Lecture 1: Client-Server Architectures

- Adopted from the notes of:
- Dr. Hamed Hamzeh
- January 23, 2025

Module Lecturers

- Cassim Farook
- Saadh Jawwadh
- Ruwan Egodawatte
- Salitha Perera

Module Tutors

- Jiehfeng Hsu
- Mahfoos Ahamed
- Santhusha Mallawatantri
- Ammar Raneez
- Adshayani Pirapaharan
- Mohamed Shazeen
- Vathila De Silva
- Krishnamoorthy Caucidheesan

Assessment

Lab-based practical (40%)

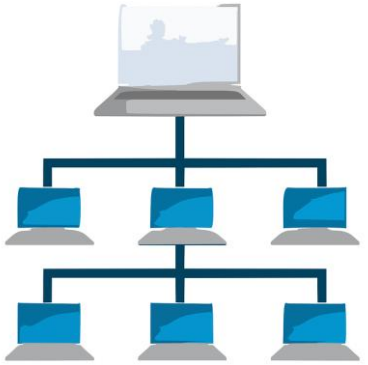
- Includes socket programming
- You will be given by a ZIP file that you should import to NetBeans and work on it.
- You will need to complete the codes using given questions inside the codes
- Finally, you will need to run your project and then export it as a ZIP file and submit it to BB

Coursework (60%)

- You will work on a scenario in which you need to develop a REST API using JAX-RS
- You will need to prepare a report and video with clear explanation of all steps that you have taken to implement it.

Let’s get started … !

Computer Networks



Definition:	• A computer network is a set of interconnected computers that communicate with each other and share resources.
Purpose:	• Facilitate communication, share resources, and provide connectivity between devices.
Types of Networks:	• LAN (Local Area Network), WAN (Wide Area Network), MAN (Metropolitan Area Network), PAN (Personal Area Network), etc.

Goal of Networking

Communication: Enable seamless communication and data exchange between devices.

Resource Sharing: Facilitate the sharing of resources such as files, printers, and applications.

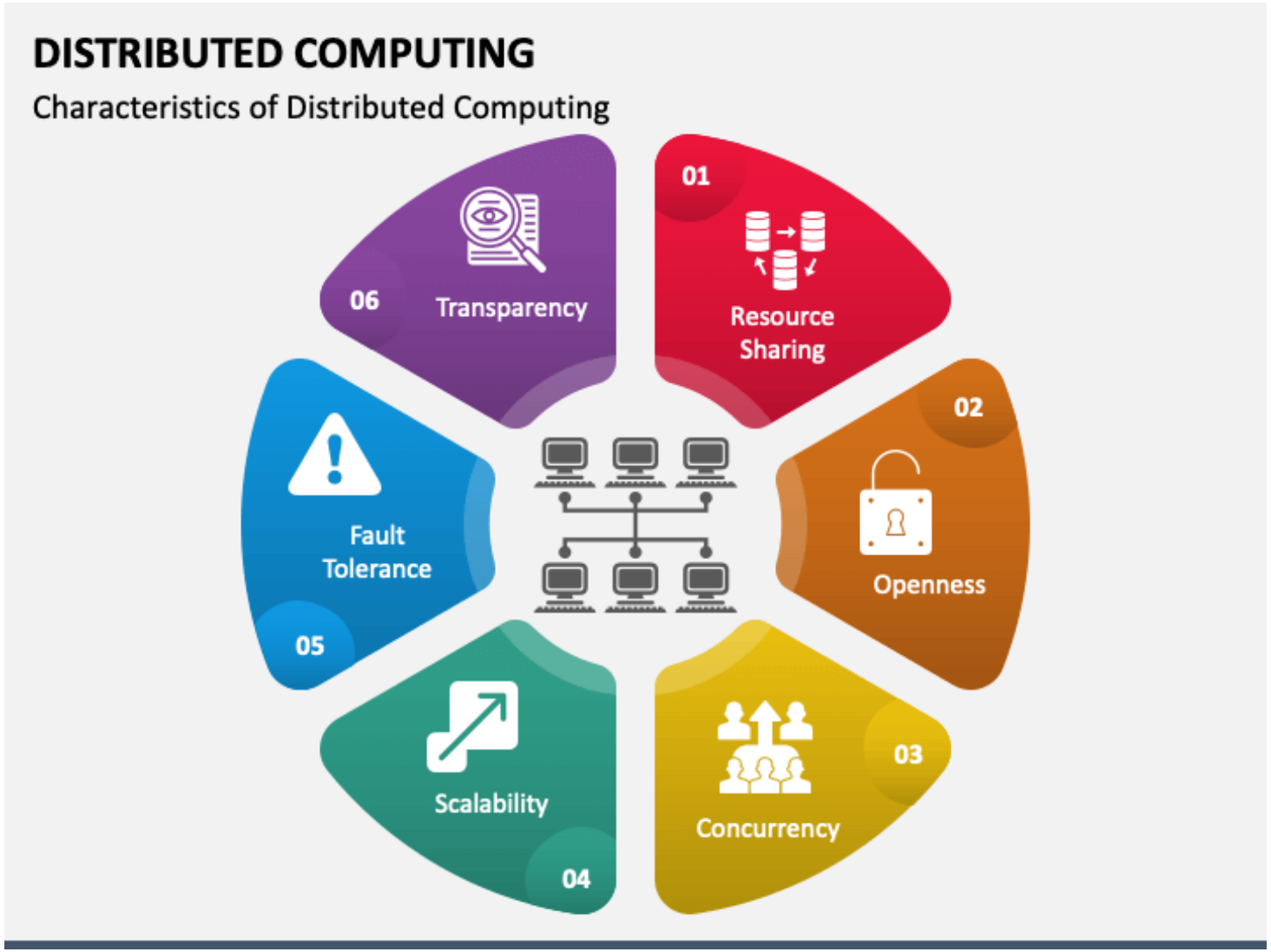
Reliability: Ensure reliable and efficient data transfer.

Scalability: Allow for easy expansion and addition of new devices.

Cost Efficiency: Streamline processes and reduce costs through shared resources.

Distributed Systems

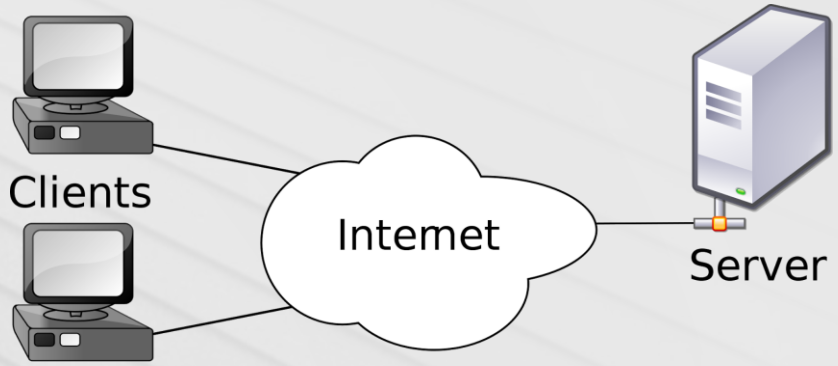
- **Definition:** A collection of independent computers that appears to the user as a single coherent system.
- **Key Characteristics:** Multiple computers (nodes)
  - Networked together
  - Work together to achieve a common goal
  - Illusion of a single system



DISTRIBUTED COMPUTING

How does Distributed Computing Work?

Client-Server Architecture



Definition:	Client-server architecture is a network architecture where tasks are divided between clients (service requesters) and servers (service providers).
Roles:	Clients make requests, and servers fulfill those requests.
Advantages:	Centralized control, scalability, easier maintenance, and resource sharing.
Examples:	Web browsing (client - browser, server - website), email (client - email client, server - email server).

Client-Server Architecture Components

Client-side Components:

- **User Interface (UI):** The part of the application the user interacts with.
- **Client-Side Scripting:** Code executed on the client's browser.
- **Web Browser:** Platform for accessing and displaying web content.

Server-side Components:

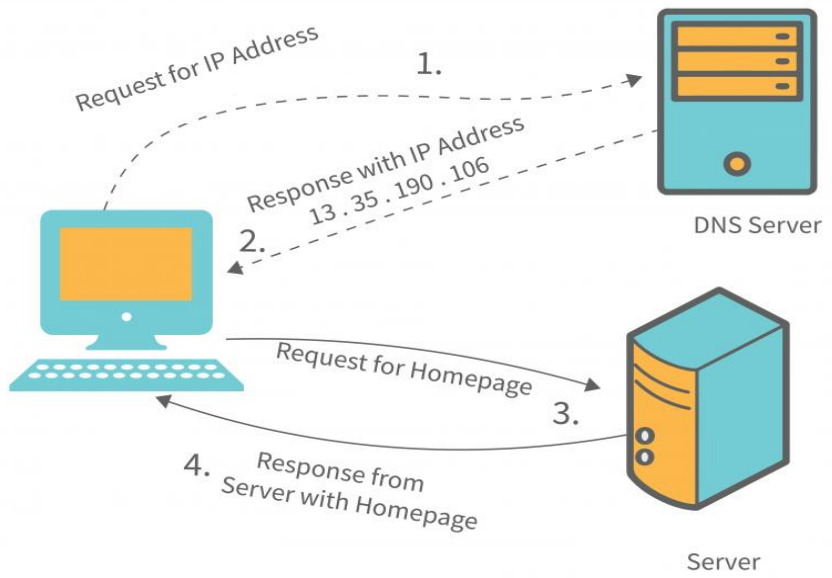
- **Application Server:** Manages application logic and business rules.
- **Database Server:** Stores and manages data.
- **Web Server:** Manages and delivers web content to clients.

Communication Protocols:

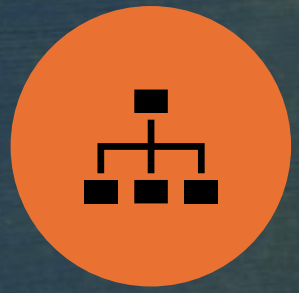
- **HTTP/HTTPS:** HyperText Transfer Protocol for secure communication.
- **TCP/IP:** Transmission Control Protocol/Internet Protocol for data transmission.
- **RESTful APIs:** Representational State Transfer for web services.

How Does Client-Server Architecture Work?

Communication Flow:	Clients send requests to the server. Servers process requests and send back responses to clients.
Request-Response Model:	Clients request services or resources. Servers respond by providing the requested services or resources.
Stateless Nature:	Each request from the client is independent, and the server doesn't retain information about previous requests.
Examples:	Web browsers (clients) requesting web pages from servers. Database queries where the client requests data from a database server.



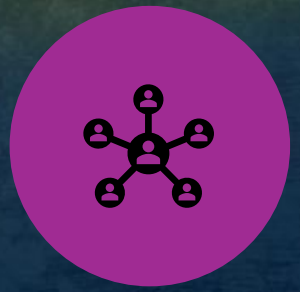
Types of Client-Server Models



One-Tier Model

Two-Tier (Client-Server) Model

Three-Tier Model



N-Tier Model:

The Standard Logical Layers/Tiers

Presentation Tier:

- **Purpose:** User interface and interaction.
- **Components:** UI elements, client-side scripts, and user input.
- **Example:** Web browsers, mobile app interfaces.

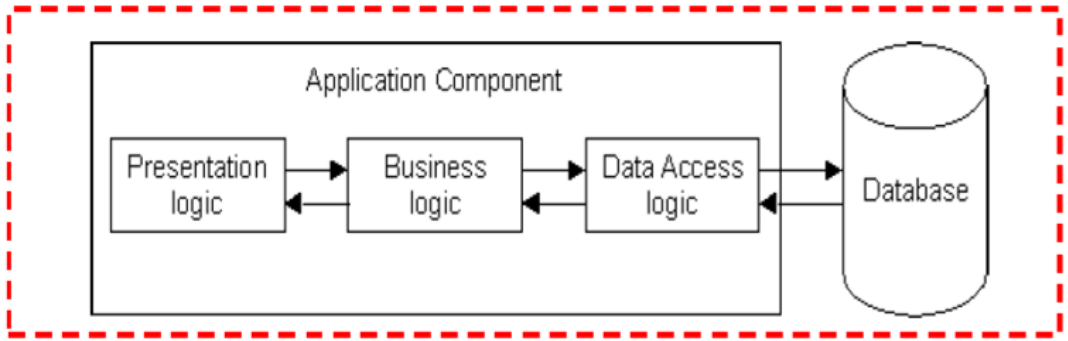
Application (Logic) Tier:

- **Purpose:** Business logic and application processing.
- **Components:** Application server, middleware, and processing logic.
- **Example:** Processing orders, calculations, validations.

Data (Persistence) Tier:

- **Purpose:** Storage and retrieval of data.
- **Components:** Database server, data storage, and management systems.
- **Example:** Database servers, file systems.

1-Tier Architecture



Definition:

- Also known as Single-Tier architecture.

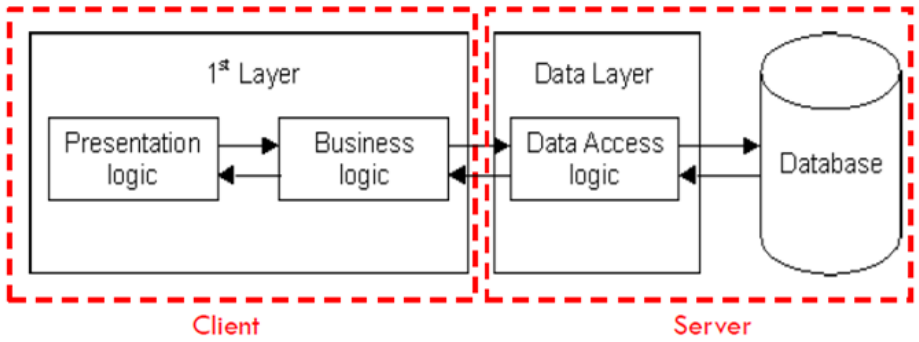
Characteristics:

- Application logic, presentation, and data management all reside on a single machine.
- Typically used for simple applications with limited functionality.

Example:

- Standalone desktop applications with no network connectivity.

2-Tier Architecture



Definition:

- Also known as Client-Server architecture.

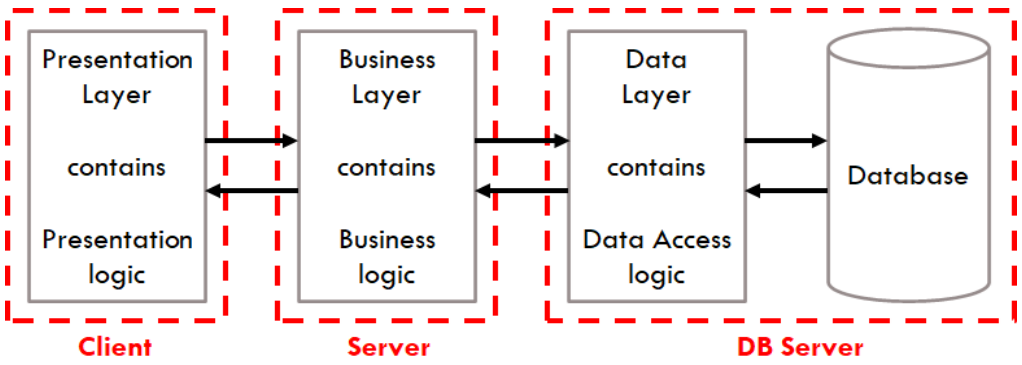
Characteristics:

- Separation of client and server roles.
- Clients handle the user interface and application logic.
- Servers manage data storage and processing.

Example:

- Traditional client-server applications, database applications.

3-Tier Architecture



Definition:

- Adds an application server layer to the 2-Tier architecture.

Characteristics:

- Clients handle presentation.
- Application servers manage application logic.
- Servers manage data storage and retrieval.

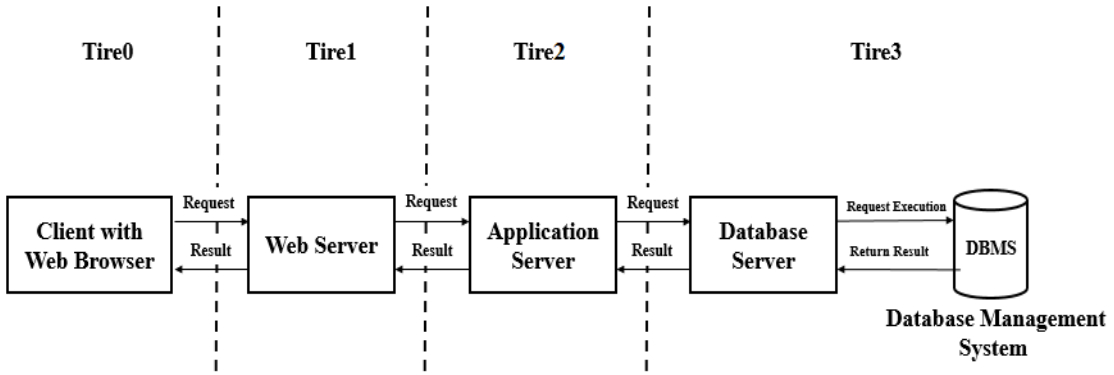
Advantages:

- Improved scalability, easier maintenance, and flexibility.

Example:

- Web-based applications with client-side interfaces, separate application servers, and database servers.

N-Tier Architecture



Definition:

- Generalization of the 3-Tier architecture.

Characteristics:

- Multiple tiers for specific functionalities.
- Enhances scalability, flexibility, and separation of concerns.

Examples:

- Large-scale enterprise applications with multiple specialized layers.
- Cloud-native architectures with microservices.

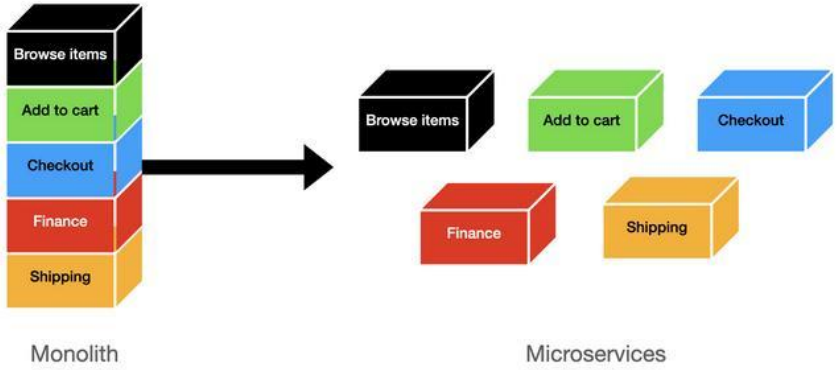
Introduction to Microservices Architecture

Definition:

- Microservices architecture is an approach to software development where an application is composed of small, independent services that communicate over well-defined APIs.

Key Characteristics:

- Decentralized and independently deployable services.
- Services can be developed, deployed, and scaled independently.
- Each service has its own database or data storage.



Advantages of Microservices Architecture

01

Scalability: Each microservice can be scaled independently, allowing for efficient resource utilization.

02

Flexibility and Agility: Independent development and deployment enable faster updates and feature releases.

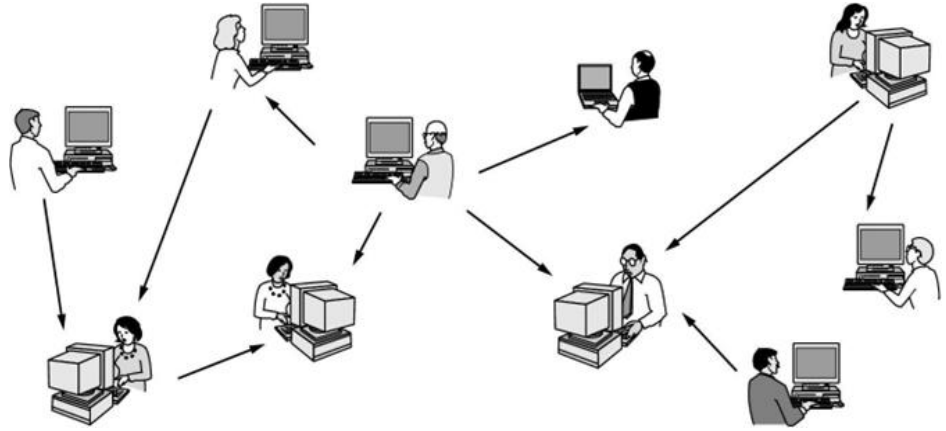
03

Fault Isolation: Failures in one microservice do not necessarily affect the entire system.

04

Technology Diversity: Different microservices can be developed using different technologies best suited for their specific functions.

Peer-to-Peer System



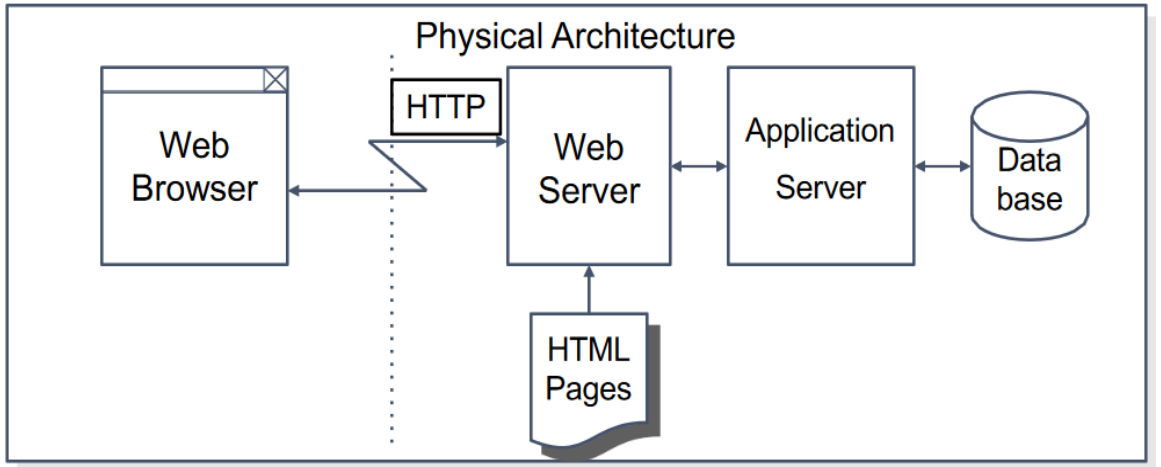
Definition:

- A Peer-to-Peer (P2P) system is a decentralized network where each device (peer) can act as both a client and a server.

Characteristics:

- Peers share resources directly without a central server.
- Distributed nature, with no single point of control.
- Examples: File-sharing networks like BitTorrent, decentralized cryptocurrencies.

Thin Client



Definition:

- A thin client is a lightweight device that relies on a central server for processing and storage.

Characteristics:

- Minimal processing power and storage.
- Relies heavily on the server for application execution and data storage.
- Often used in virtual desktop environments and cloud computing.

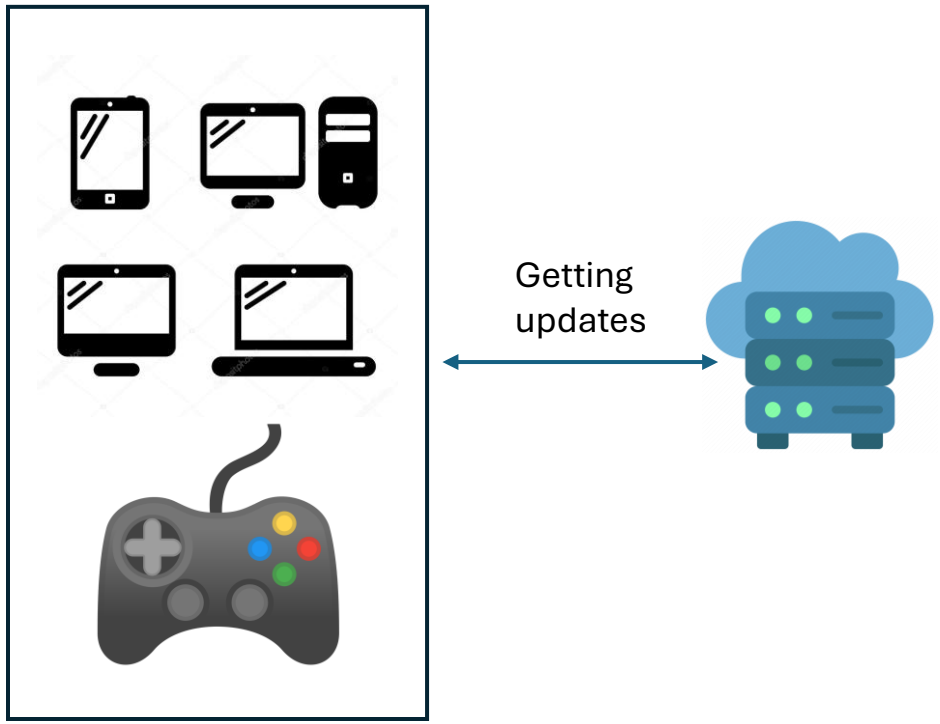
Thick/Fat Client

Definition:

- A thick or fat client is a device with substantial processing power and storage capabilities.

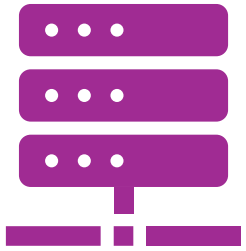
Characteristics:

- Executes applications locally.
- Stores a significant amount of data locally.
- Common in standalone desktop applications and traditional computing models.



Video games installed on PC/Laptop

Client-Server Architecture VS Peer-to-Peer

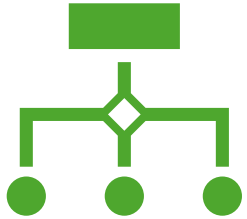


Client-Server:

Centralized model with distinct client and server roles.

Efficient resource management and centralized control.

Scalability may require significant server upgrades.



Peer-to-Peer:

Decentralized model with peers acting as both clients and servers.

No central point of control, potentially more resilient.

Resource usage depends on the number of peers.

Network Classification

Based on Size and Scope:

- **Local Area Network (LAN):** Limited geographic area, often within a building or campus.
- **Metropolitan Area Network (MAN):** Covers a larger area, like a city or metropolitan region.
- **Wide Area Network (WAN):** Spans across large distances, possibly across cities or countries.

Based on Access Method:

- **Peer-to-Peer Networks:** Direct communication between devices without a central server.
- **Client-Server Networks:** Clients make requests, and servers fulfill them.

Types of Services

Web Server

Definition:

- A web server is software that processes client requests and delivers web pages to users' browsers.

Functionality:

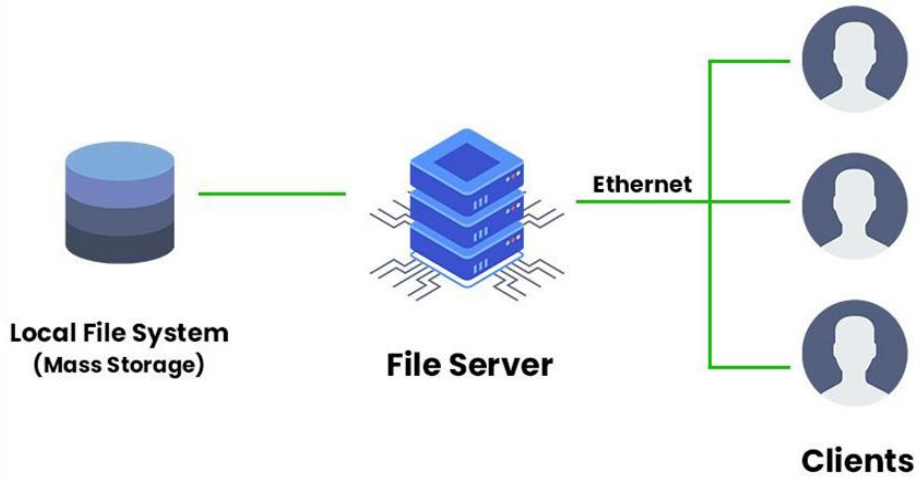
- Handles HTTP requests.
- Serves static content (HTML, images, etc.) and may support dynamic content through application servers.

Example:

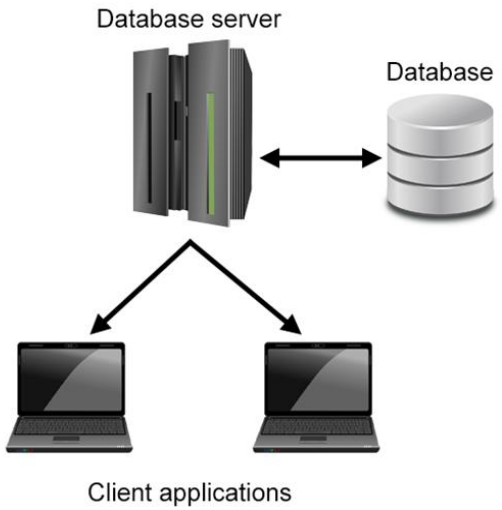
- Apache HTTP Server, Nginx, Microsoft IIS.

File Server

- **Definition:**
  - A file server is a dedicated server that manages and provides access to files for clients on a network.
- **Functionality:**
  - Centralized file storage and management.
  - Allows users to access, modify, and share files.
- **Example:**
  - Windows File Server, FileZilla Server.



Database Server



Definition:

- A database server manages and provides access to databases and their data.

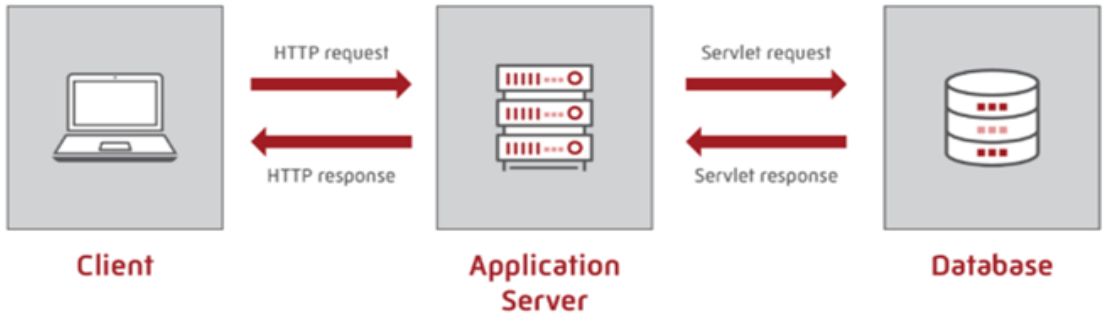
Functionality:

- Stores, retrieves, and manipulates data.
- Supports query processing and ensures data integrity.

Example:

- MySQL Server, Microsoft SQL Server, Oracle Database.

Application Server



Definition:

- An application server executes and manages applications, providing services to clients.

Functionality:

- Runs business logic, application code, and middleware services.
- Enables communication between the client and database server.

Example:

- Apache Tomcat, JBoss (Java EE Application Servers), Microsoft ASP.NET.