Distributed Systems

Soma Ghosh
Email: gsn.comp@coeptech.ac.in

Definition

- A distributed operating systems is one that looks to its users like an ordinary centralized operating system but runs on multiple, independent CPUs.
- The use of multiple processors should be invisible to the user.

Introduction

- Reasons for using Distributed Systems:
- To connect several computers so that applications can communicate
- **Tightly coupled systems**: memory shared by all processors referred as *parallel processing systems*.
- **Loosely coupled systems**: processor do not share memory has its own local memory referred as *distributed computing systems*.

Examples of Distributed Systems

- Telephone network and Cellular networks
- Computer networks like internet
- Wireless sensor networks

Applications of Distributed Systems

- World wide web and peer-to-peer networks
- Multiplayer games and
- Distributed databases and Distributed database management systems

Distributed Architecture

- In distributed architecture, components are presented on different platforms and several components can cooperate with one another over a communication network in order to achieve a specific objective or goal.
- In this architecture, information processing is not confined to a single machine rather it is distributed over several independent computers.

Issues in designing a DOS

- Transparency
- Reliability
- Flexibility
- Performance
- Scalability
- Heterogeneity
- Security

Transparency

- One of the main goals of distributed operating system is to make the existence of multiple computers invisible (transparent) and provide a single system image to its users.

Transparency

- **Access transparency**:- users need not know whether a resource is remote or local.
- **Location transparency** :-
- **(i) Name transparency** :- Name of resource should not reveal any hint of the physical location of the resource. Name of resources should be unique irrespective of the location of the resource e.g.: a file
- **(ii) User mobility**:- user should be able to access any resource irrespective if its local or remote.

- **Replication transparency** :- DOS have the provision to create replicas of files and other resources on different nodes of the DS.

- **Failure transparency :-** masking from the user partial failures in the system.

Transparency cntd…

- **Migration transparency :-** for better performance, reliability and security reasons an object or resource like a file is moved i.e., migrated from one node to other automatically by the system in a user transparent manner.

Transparency cntd….

- **Concurrency transparency :-** a user should feel that he or she is sole user of the resource DOS must have following properties for this : event-ordering, mutual-exclusion, no-starvation and no deadlock.
- **Performance transparency :-** system should automatically reconfigure to increase the performance e.g.:- load balancing
- **Scaling transparency :-** Allow the system to expand without disrupting activities of a user. DOS must use scalable algorithms.

Reliability

Reliability

- **Fault Avoidance:-** System should be designed in such a way that occurrence of faults is minimized.
- **Fault tolerance:-** system should continue functioning in event of partial failure.
- (i) Redundancy techniques: to avoid single point of failure, critical h/w & s/w is replicated so that if one of them fails the others can be used to continue.
- (ii) Distributed control: many protocols in DOS should employ a distributed control mechanism to avoid single point of failure.

- **Fault detection & recovery**
- **(i) atomic transactions**
- **(ii) stateless servers**
- **(iii) acknowledgements & timeout-based retransmission of messages**

Flexibility

- Design of DOS should be flexible due to following reasons:-

1. **Ease of modification**: system needs to be modified to fix a bug or to accommodate user requirements.
2. **Ease of enhancement**: new functionalities has to be added from time to time to increase performance and for ease of usage.

Performance

1. **Batch if possible**: transfer of data in chunks.
2. **Cache whenever possible**
3. **Minimize copying of data**
4. **Minimize network traffic**
5. **Take advantage of fine grain parallelism for multiprocessing**

Scalability

1. **Capability of a system to adapt to increase service load. DOS should be designed to easily cope with growth with nodes/users.**
2. **Avoid centralized entities :** central file server or single database for entire system should be avoided.
3. **Avoid centralized algorithms :** algorithms that collect data from all nodes and process information on a single node should be avoided.
4. **Perform most operations on client workstations:** performing operations on server should be avoided as server cycles are precious.

Security

1. It should be possible for sender of a message to confirm the message is received by the intended receiver
2. It should be possible for receiver of a message to confirm the message is sent by the genuine receiver
3. It should be possible for both sender & receiver of a message to confirm the contents of the message were not changed

Pros of Distributed Systems:

Advantages:

- **Resource sharing** – Sharing of hardware and software resources.
- **Openness** – Flexibility of using hardware and software of different vendors.
- **Concurrency** – Concurrent processing to enhance performance.
- **Scalability** – Increased throughput by adding new resources.
- **Fault tolerance** – The ability to continue in operation after a fault has occurred.

Cons of Distributed Systems:

Disadvantages

- **Complexity** – They are more complex than centralized systems.
- **Security** – More susceptible to external attack.
- **Manageability** – More effort required for system management.
- **Unpredictability** – Unpredictable responses depending on the system organization and network load.
- **No global clocks**
- **No shared memory**

Centralized System vs. Distributed System

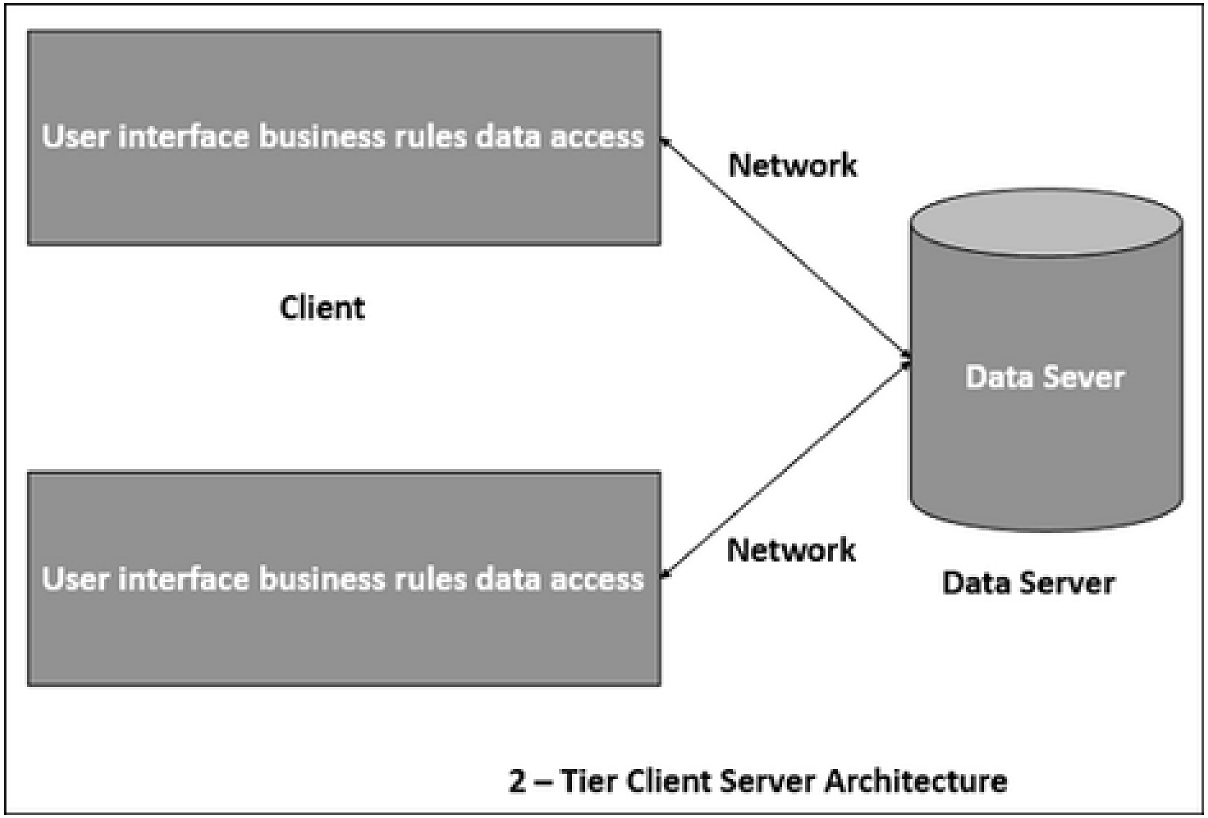
Criteria	**Centralized system**	**Distributed System**
Cost	Low	High
Availability of resources	Low	High
Complexity of system	Low	High
Consistency	Simple	High
Scalability	Poor	Good
Technology	Homogeneous	Heterogeneous
Security	High	Low

System Level Architecture

The two major system level architectures that we use today are

- **Client-server**
- **Peer-to-peer** (P2P)

Client-Server Architecture


User interface business rules data access
Network
Client
Data Sever
User interface business rules data access
Network
Data Server
2 – Tier Client Server Architecture

Client-Server Architecture

- The client server architecture has two major components:-
- The **client:** the Client is where the user can access the services and resources given by the Server (Remote Server)
- The **server**: The Server is where all the processing, computing and data handling is happening takes place.

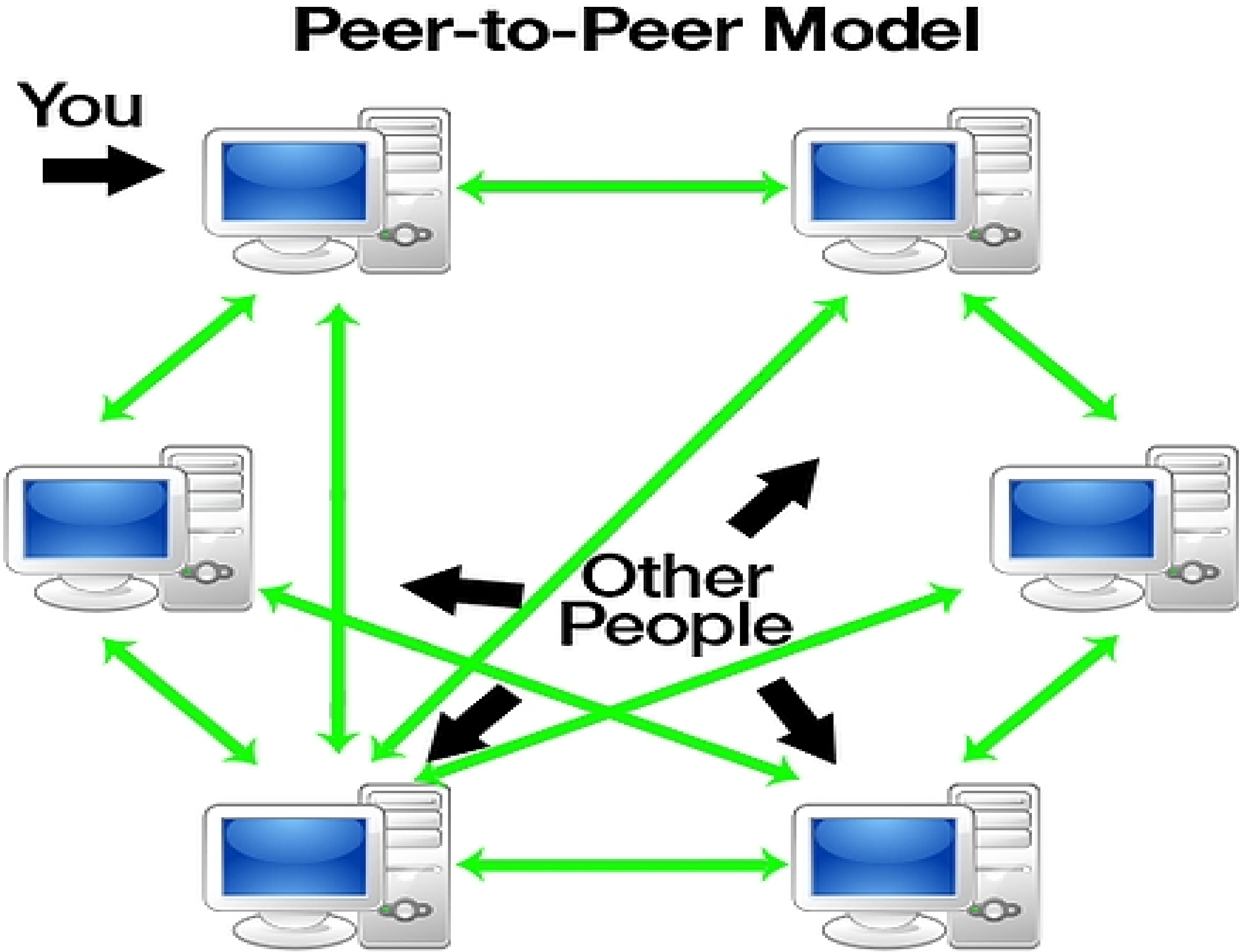
Client-Server Architecture

Advantages:

- Easier to Build and Maintain
- Better Security
- Stable

Disadvantages:

- Single point of failure
- Less scalable


Peer-to-Peer Model
You
Other People

Peer to Peer (P2P)

- The general idea behind peer to peer is where there is no central control in a distributed system.
- Each node can either be a client or a server at a given time. If the node is requesting something, it can be known as a **client**, and if some node is providing something, it can be known as a **server**. In general, each **node** is referred to as a **Peer**.

Peer to Peer (P2P)

- **Centralized Lookup Server :** The new node has to register with the centralized look up server and mention the services it will be providing, on the network.
- **Decentralized System** - A node desiring for specific services must, broadcast and ask every other node in the network, so that whoever is providing the service will respond.

Client Server and Peer to Peer Architectures

Criteria	Client-server	P2P
Basic	There is a specific server and clients connected to it	Clients and server are not distinguished
Service	The client requests for service and server responds with the service	Each node can request for services and can also provide services
Focus	Sharing of information	Connectivity
Data	Data is stored in a centralized server	Each peer has its own data
Server	A server can get bottlenecked	Many servers so a server is not overloaded
Expense	expensive	Less expensive