UNIT II

CLIENTS,SERVERS, TRANSACTIONS AND OPERATING SYSTEMS

CLIENTS,SERVERS, TRANSACTIONS AND OPERATING SYSTEMS

The Anatomy of a server program, Operating System Basic and Extended Services for server applications, Server Scalability, Client Anatomy, Client/server Hybrids - Comparison of two and three tier- Client side, Server side and Middleware side- Hardware and Software requirements- Transaction servers-TP lite Vs TP Heavy

Client–Server Model

- Clients request services
- Servers provide services
- Communication via network

What is a Client/Server System (1)

Client/server computing may be viewed as an extension of modular programming.

A modular framework simplifies development and maintenance.

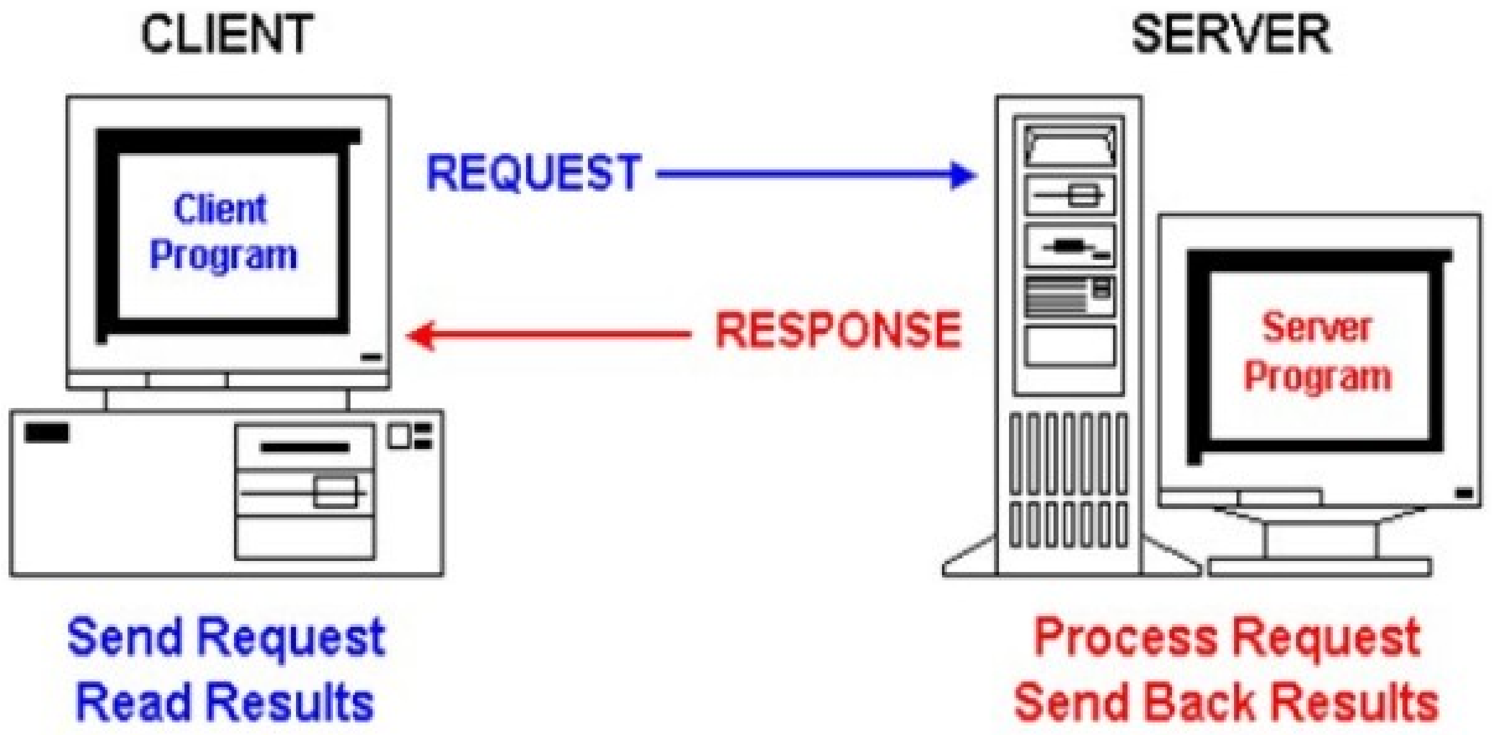
As modules become obsolete, they can be replaced or upgraded without replacing the entire system.

What is a Client/Server System (2)

Client server computing recognizes that all modules do not have to be executed with the same memory location—the calling module or "client" can request information from the service module or "servers."

Because client/server applications rely on communications networks for processing and retrieving information, the management of the entire network is absolutely critical.

What is a Client/Server System (3)



What is a Client Process

The client's responsibility is usually to:

- Handle the user interface
- Translate the user's request into the desired protocol
- Send the request to the server
- Wait for the server's response
- Translate the response into "human-readable" results
- Present the results to the user

What is a Server Process

The server's functions include:

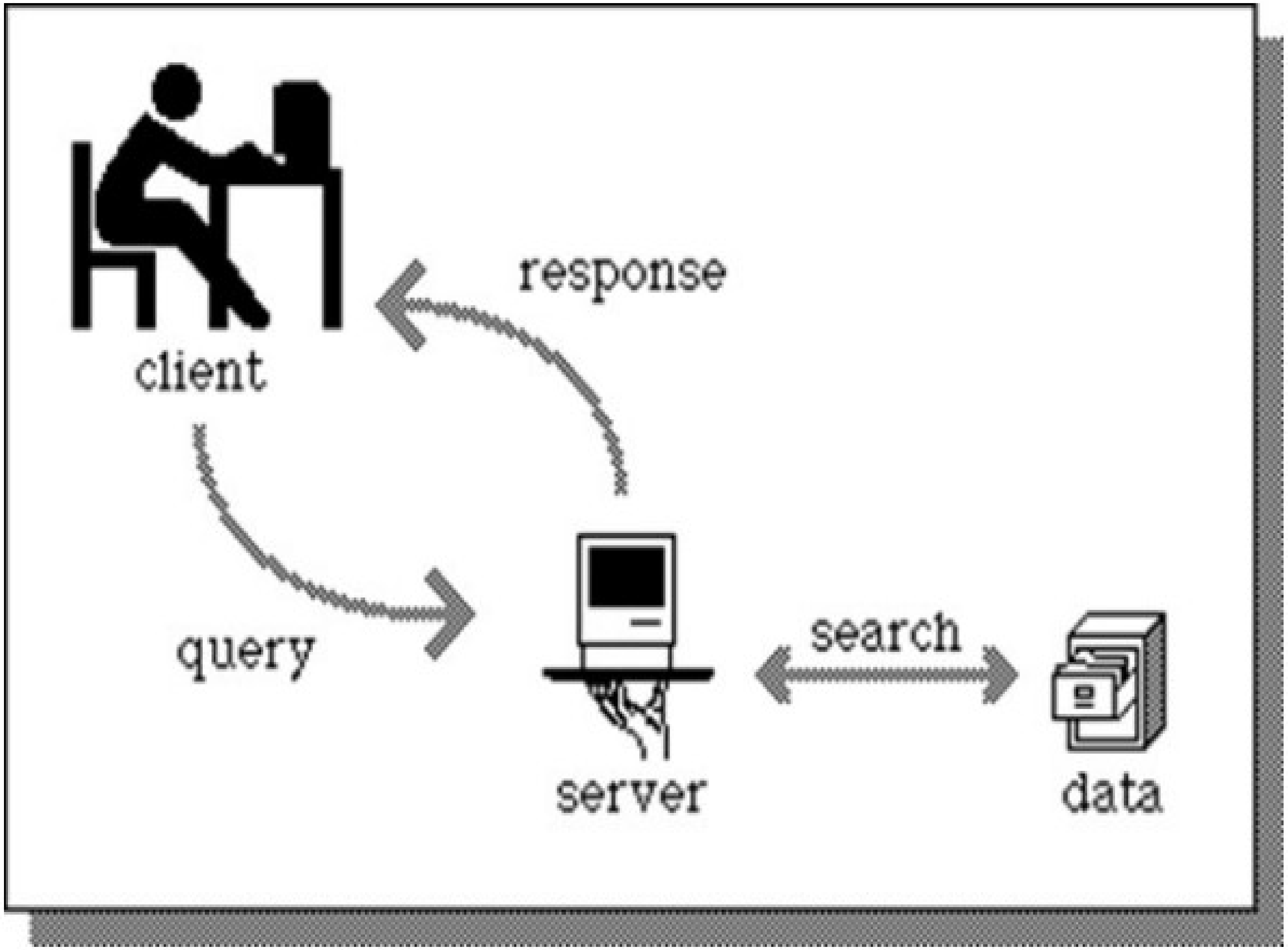
- Listen for a client's query
- Process that query
- Return the results back to the client

Client/Server Interaction (1)

A typical client/server interaction goes like this:

1. **The user runs client software to create a query**
2. **The client connects to the server**
3. **The client sends the query to the server**
4. **The server analyzes the query**
5. **The server computes the results of the query**
6. **The server sends the results to the client**
7. **The client presents the results to the user**
8. **Repeat as necessary**

Client/Server Interaction (2)



What is a Network

A group of two or more computer systems linked together. There are many types of computer networks, including:

Local Area Networks (LANs) : The computers are geographically close together (that is, in the same building).

Wide Area Networks (WANs) : The computers are farther apart and are connected by telephone lines or radio waves.

Metropolitan Area Networks (MANs): A data network designed for a town or city.

Architecture

Networks can be broadly classified as using either a peer-to-peer or client/server architecture.

Peer-to-Peer architecture : A type of network in which each workstation has equivalent capabilities and responsibilities. The networks are generally simpler, but they usually do not offer the same performance under heavy loads.

Client/Server architecture : Each computer on the network is either a *client* or a *server*. Servers are powerful computers dedicated to managing disk drives (*file servers*), printers (*print servers*), or network traffic (*network servers)* . Clients are PCs or workstations on which users run applications.

System Architectures : Functional Layers

Every client/server application contains three functional units:

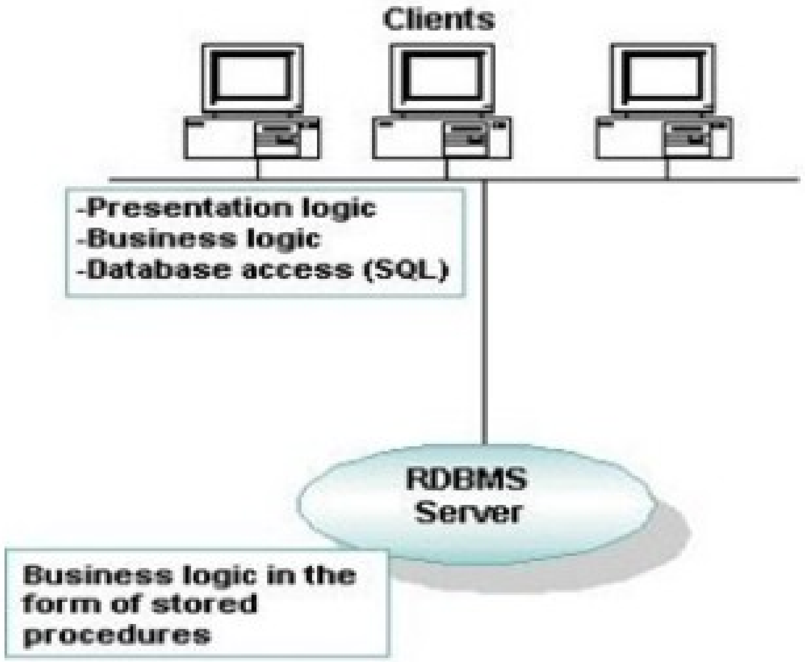
- **Presentation logic** or **user interface** (for example, ATM machines)
- **Business logic** (for example software that enables a customer to request an account balance)
- **Data** (for example, records of customer accounts)

These functional units can reside on either the client or on one or more servers in your application. Which of the many possible variations you choose depends on how you split the application and which middleware you use to communicate between the tiers.

System Architectures : Two-tier and Three-tier (1)

2-Tiers Architecture

In 2-tier client/server applications, the business logic is buried inside the user interface on the client or within the database on the server in the form of stored procedures. Alternatively, the business logic can be divided between the client and server. File servers and database servers with stored procedures are examples of 2-tier architecture.



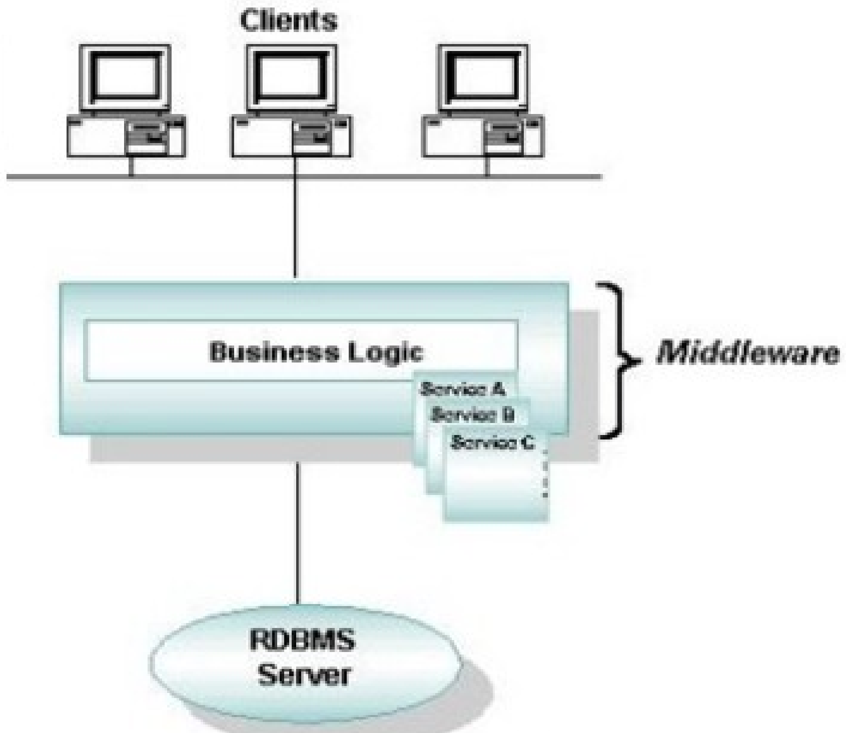
2-TIER CLIENT/SERVER

Two or more operating systems
One or more programming languages
Local and remote databases
Networking/communication issues
Inter-program communications

System Architectures : Two-tier and Three-tier (2)

3-Tiers Architecture

In 3-tier client/server applications, the business logic resides in the middle tier, separate from the data and user interface. In this way, processes can be managed and deployed separately from the user interface and the database. Also, 3-tier systems can integrate data from multiple sources.



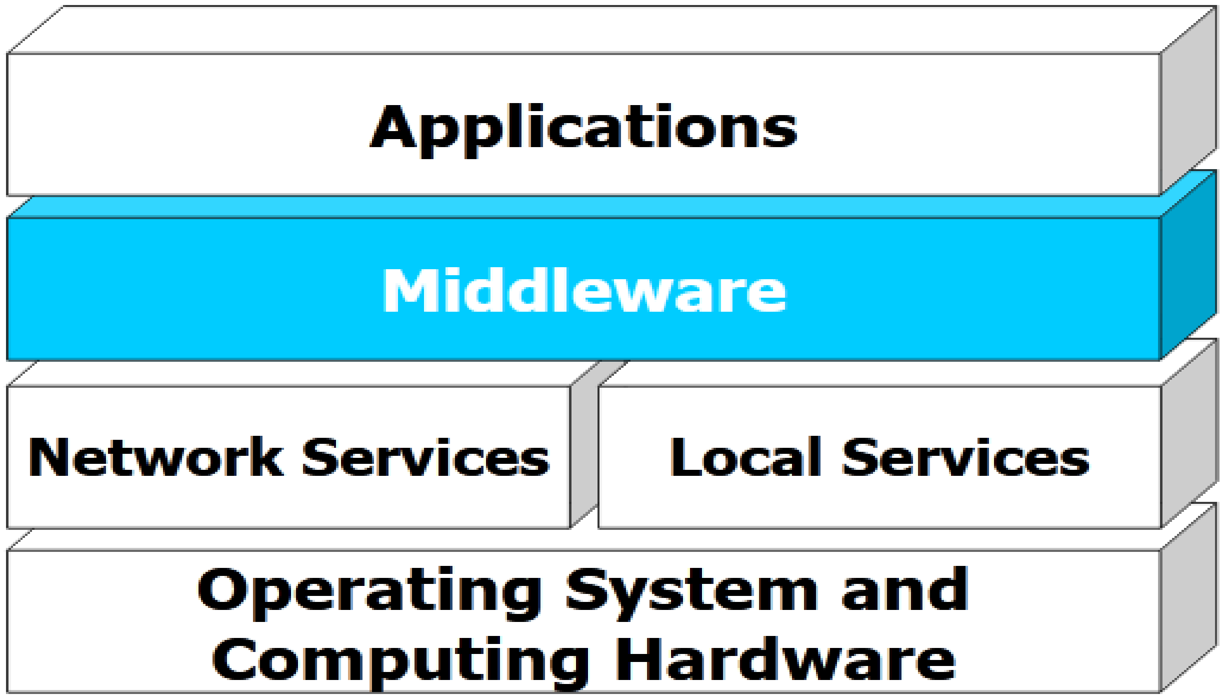
3-TIER CLIENT/SERVER
Multiple operating systems
One or more programming languages
Local and remote databases
Networking/communication issues
Inter-program communications
Message routing

What is a Middleware (1)

- Software which allows an application to ***interoperate*** with other software
- No need for programmer/user to understand internal processing
- Accomplished via ***Application Program Interface*** (API)

The "glue" that holds client/server applications together

What is a Middleware (2)



Type of Middleware

- RPC – Remote Procedure Calls (RPC)
  - client makes calls to procedures running on remote computers
  - synchronous and asynchronous
- Message-Oriented Middleware (MOM)
  - asynchronous calls between the client via message queues
- Publish/Subscribe
  - push technology → server sends information to client when available
- Object Request Broker (ORB)
  - Object-oriented management of communications between clients and servers
- SQL-oriented Data Access
  - Middleware between applications and database servers

Database Middleware

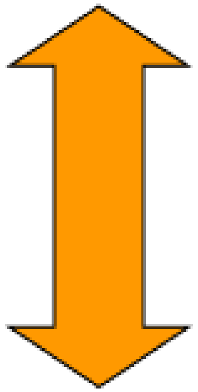
- ODBC – Open Database Connectivity
  - Most DB vendors support this
- OLE-DB
  - Microsoft enhancement of ODBC
- JDBC – Java Database Connectivity
  - Special Java classes that allow Java applications/applets to connect to databases

Client/Server Architectures (1)

File Server Architecture

Client does extensive processing

Database Server Architecture



Three-tier Architecture

Client does little processing

Client/Server Architectures (2)

<u>File Server Architecture</u>

- All processing is done at the PC that requested the data
- Entire files are transferred from the server to the client for processing.

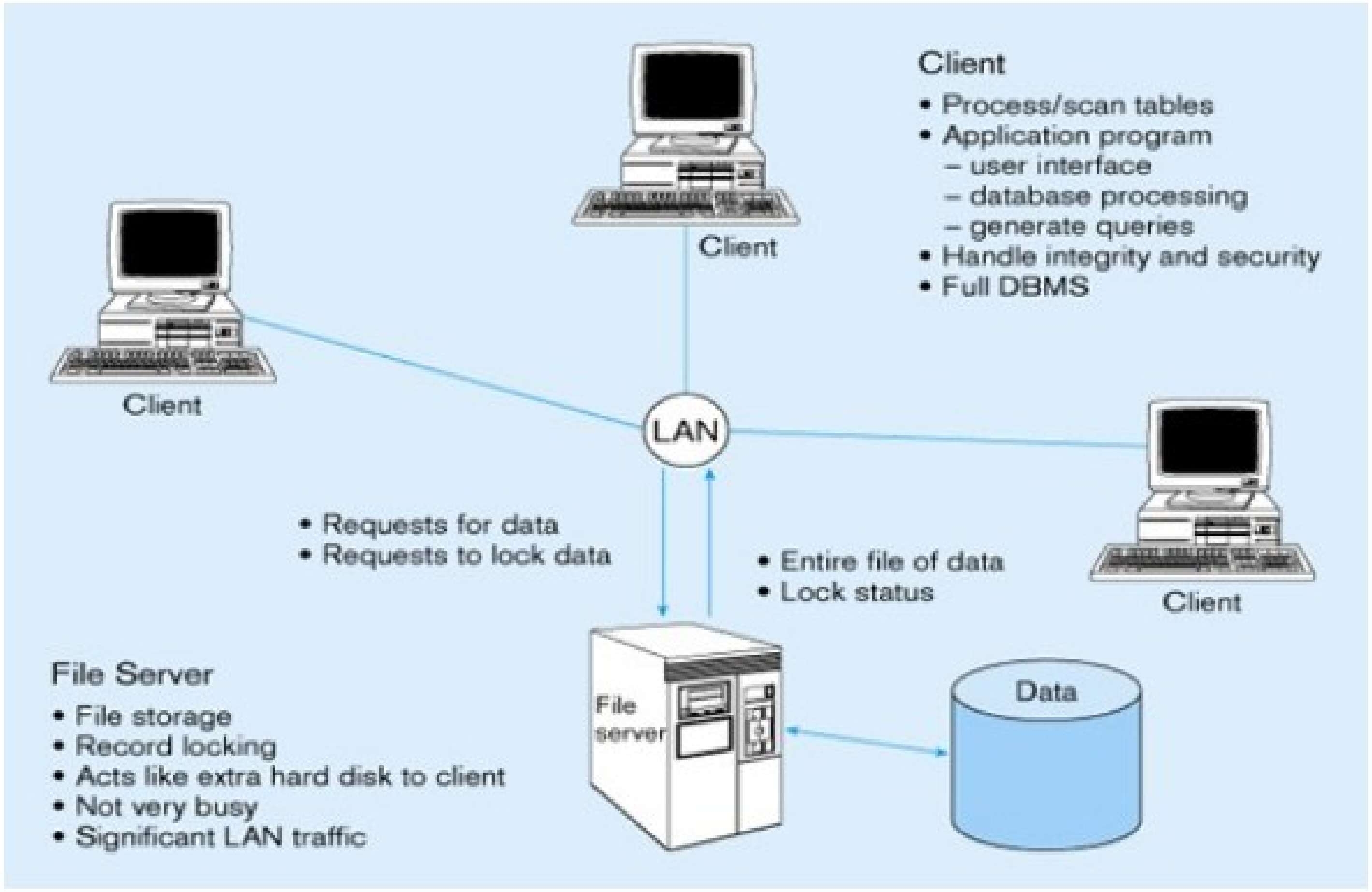
FAT CLIENT

Problems:

- Huge amount of data transfer on the network
- Each client must contain full DBMS
  - Heavy resource demand on clients
  - Client DBMSs must recognize shared locks, integrity checks, etc.

Client/Server Architectures (3)

File Server Architecture



Client/Server Architectures (4)

Database Server Architecture

This is a 2-tiered Approach

Thinner clients

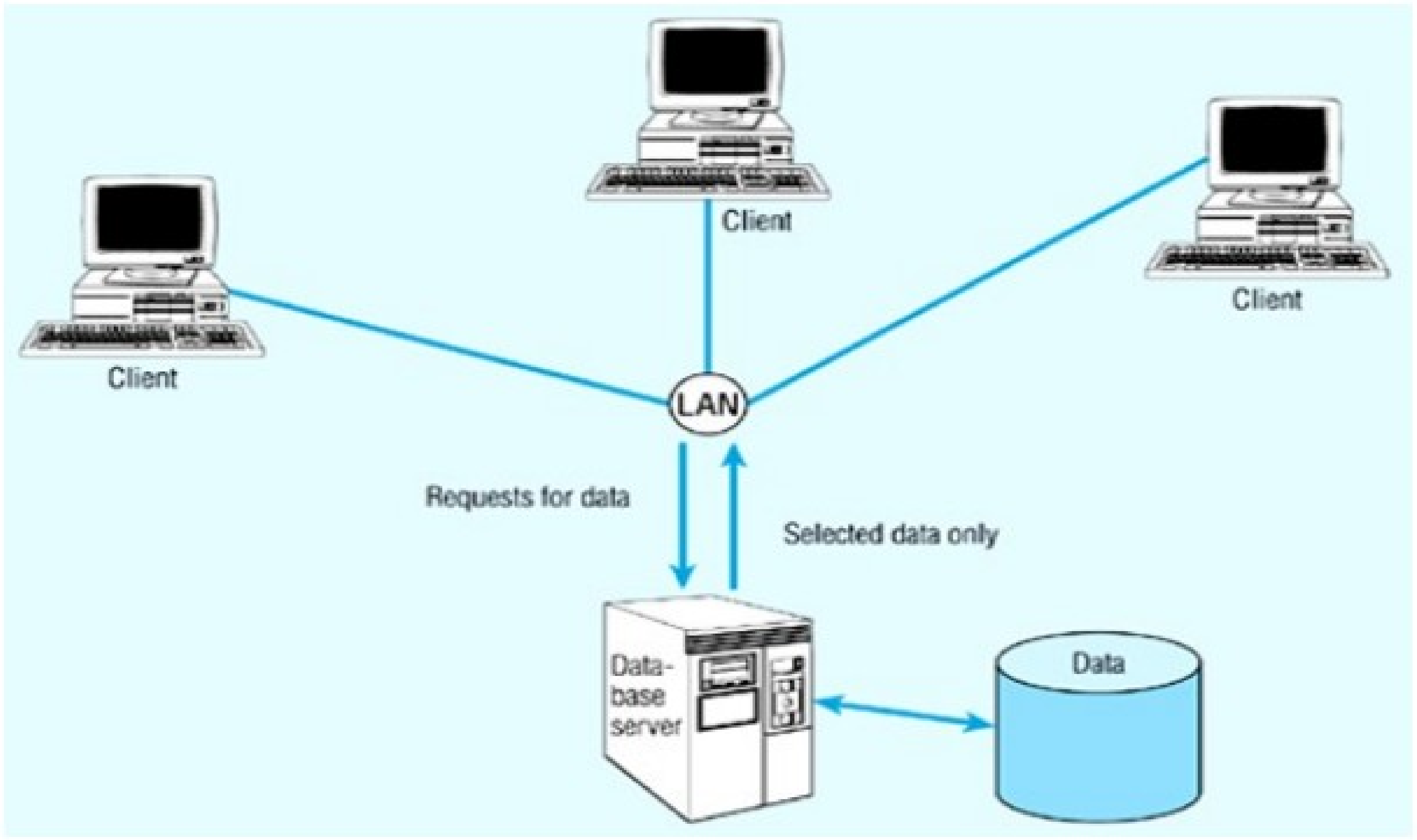
- Client is responsible for
  - I/O processing logic
  - Some business rules logic
- Server performs all data storage and access processing **{DBMS is only on server}**

Advantages

- Clients do not have to be as powerful
- Greatly reduces data traffic on the network
- Improved data integrity since it is all processed centrally
- **Stored procedures** → some business rules done on server

Client/Server Architectures (5)

Database Server Architecture



Client/Server Architectures (6)

Three-Tier Architecture

Three layers:

Client	**GUI interface (I/O processing)**	***Browser***
Application server	**Business rules**	***Web Server***
Database server	**Data storage**	***DBMS***

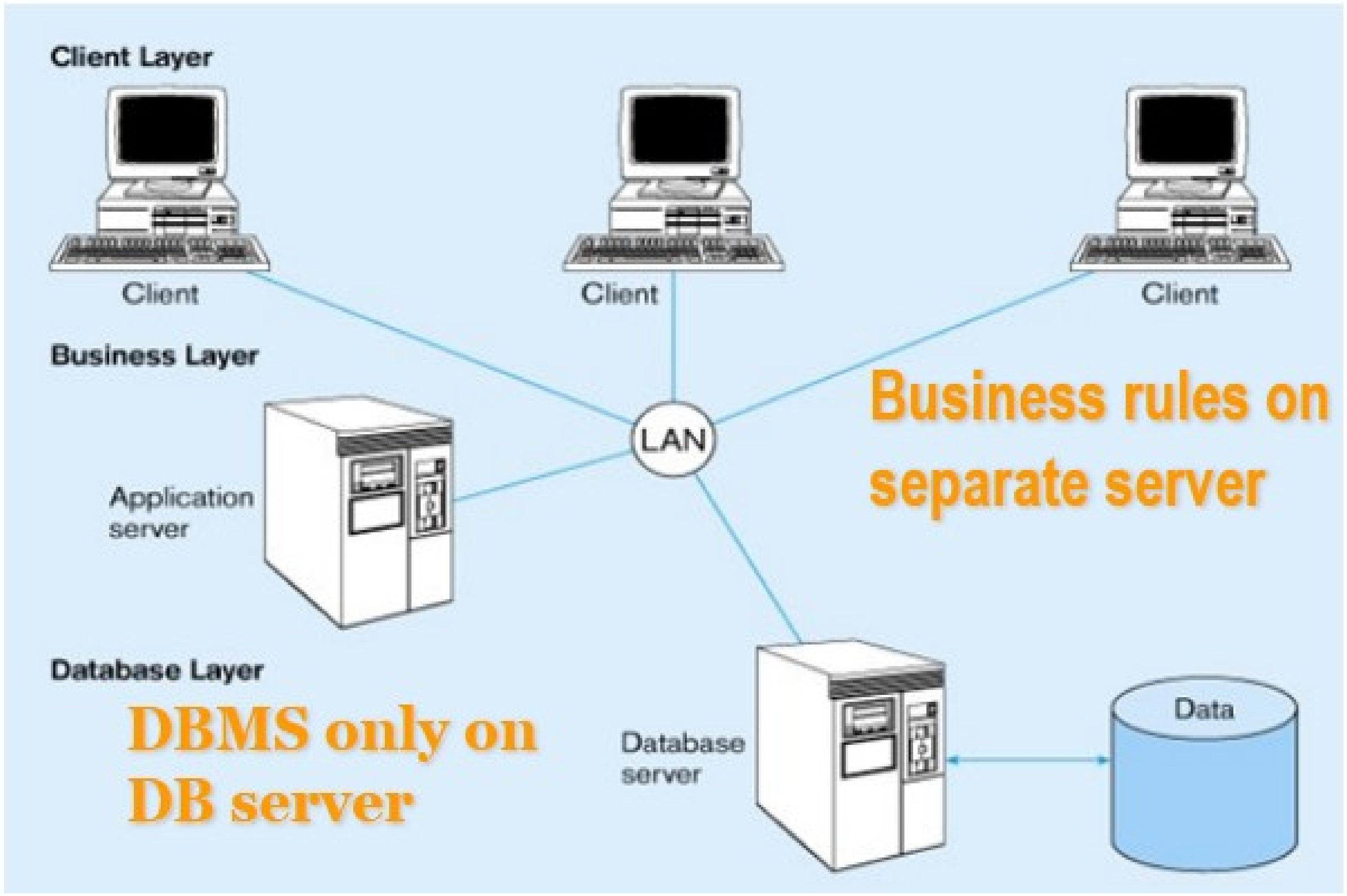
Thinnest clients

Thin Client

- PC is just for user interface and a little application processing.
- Limited or no data storage (sometimes no hard drive)

Client/Server Architectures (7)

Three-Tier Architecture



The Anatomy of a Server Program

- A **server program** is a software component that provides services and resources to client programs.
- **Listens for requests**: It constantly runs, waiting for client requests over a network.
- **Processes requests**: It receives a request, performs the necessary operations (e.g., querying a database, running business logic), and generates a response.
- **Manages resources**: It controls access to shared resources like databases, files, printers, and other hardware or software components.
- **High concurrency**: Server programs are designed to handle many client requests simultaneously, often using multitasking or multithreading within the operating system.

Anatomy of a Server Program

- Initialization
- Listening for requests
- Request handling
- Business logic
- Transaction management
- Response generation

Operating System Services

- Basic Services
- Extended Services

Operating System: Basic and Extended Services for Server Applications

- Server applications require specific services from their operating systems (OS) to function efficiently.
- **Basic Services**: These are standard features of the OS:
  - **Multitasking/multithreading**: Essential for handling numerous concurrent client requests.
  - **Memory management**: Efficient allocation and management of memory for performance and stability.
  - **Inter-process communication (IPC)**: Mechanisms for different parts of the server program or different programs to communicate.
  - **Task prioritization**: The ability to prioritize important client requests or background tasks.

- **Extended Services**: These are modular, add-on software components layered on top of the base OS:
  - **Database management systems (DBMS)**: Software for managing data storage and retrieval.
  - **Transaction processing monitors (TPMs)**: Ensure the integrity of transactions across a network.
  - **Middleware**: Software that facilitates communication between diverse clients and servers.
  - **Security services**: Advanced authentication and authorization mechanisms like directory services (e.g., Active Directory).

Basic OS Services

- Process management
- Memory management
- File systems
- Networking
- Device management

Extended OS Services

- Multithreading
- Security
- Transaction support
- Fault tolerance
- Load balancing

Server Scalability

- Vertical scalability (Scale-up)
- Horizontal scalability (Scale-out)
- Load balancing

Server Scalability

- **Scalability** is the ability of a system to handle a growing amount of work, such as an increased number of clients or requests.
- **Vertical Scalability**: Increasing the capacity of a single server by adding more resources (e.g., more CPU power, more RAM, faster storage).
- **Horizontal Scalability**: Adding more servers to the system and distributing the workload among them, often using techniques like load balancing.
- **Partitioning/Replication**: Servers can be logically partitioned by function or data, or replicated for redundancy and performance.

Client Anatomy

- A **client program** is the consumer of services, initiating requests to a server.
- **User interface**: The client is typically responsible for the presentation layer, providing the interface with which the end user interacts.
- **Initiates communication**: The client starts the interaction by sending a request to the server.
- **Receives and presents data**: It receives the server's response and presents the information in a human-readable format.
- **Can be diverse**: Clients can run on various devices and operating systems (desktops, laptops, mobile phones, IoT devices, web browsers).

Client Anatomy

- User Interface
- Presentation logic
- Communication module
- Local processing
- Session handling

Transactions in Client/Server

- Atomicity
- Consistency
- Isolation
- Durability (ACID)

Client/Server Hybrids

- In many real-world scenarios, a single machine or application can function as both a client and a server, creating a hybrid model.
- **Peer-to-peer (P2P) systems**: A classic example where each node can act as both a client (requesting files) and a server (providing files).
- **Multi-tier architecture**: In a 3-tier system, the application server acts as a client to the database server, while acting as a server to the presentation layer client.
- **Local communication**: A client and server can run on the same physical computer and communicate with each other.

Summary

- Clients handle interaction
- Servers handle shared resources
- OS ensures performance and scalability