CLOUD COMPUTING

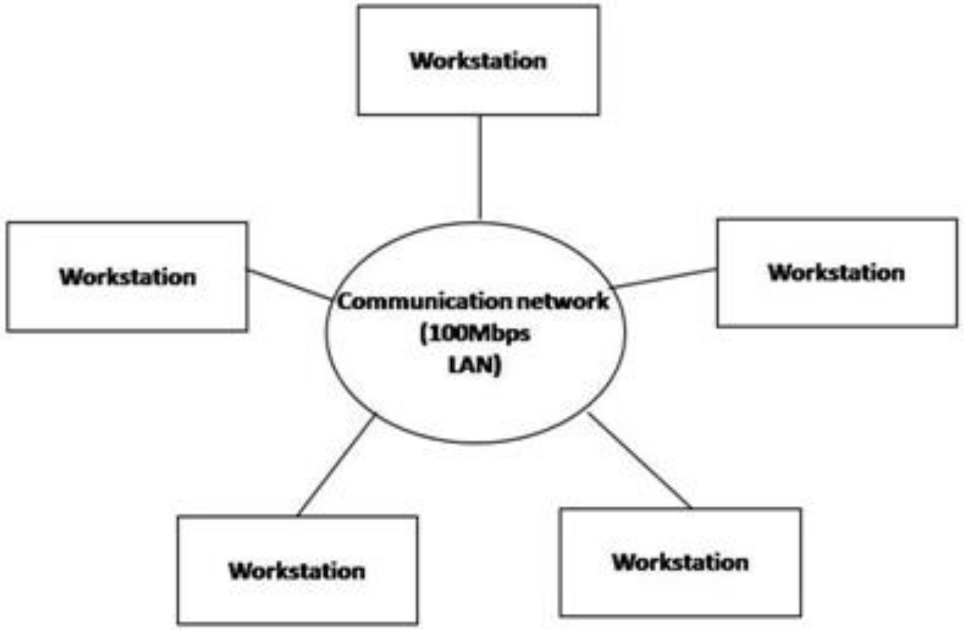
Master-slave v/s p2p models

Prof.S.Suganthi

Department of Computer Science and Engineering

Distributed Systems

A **distributed system**, also known as **distributed computing**, is a **system** with multiple components located on different machines that communicate and coordinate actions in order to appear as a single coherent **system** to the end-user



Distributed Systems Architecture

Two main architectures:

- Master-Slave architecture
  - Roles of entities are *asymmetric*
- Peer-to-Peer architecture
  - Roles of entities are *symmetric*

Master-Slave Architecture

- A master-slave architecture can be characterized as follows:

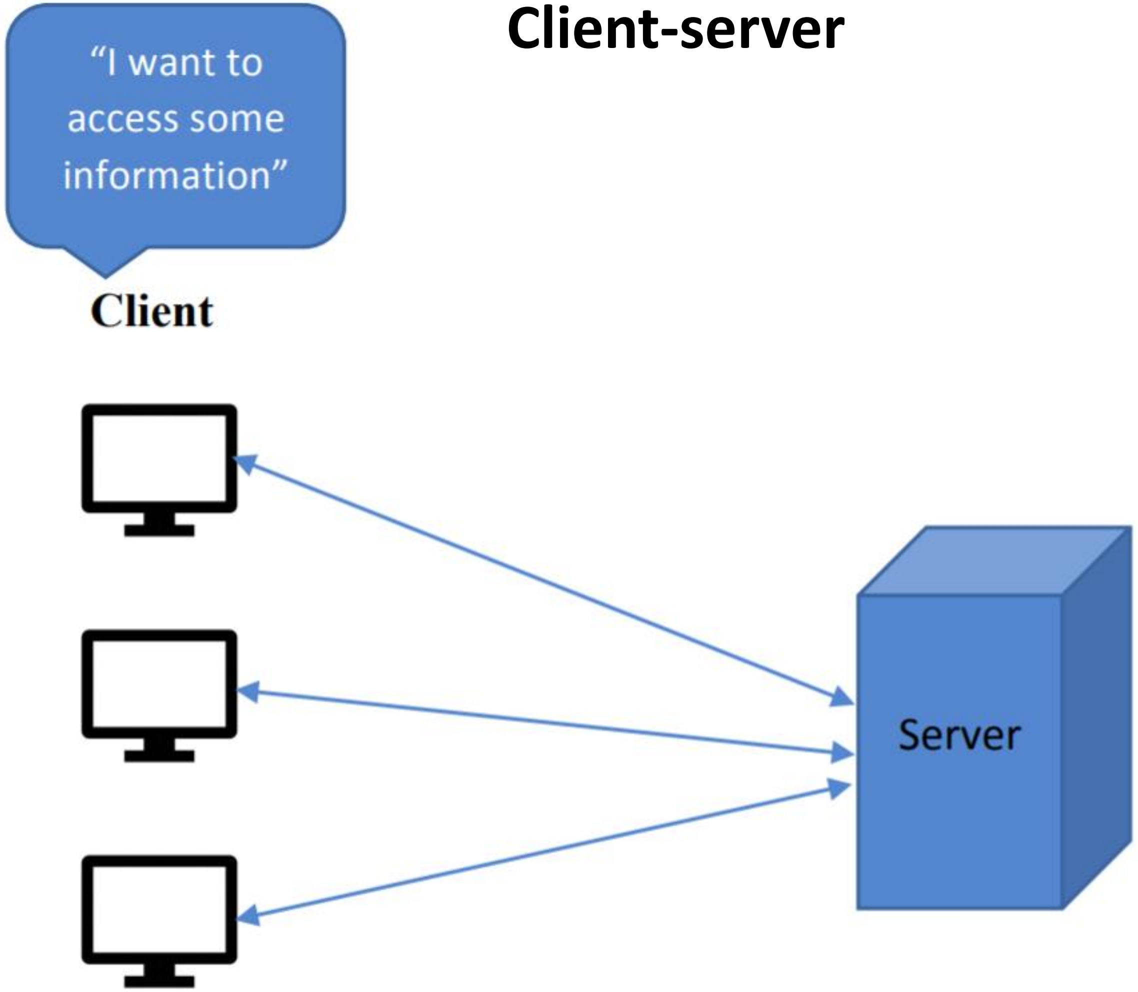
1) Nodes are *unequal* (there is a hierarchy)
   - Vulnerable to *Single-Point-of-Failure* (SPOF)

2) The master acts as a *central coordinator*
   - Decision making becomes easy

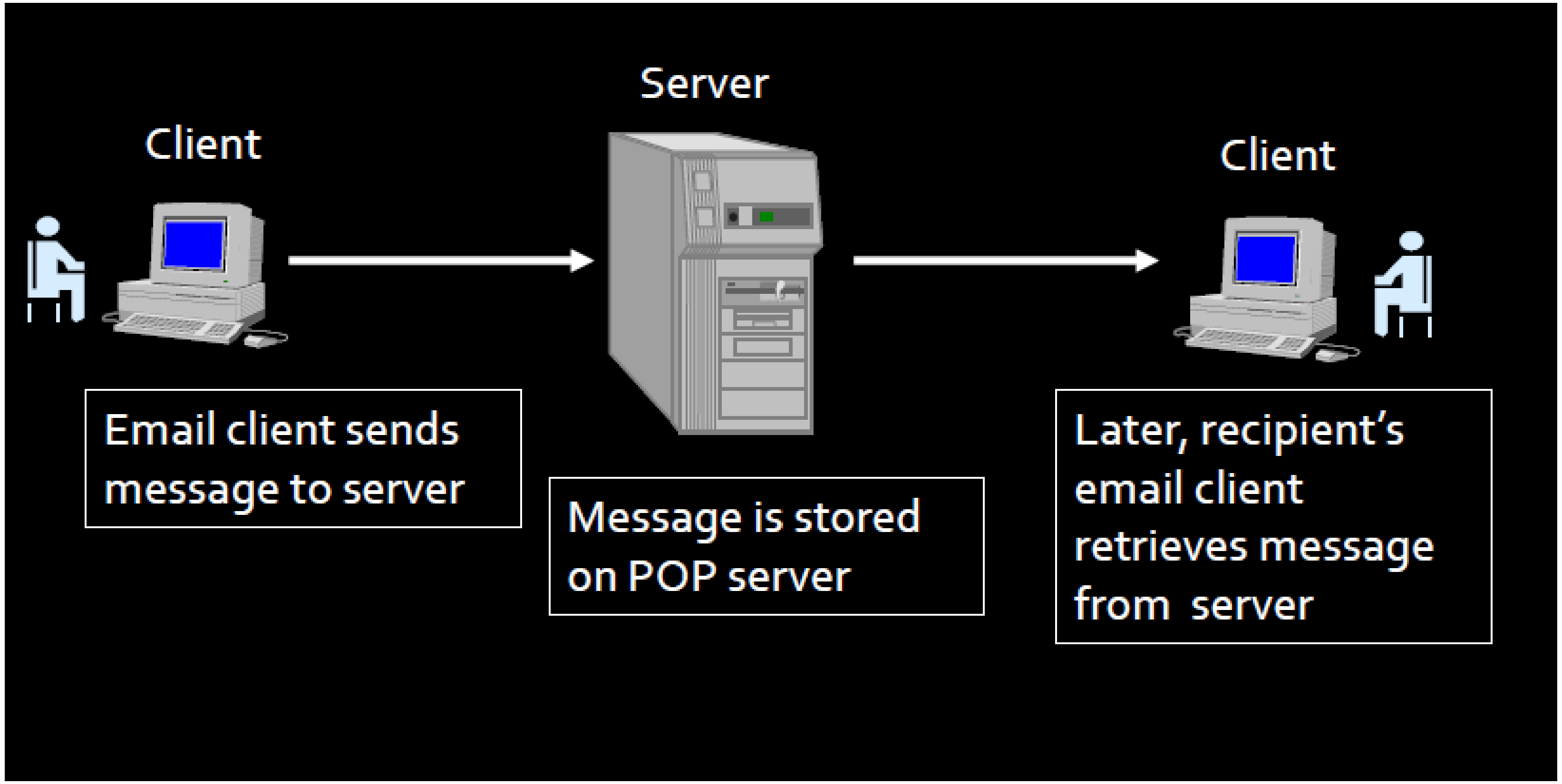
3) The underlying system *cannot scale out* indefinitely
   - The master can render a *performance bottleneck* as the number of workers is increased

Master-Slave Architecture

Client-server



Examples - Email Application

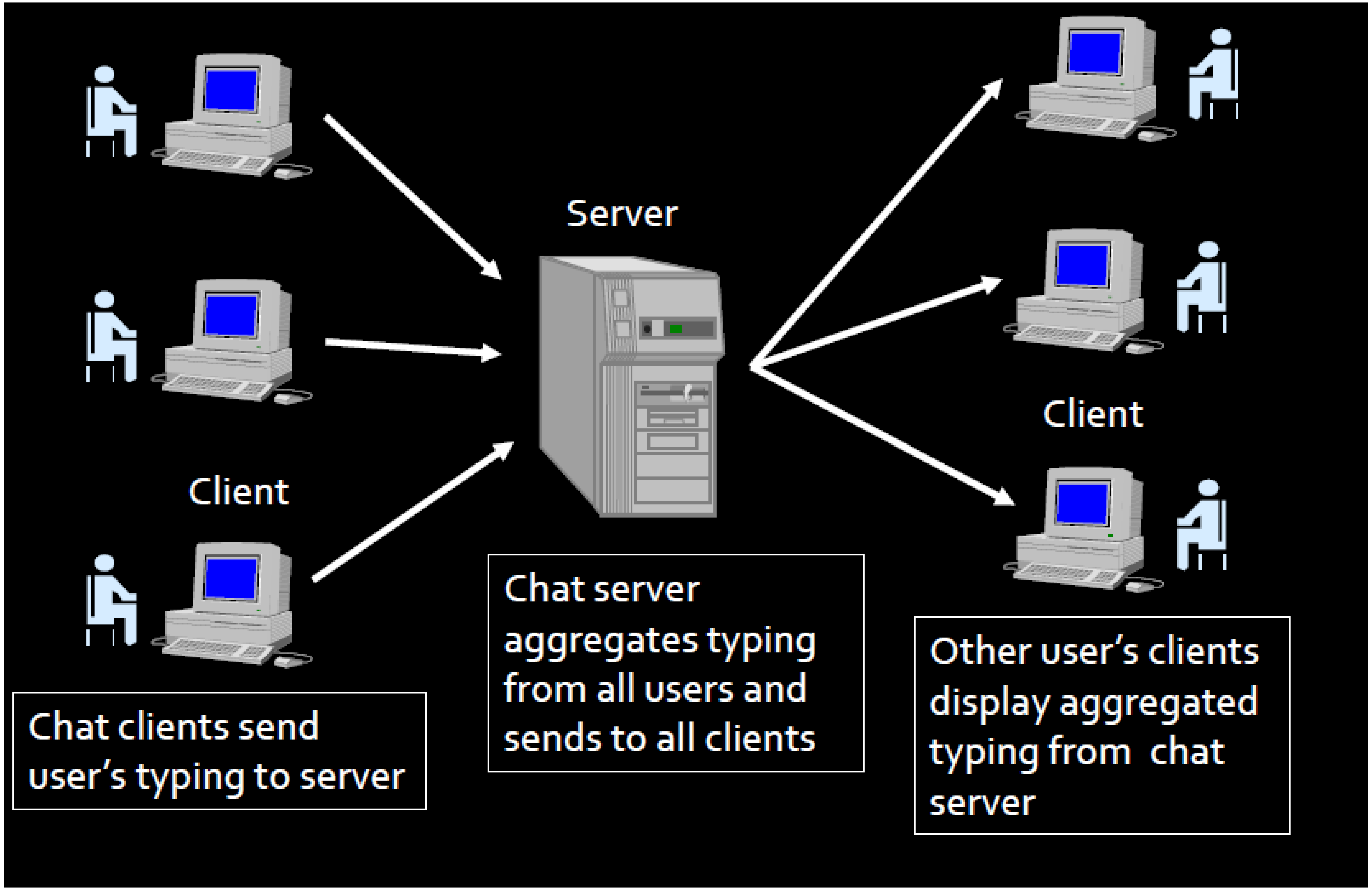


Email client sends message to server

Message is stored on POP server

Later, recipient's email client retrieves message from server

Examples - Chat Application

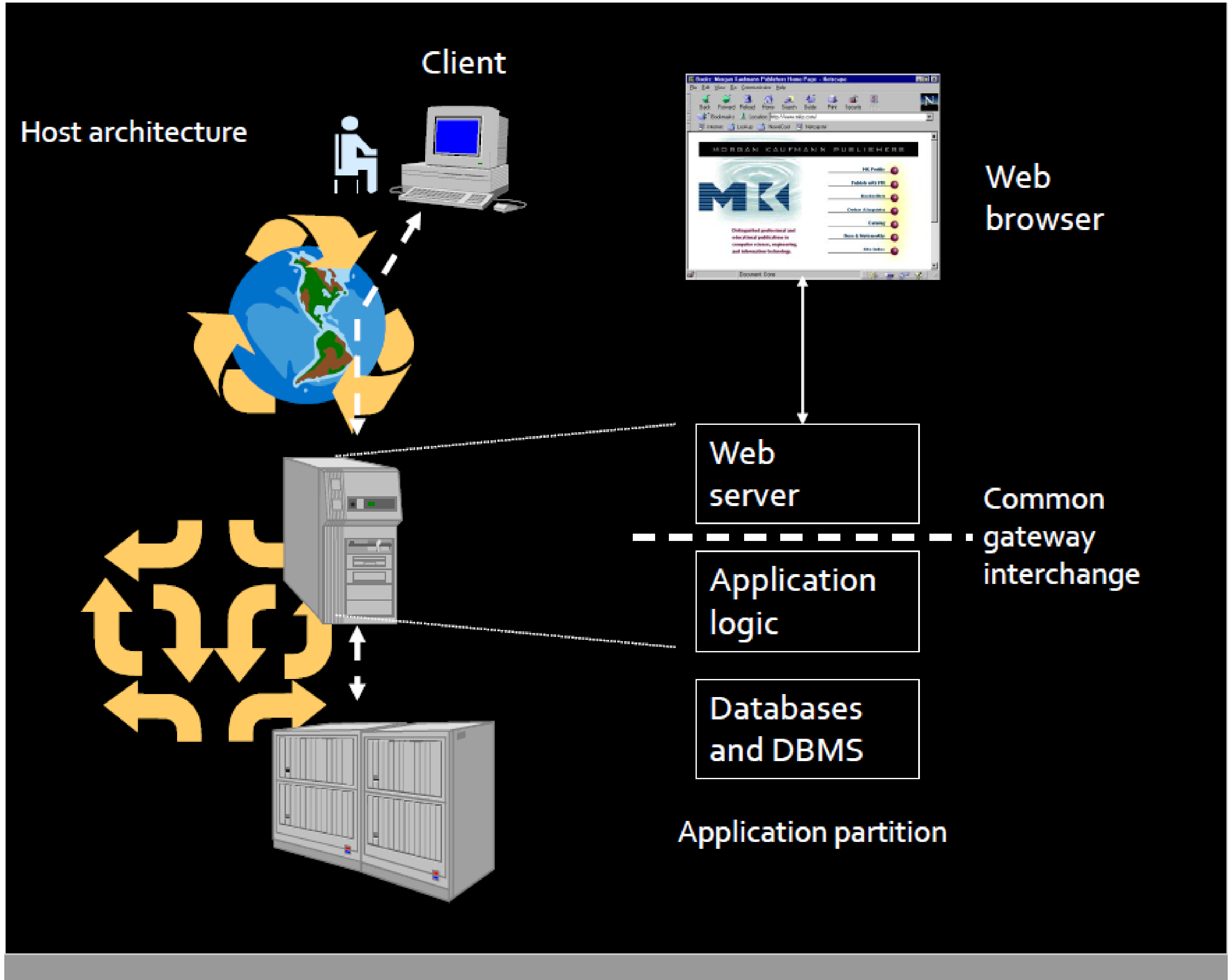


Chat clients send user's typing to server

Chat server aggregates typing from all users and sends to all clients

Other user's clients display aggregated typing from chat server

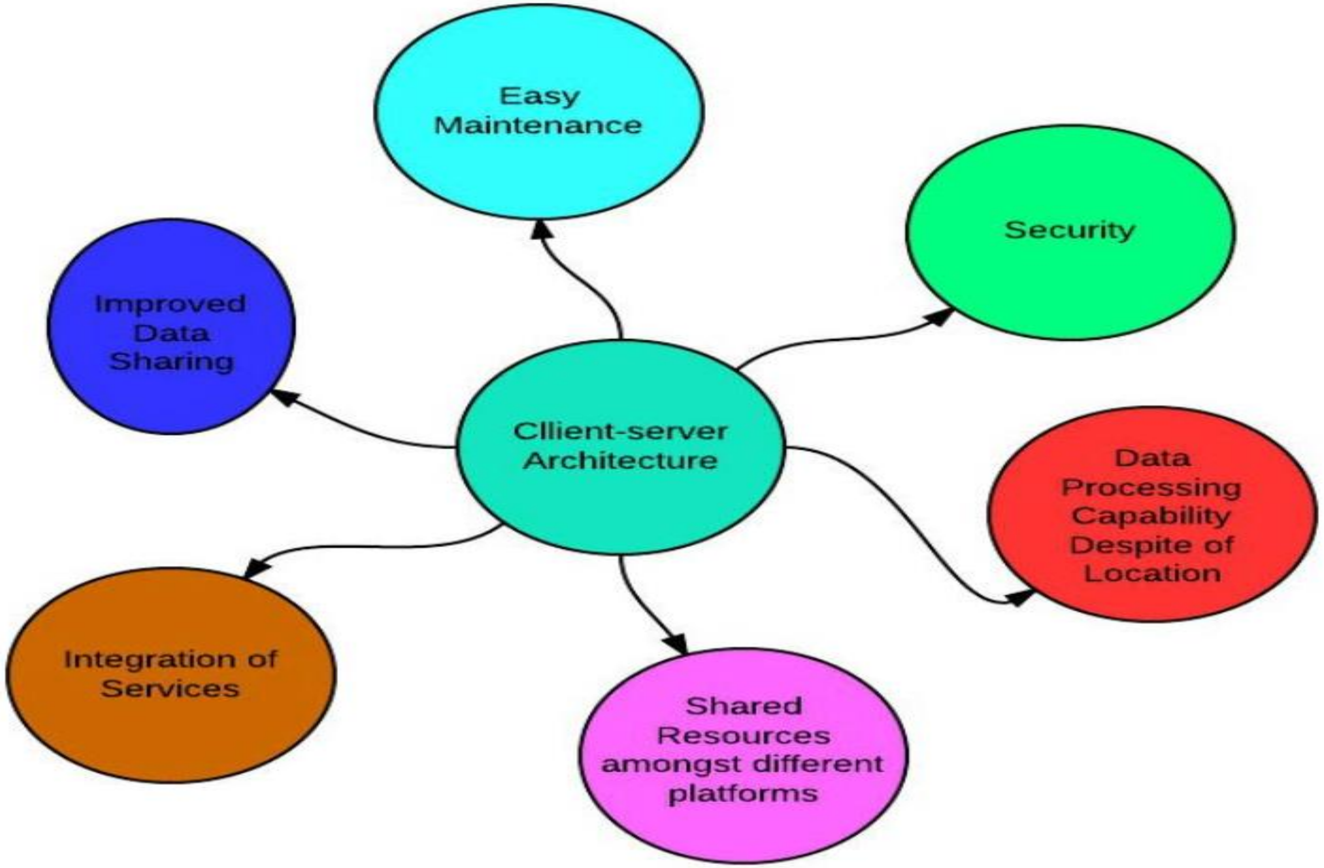
3 tier Client-Server Architecture



Information about the client is stored in a middle tier rather than on the client to simplify application deployment. This architecture model is most common for web applications.

Client-Server Architecture

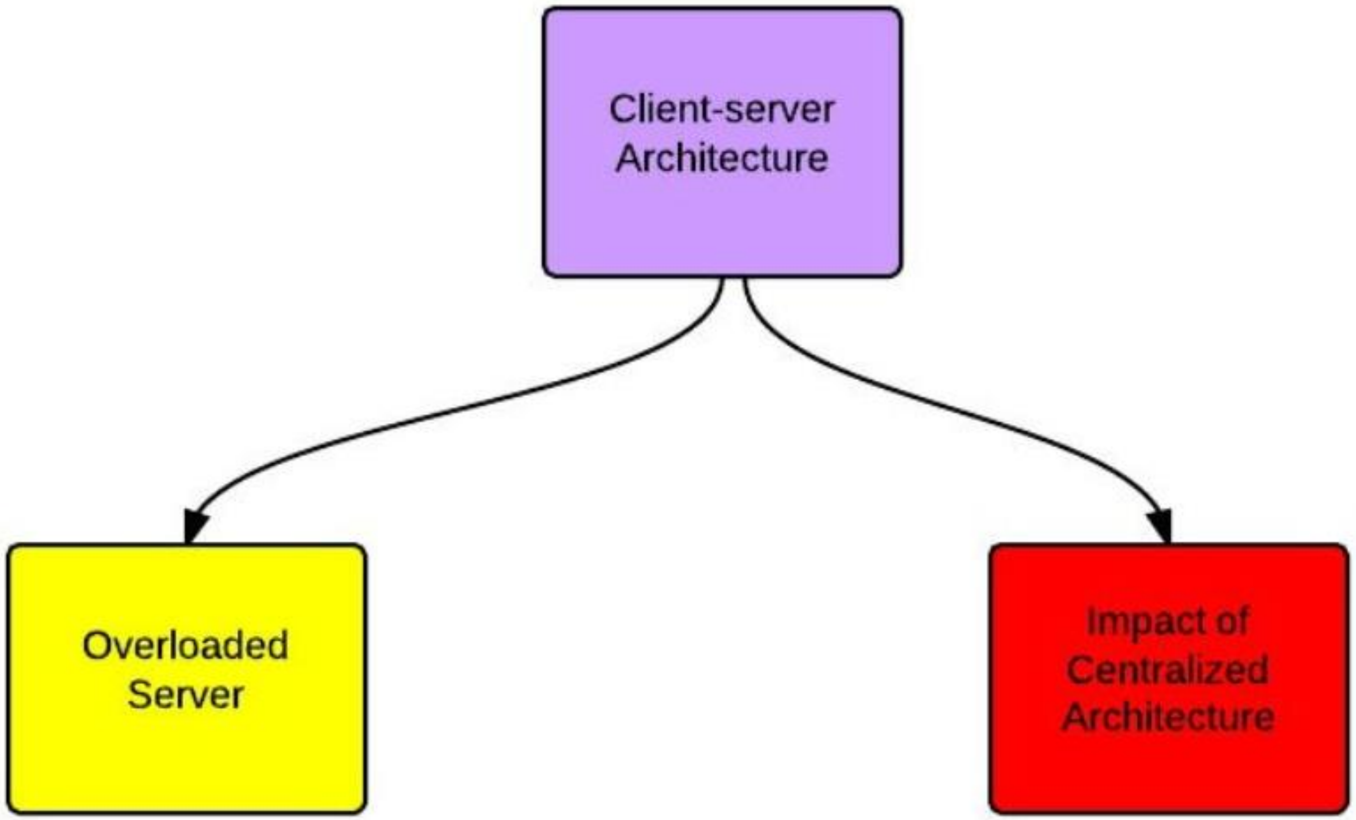
- Advantages


Easy Maintenance
Security
Improved Data Sharing
Cllient-server Architecture
Data Processing Capability Despite of Location
Integration of Services
Shared Resources amongst different platforms

Source: https://sites.google.com/site/clientserverarchitecture/advantages-of-client-server-architecture

Client-Server Architecture

- Dis-advantages


Client-server Architecture
Overloaded Server
Impact of Centralized Architecture

Source: https://sites.google.com/site/clientserverarchitecture/advantages-of-client-server-architecture

What is Peer-to-Peer?

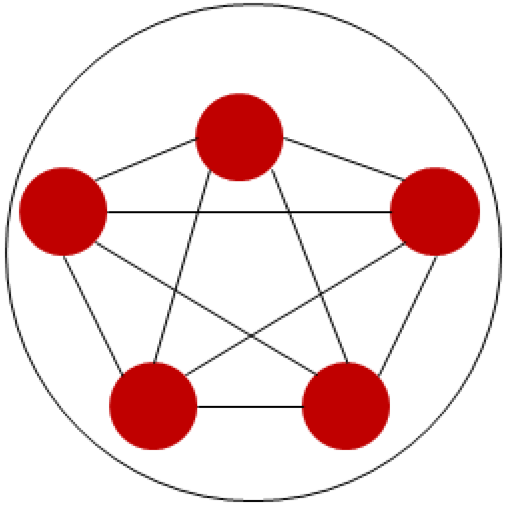
- A model of communication where every node in the network acts alike.
- As opposed to the Client-Server model, where one node provides services and other nodes use the services.

P2P Architecture

- A peer-to-peer (P2P) architecture can be characterized as follows:

1) All nodes are equal (no hierarchy)
   - No Single-Point-of-Failure (SPOF)

2) A central coordinator is not needed
   - But, decision making becomes harder



3) The underlying system can scale out indefinitely
   - In principle, no performance bottleneck

P2P Architecture

4) Peers can interact directly, forming groups and sharing contents (or offering services to each other)
   - At least one peer should share the data, and this peer should be accessible
   - Popular data will be highly available (it will be shared by many)
   - Unpopular data might eventually disappear and become unavailable (as more users/peers stop sharing them)

5) Peers can form a virtual *overlay network* on top of a physical network topology
   - *Logical paths* do not usually match *physical paths* (i.e., higher latency)
   - Each peer plays a role in routing traffic through the overlay network

Advantages of P2P Computing

- No central point of failure
  - E.g., the Internet and the Web do not have a central point of failure.
  - Most internet and web services use the client-server model (e.g. HTTP), so a specific service does have a central point of failure.
- Scalability
  - Since every peer is alike, it is possible to add more peers to the system and scale to larger networks.

Disadvantages of P2P Computing

- Decentralized coordination
  - How to keep global state consistent?
  - Need for distributed coherency protocols.
- All nodes are not created equal.
  - Computing power, bandwidth have an impact on overall performance.
- Programmability
  - As a corollary of decentralized coordination.

P2P Computing Applications

- File sharing
- Process sharing
- Collaborative environments

P2P File Sharing Applications

- Improves data availability
- Replication to compensate for failures.
- E.g., Napster, Gnutella, Freenet, KaZaA (FastTrack), your DFS project.

P2P Process Sharing Applications

- For large-scale computations
- Data analysis, data mining, scientific computing
- E.g., SETI@Home, Folding@Home, distributed.net, World-Wide Computer

P2P Colloborative Applications

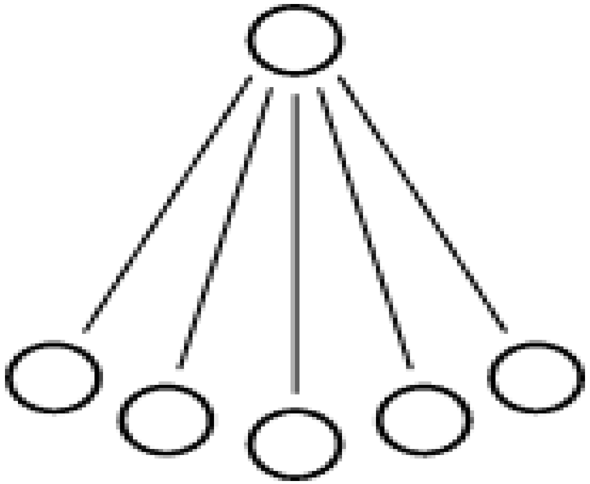
- For remote real-time human collaboration.
- Instant messaging, virtual meetings, shared whiteboards, teleconferencing, tele- presence.
- E.g., talk, IRC, ICQ, AOL Messenger, Yahoo! Messenger, Jabber, MS Netmeeting, NCSA Habanero, Games

P2P Topologies

- Centralized
- Ring
- Hierarchical
- Decentralized
- Hybrid

P2P Topologies: Centralized Topology

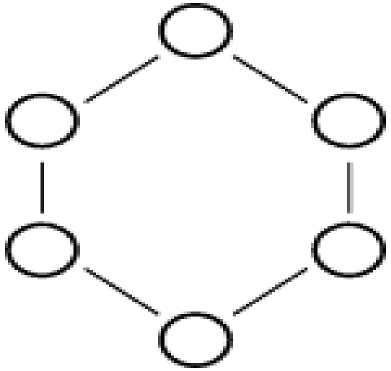
- Centralized



Manageable	✓	System is all in one place
Coherent	✓	All information is in one place
Extensible	X	No one can add on to system
Fault Tolerant	X	Single point of failure
Secure	✓	Simply secure one host
Lawsuit-proof	X	Easy to shut down
Scalable	?	One machine. But in practice?

P2P Topologies: Ring Topology

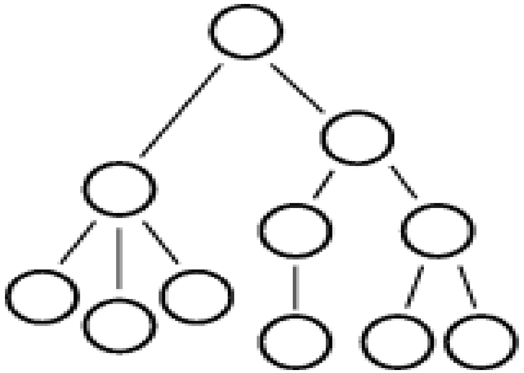
- Ring



Manageable	✓	Simple rules for relationships
Coherent	✓	Easy logic for state
Extensible	X	Only ring owner can add
Fault Tolerant	✓	Fail-over to next host
Secure	✓	As long as ring has one owner
Lawsuit-proof	X	Shut down owner
Scalable	✓	Just add more hosts

P2P Topologies: Hierarchical Topology

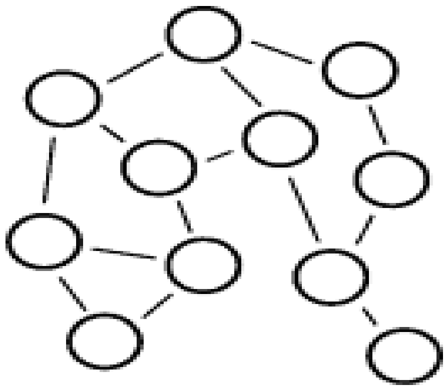
- Hierarchical



Manageable	½	Chain of authority
Coherent	½	Cache consistency
Extensible	½	Add more leaves, rebalance
Fault Tolerant	½	Root is vulnerable
Secure	X	Too easy to spoof links
Lawsuit-proof	X	Just shut down the root
Scalable	✓	Hugely scalable – DNS

P2P Topologies: Decentralized Topology

- Decentralized



Manageable	X	Very difficult, many owners
Coherent	X	Difficult, unreliable peers
Extensible	✓	Anyone can join in!
Fault Tolerant	✓	Redundancy
Secure	X	Difficult, open research
Lawsuit-proof	✓	No one to sue
Scalable	?	Theory – yes : Practice – no

THANK YOU

Ms. Suganthi S
Mail-id: suganthis@pes.edu

Department of Computer Science and Engineering