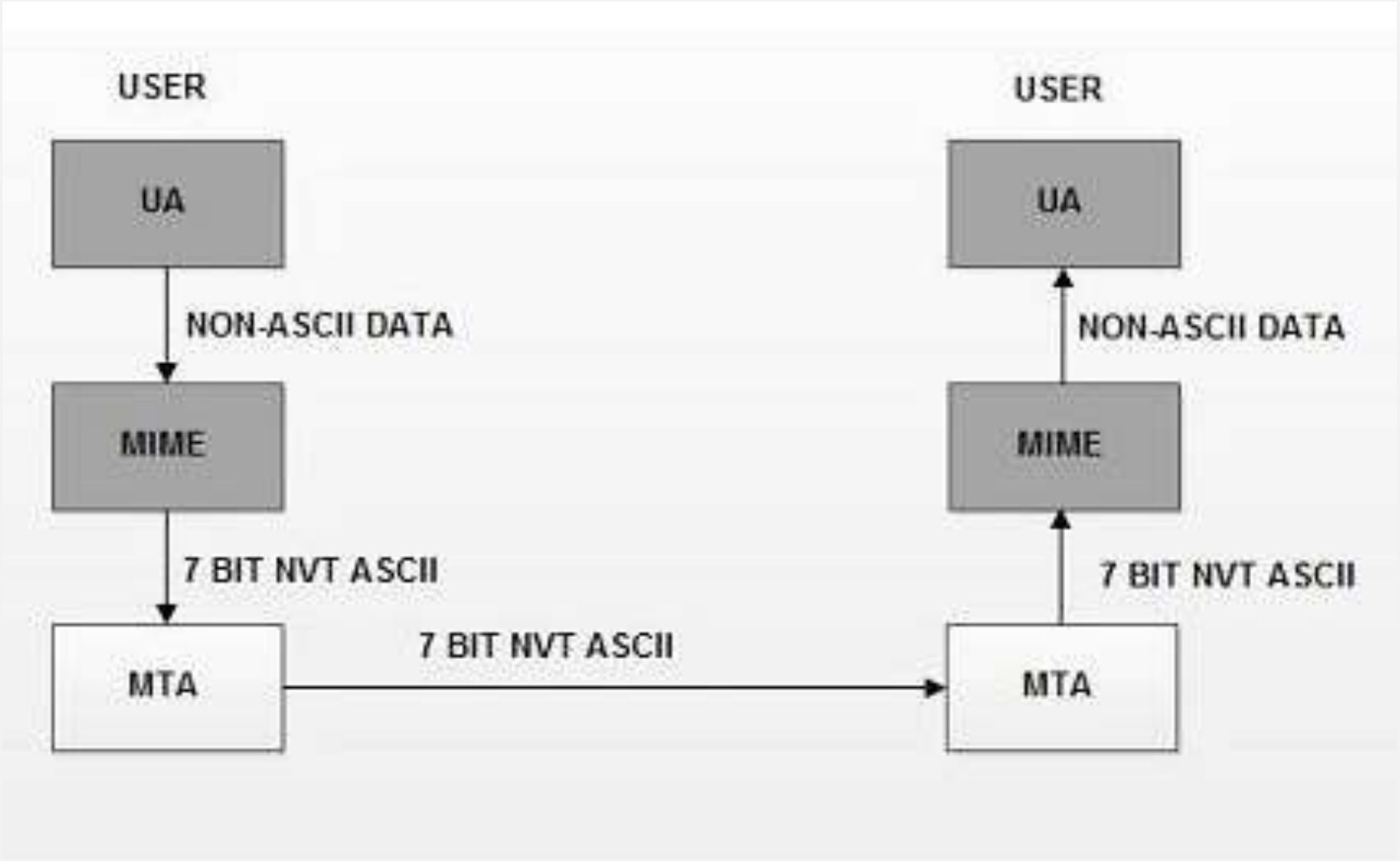
IMAP VS POP

- The main difference, as far as we are concerned here, is the way in which IMAP or POP controls your e-mail inbox.
- When you use IMAP you are accessing your inbox on the mail server. IMAP does not actually move messages onto your computer. You can think of an e-mail program using IMAP as a window to your messages on the server. Although the messages appear on your computer while you work with them, they remain on the central mail server.
- POP does the opposite. Instead of just showing you what is in your inbox on the U's mail server, it checks the server for new messages, downloads all the new messages in your inbox onto your computer, and then deletes them from the server. This means that every time you use POP to view your new messages, they are no longer on the central mail server.


IMAP
check e-mail
download message
Office
check e-mail later
download message
Central
E-mail Server
Home
POP
check e-mail
download message
Office
check e-mail later
message absent
Central
E-mail Server
Home

- **multipurpose internet mail extensions(MIME)**
- Multipurpose Internet Mail Extension (MIME) is a standard that was proposed by Bell Communications in 1991 in order to expand the limited capabilities of email.
- MIME is a kind of add-on or a supplementary protocol that allows non-ASCII data to be sent through SMTP. It allows the users to exchange different kinds of data files on the Internet: audio, video, images, application programs as well.

- **Why do we need MIME?**
- Limitations of Simple Mail Transfer Protocol (SMTP):
- SMTP has a very simple structure
- Its simplicity however comes with a price as it only sends messages in NVT 7-bit ASCII format.
- It cannot be used for languages that do not support 7-bit ASCII format such as French, German, Russian, Chinese and Japanese, etc. so it cannot be transmitted using SMTP. So, in order to make SMTP more broad, we use MIME.
- It cannot be used to send binary files or video or audio data


USER
USER
UA
UA
NON-ASCII DATA
NON-ASCII DATA
MIME
MIME
7 BIT NVT ASCII
7 BIT NVT ASCII
MTA
7 BIT NVT ASCII
MTA

- **Telnet:**
- Telnet is a protocol that allows user to log in to remote computers (called hosts) over a TCP/IP network (such as the internet). Using telnet client software on your computer, you can make s connection to a telnet server (that is, the remote host). Once your telnet client establishes a connection to the remote host, your client becomes a virtual terminal, allowing you to communicate with the remote host from your computer. In most cases, you'll need to log into the remote host, which requires that you have an account on that system. Occasionally, you can log in as guest or public without having an account.

- Although Telnet requires a logging name and password, it is vulnerable to hacking because it sends all data including the password in plaintext (not encrypted). A hacker can eavesdrop and obtain the logging name and password. Because of this security issue, the use of Telnet has diminished in favor of another protocol, Secure Shell (SSH).
- The simple plaintext architecture of Telnet allows us to explain the issues and challenges related to the concept of remote logging, which is also used in SSH when it serves as a remote logging protocol.
- Telnet clients are available for all major operating systems. Command-line telnet clients are built into most versions of macOS, Windows, Unix, and Linux.

- How telnet works?
- • Telnet also operates on the client/server principle. The local computer uses a telnet client program and the remote, or host, computer uses a telnet server program.
- Telnet uses software installed on your computer, to create a connection with the remote host.
- The telnet client (local computer), at your command, will send a request to the telnet server (remote host).
- The server will reply asking for a user name and password. If accepted, the Telnet client will establish a connection to the host, thus making your computer a virtual terminal and allowing you complete access to the host's computer.
- Telnet requires the use of a user name and password, which means you need to have previously set up an account on the remote computer.
- In some cases, however, computers with telnet will allow guests to log on with restricted access.

- **• Instant Messaging**
- • One of the fastest growing Internet applications has been instant messaging (IM). With IM, you can exchange real-time typed messages or chat with your friends. Some IM software also enables you to verbally talk with your friends in the same way as you might use the telephone or to use cameras to exchange real-time video in the same way you might use a videoconferencing system. Several types of IM currently exist, including Google Talk and AOL Instant Messenger.
- IM works in much the same way as the Web. The client computer needs an IM client software package, which communicates with an IM server software package that runs on a server. When the user connects to the Internet, the IM client software package sends an IM request packet to the IM server informing it that the user is now online. The IM client software package continues to communicate with the IM server to monitor what other users have connected to the IM server. When one of your friends connects to the IM server, the IM server sends an IM packet to your client computer so that you now know that your friend is connected to the Internet. The server also sends a packet to your friend’s client computer so that he or she knows that you are on the Internet

- **How IM Works:**

1. **IM Client Software:**
    1. Installed on the user's computer.
    2. Communicates with the IM server software.
2. **IM Server Software:**
    1. Runs on a server.
    2. Manages connections and communications between users.
3. **Connecting to the Internet:**
    1. When the user goes online, the IM client software sends a request to the IM server, indicating the user is now online.
4. **User Monitoring:**
    1. The IM client continuously communicates with the IM server to check which users are online.
5. **Notification of Friends:**
    1. When a friend connects to the IM server, the server notifies your IM client, and vice versa.
    2. This way, both users are aware of each other's online status.

- **Videoconferencing:**
- • Video conferencing is a visual communication session between two or more users regardless of their location, featuring audio and video content transmission in real time. In broad terms, video conferencing software is a solution that allows two or more persons to talk and see each other remotely. Initially video conferencing software only enabled users to make video calls or hold group video conferences. However, as technology advanced, video conferencing software's acquired plenty of helpful tools and features for remote communication and learning.
- Today, video conferencing is rather a tool for integrated into video collaboration and unified communications platforms that also offer screen sharing, slideshow, recording, instant messaging. project management tools, telephony integration among other features. As an increasing number of companies switch to remote work, video conferencing in gradually becoming the new normal, shaping the way we communicate, learn and work.
- The transmission of video requires a lot of network capacity. Most videoconferencing uses data compression to reduce the amount of data transmitted. Surprisingly, the most common complaint is not the quality of the video image but the quality of the voice transmissions. Special care needs to be taken in the design and placement of microphones and speakers to ensure quality sound and minimal feedback

- Most videoconferencing systems were originally developed by vendors using different formata, so many products were incompatible. The best solution was to ensure that all hardware and software used within an organization was supplied by the same vendor and to hope that any other organizations with whom you wanted to communicate used the same equipment. Today, three standards are in common use: H.320, H.323, and MPEG-2 (also called ISO 13818-2). Each of these standards was developed by different organizations and is supported by different products. They are not compatible, although some application software packages understand more than one standard.
- H.320 is designed for room-to-room videoconferencing over high-speed telephone lines.
- H.323 is a family of standards designed for desktop videoconferencing and just simple audio conferencing over the Internet.
- MPEG-2 is designed for faster connections, such as a LAN or specially designed, privately operated WAN
- Webcasting is a special type of one-directional videoconferencing in which content is sent from the server to the user. The developer creates content that is downloaded as needed by the users and played.