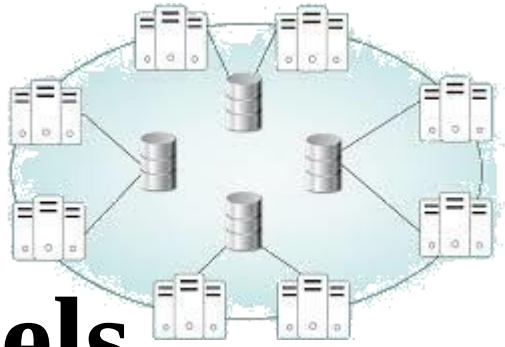
**Distributed and Parallel Computer Systems**

**CSC 423**



**Lecture 3**

# Challenges & System Models

## INSTRUCTOR

DR / AYMAN SOLIMAN

# ➢ Contents

1) Challenges
2) System Models Introduction
3) Architectural Models
   - Software Layers
   - System Architectures
     - Client-Server
       - ✓ Clients and a Single Server, Multiple Servers, Proxy Servers with Caches, Peer Model

# ❑Challenges

- The challenges arising from the construction of distributed systems are:
  1) heterogeneity of its components.
  2) openness, which allows components to be added or replaced.
  3) Security.
  4) scalability - the ability to work well when the number of users increases.
  5) failure handling.
  6) concurrency of components and
  7) transparency.

# ❑1- Heterogeneity

➢varying software and hardware

- Networks.
- Computer Hardware.
- Operating Systems.
- Programming Languages.
- Implementations by different developers.
- need for standards (protocols, middleware)

# ❑2-Openness

- determines whether the system can be extended and re-implemented in various ways.
- independence of vendors
- publishable key interfaces
  - CORBA
- publishable communication mechanisms
  - Java RMI

# ❑3-Security

- Confidentiality (protect against disclosure)
  - medical records
- Integrity (protect against alteration and interference)
  - financial data
- Availability (Denial of services attacks).
- Need encryption and knowledge of identity.

# ❑4-Scalability

- will it remain effective with growth (resources\users)?

- need to control cost of resources, performance bottleneck,…
  - e.g., escalating growth of computers/web server ratio

# ❑5-Failure handling

- Ability to continue computation in the presence of failures.
  - detect/mask/tolerate failures
  - recovery from failures
    - redundancy

# ❑6-Concurrency

- Processes execute simultaneously and share resources.
  - synchronization
  - inter-process communication

# ❑7-Transparency

- Concealment of the separated nature of system from user/programmer.
  - Access transparency
  - Location transparency
  - Concurrency transparency
  - Replication transparency
  - Failure transparency
  - Scaling transparency
  - Mobility transparency

# ➢ Contents

1) Challenges
2) System Models Introduction
3) Architectural Models
   - Software Layers
   - System Architectures
     - Client-Server
       - ✓ Clients and a Single Server, Multiple Servers, Proxy Servers with Caches, Peer Model

# ❑ Difficulties and Threats to Distributed Systems

- Widely varying modes of use:
  - The component parts of systems are subject to wide variations in workload
- Wide ranged system environments:
  - A distributed system must accommodate heterogeneous hardware, operating systems and networks.
- Internal problems:
  - Non-synchronized clocks, conflicting data updates, many modes of hardware and software failure involving the individual components of a system.
- External threats:
  - Attacks on data integrity and secrecy, denial of service.

# ❑ Introduction

- An Architectural model of a distributed system is concerned with the placement of its parts and relationship between them.

- Examples:
  - Client-Server (CS) and peer process models.

- CS can be modified by:
  - The partitioning of data/replication at cooperative servers
  - The caching of data by proxy servers or clients
  - The use of mobile code and mobile agents
  - The requirements to add or remove mobile devices.

# ❑ Introduction

- Fundamental Models are concerned with a formal description of the properties that are common in all the architectural models.
  - There is no global time in a distributed system,
  - All communication between processes is achieved by means of messages.
  - Message communication over a computer network can be affected by delays, can surfer from a variety of failures and is vulnerable (يسهل مهاجمته ) to security attacks.

# ❑ Introduction

- Models addressing time synchronization, message delays, failures, security issues are addressed as:
  - Interaction Model – deals with performance and the difficulty of setting of time limits in a distributed system.
  - Failure Model – specification of the faults that can be exhibited by processes
  - Secure Model – discusses possible threats to processes and communication channels.

# ➢ Contents

1) Challenges
2) System Models Introduction
3) Architectural Models
   - Software Layers
   - System Architectures
     - Client-Server
       - ✓ Clients and a Single Server, Multiple Servers, Proxy Servers with Caches, Peer Model

# ❑ Architectural model

- The architecture of a system is its structure in terms of separately specified components.
  - Its goal is to meet present and likely future demands.
  - Major concerns are making the system reliable, manageable, adaptable, and cost-effective.
- Architectural Model properties:
  - Simplifies and abstracts the functions of individual components
  - The placement of the components across a network of computers
    - Define patterns for the distribution of data and workloads
  - The interrelationship between the components – i.e., functional roles and the patterns of communication between them.

# ❑ Architectural model

- An initial simplification is achieved by classifying processes as:
  - Server processes
  - Client processes
  - Peer processes
    - ✓ Cooperate and communicate in a symmetric manner to perform a task.

- This classification of processes identifies the responsibilities of each and hence helps us to assess their workloads and to determine the impact of failures in each of them.

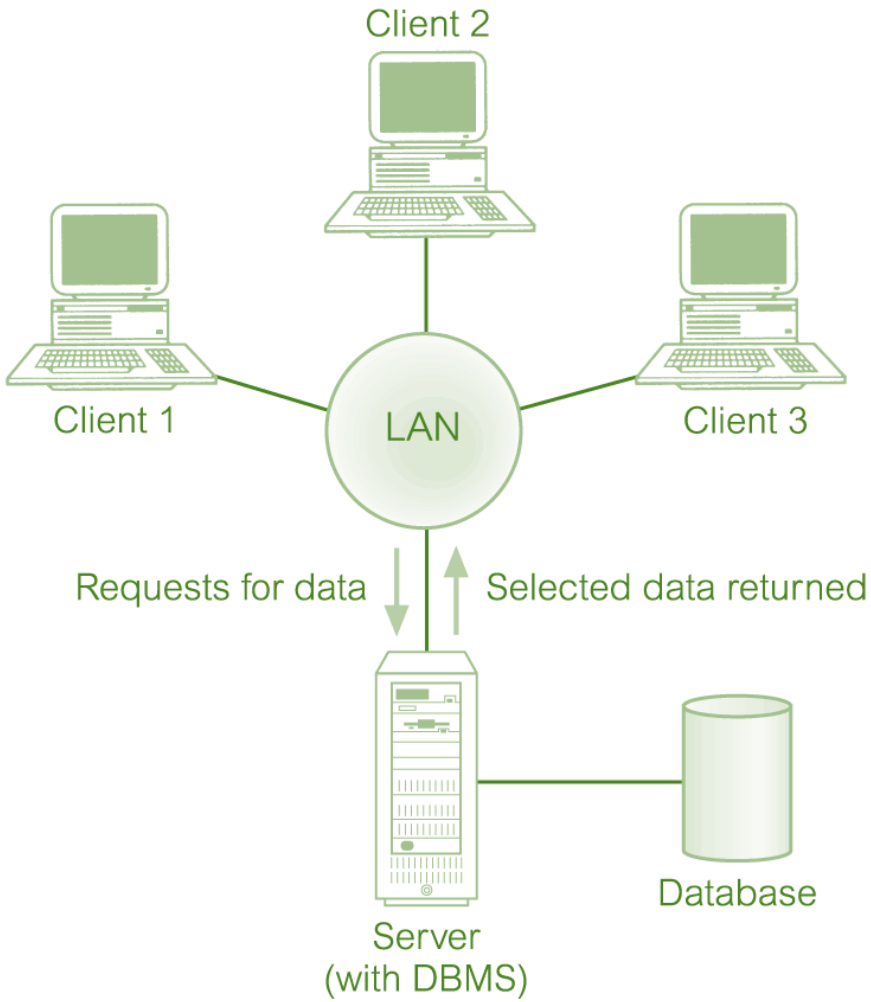
# ❑ Client-server model

- Dynamic systems can be built as variations on the client-server model:
  - The possibility of moving code from one process to another

- for example, clients can download code from servers and run it locally. Objects and the code that accesses them can be moved to reduce access delays and minimize communication traffic,

# ❑ Client-Server Model

| CLIENT | SERVER |
|---|---|
| Manages the user interface | Accepts and processes database requests from clients |
| Accepts and checks syntax of user input | Checks authorization |
| Processes application logic | Ensures integrity constraints not violated |
| Generates database requests and transmits to server | Performs query/update processing and transmits response to client |
| Passes response back to user | Maintains system catalog<br>Provides concurrent database access<br>Provides recovery control |




Database Management Systems

# ❑ Client-server model

- In Dynamic systems

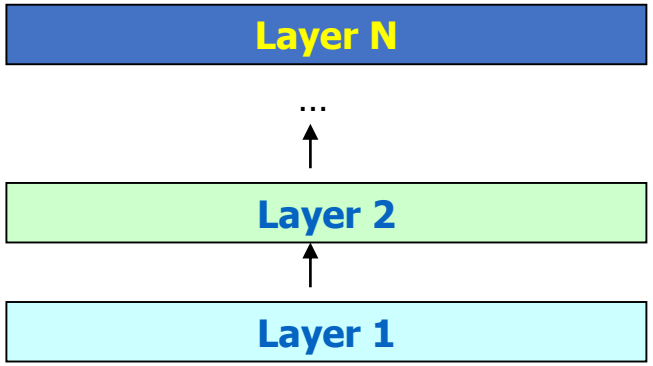
  - Some distributed systems are designed to enable computers and other mobile devices to be added or removed, allowing them to discover the available services and to offer their services to others.

# ➢ Contents

1) Challenges
2) System Models Introduction
3) Architectural Models
   - Software Layers
   - System Architectures
     - Client-Server
       - ✓ Clients and a Single Server, Multiple Servers, Proxy Servers with Caches, Peer Model

# ❑ Software Architecture and Layers

- The term software architecture referred:
  - Originally to the structure of software as layers or modules in a single computer.
- Breaking up the complexity of systems by designing them through layers and services
  - Layer: a group of related functional components
  - Service: functionality provided to the next layer.




(services offered to above layer)

## ❑ Software and hardware service layers in distributed systems

- A distributed service can be provided by one or more server processes, interacting with each other and with client processes in order to maintain a consistent system-wide view of the service's resources.

# ❑ Software and hardware service layers in distributed systems

- A server is a process that accepts requests from other processes.

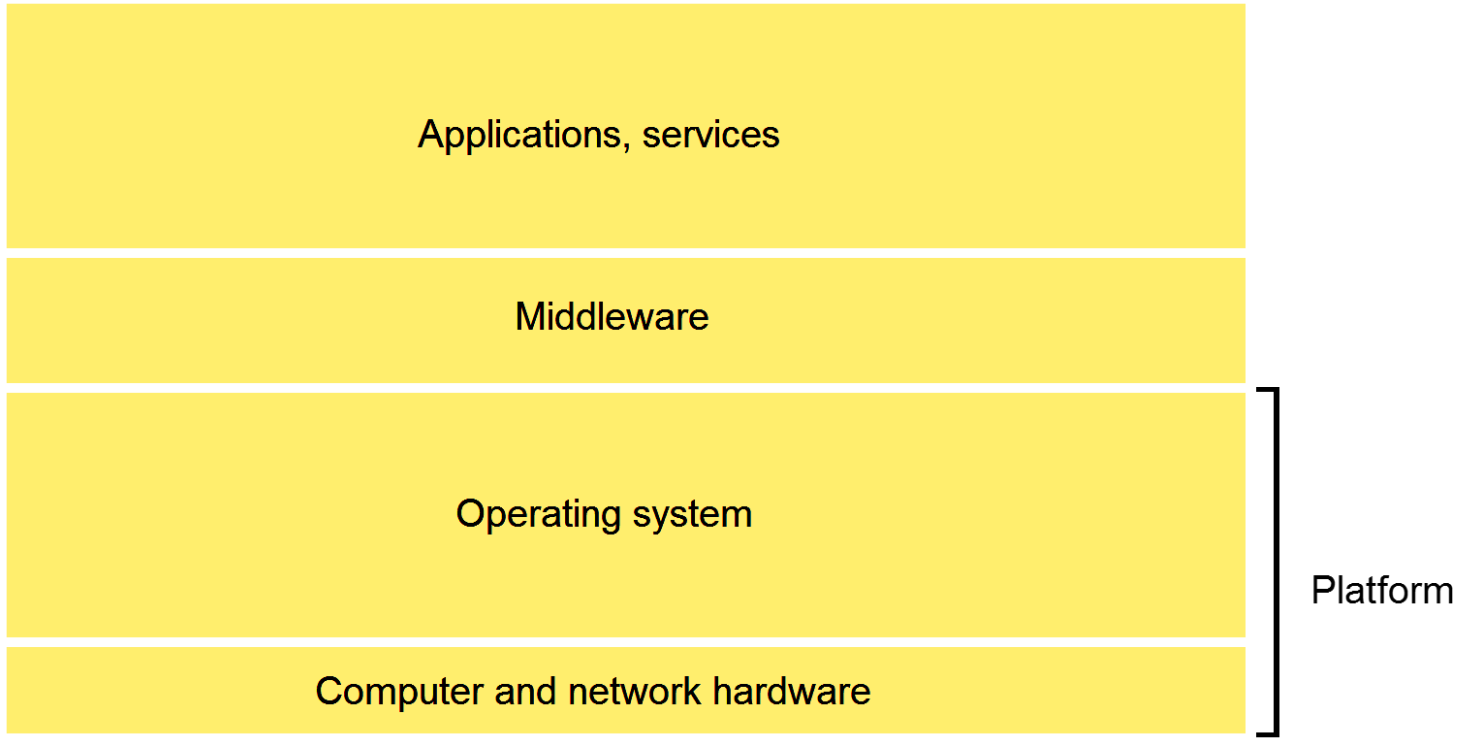



Figure shows Software and hardware service layers in distributed systems

# ❑ Platform

- The lowest hardware and software layers are often referred to as a platform for distributed systems and applications.
- These low-level layers provide services to the layers above them, which are implemented independently in each computer.
- Major Examples
  - Intel x86/Windows
  - Intel x86/Linux
  - Intel x86/Solaris
  - SPARC/SunOS
  - PowerPC/MacOS

# ❑ Middleware

- A layer of software whose purpose is to mask heterogeneity present in distributed systems and to provide a convenient programming model to application developers.

- Major Examples:
  - Sun RPC (Remote Procedure Calls)
  - OMG CORBA (Common Request Broker Architecture)
  - Microsoft D-COM (Distributed Components Object Model)
  - Sun Java RMI (Remote Object Invocation)
  - Modern Middleware:
    - IBM WebSphere
    - Microsoft .NET

# ➢ Contents

1) Challenges
2) System Models Introduction
3) Architectural Models
   - Software Layers
   - System Architectures
     - Client-Server
       - ✓ Clients and a Single Server, Multiple Servers, Proxy Servers with Caches, Peer Model

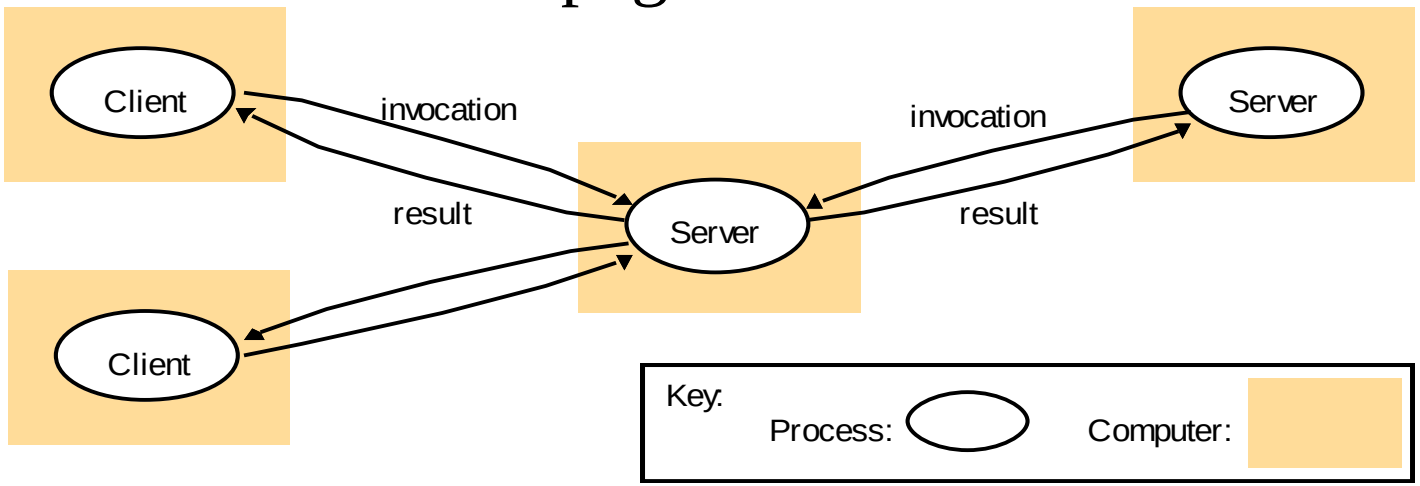
## ❑ System Architecture

- The most evident aspect of DS design is the division of responsibilities between system components (applications, servers, and other processes) and the placement of the components on computers in the network.

- It has major implication for:
  - Performance
  - Reliability
  - Security of the resulting system.

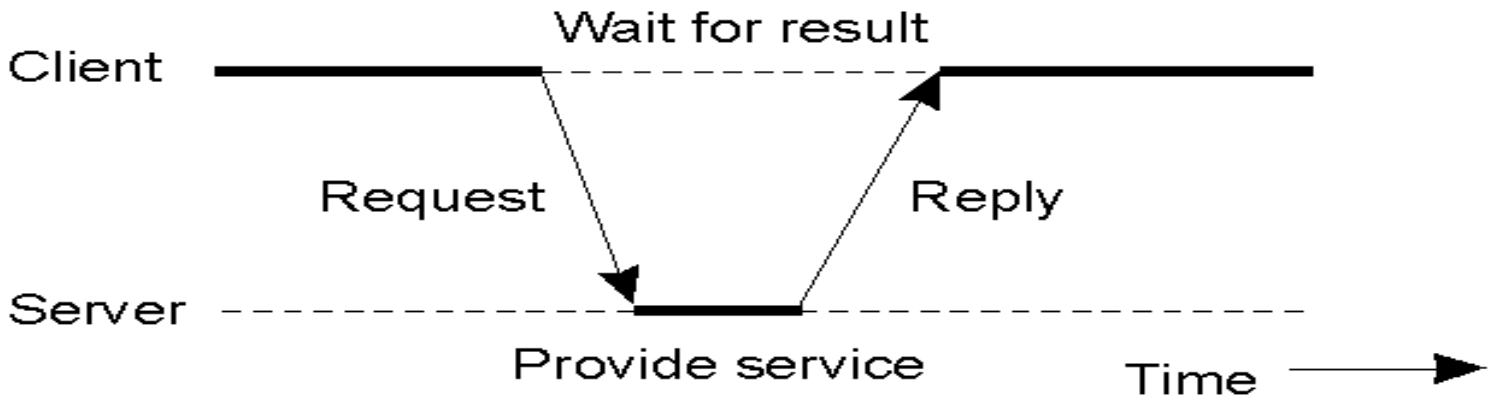
## ❑ Client Server Model

- Client process interact with individual server processes in a separate computer in order to access data or resource.
- The server in turn may use services of other servers.
- **Example**:
  - A Web Server is often a client of file server, that manages the files in which the web pages are stored.



## ❑ Clients and Servers

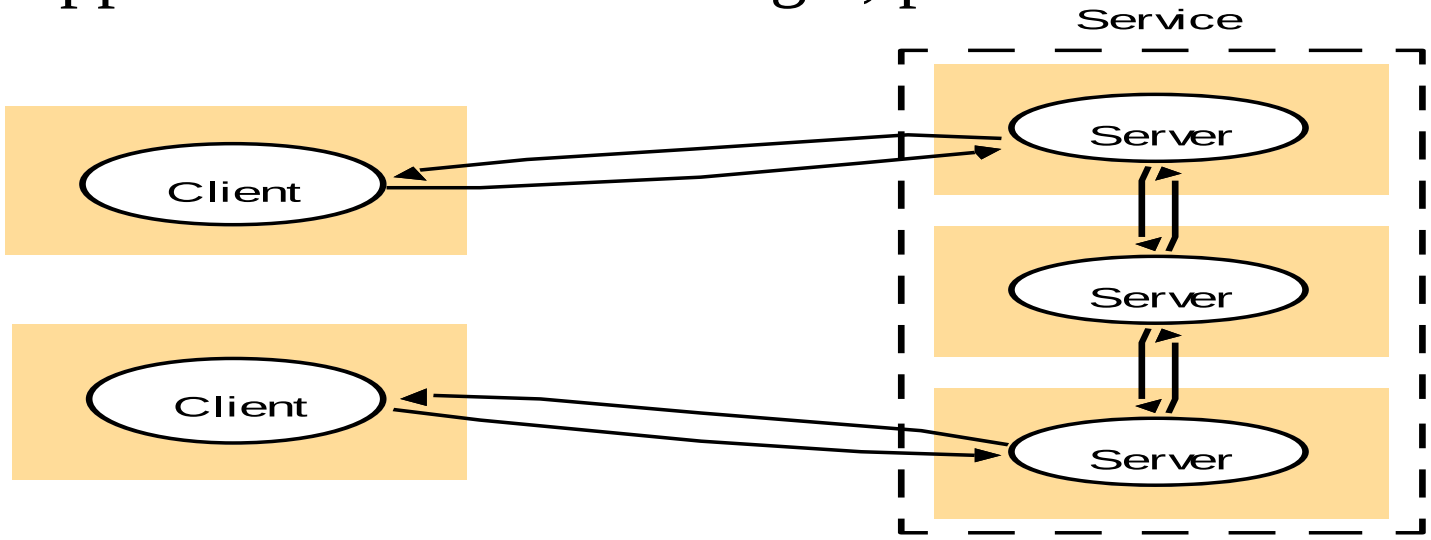
- General interaction between a client and a server.




- This is a typical interaction for a single threaded.
- A client requests some processing or information from a server that it needs.
- It waits in a blocking fashion for the reply containing the result, then it can proceed with its execution.

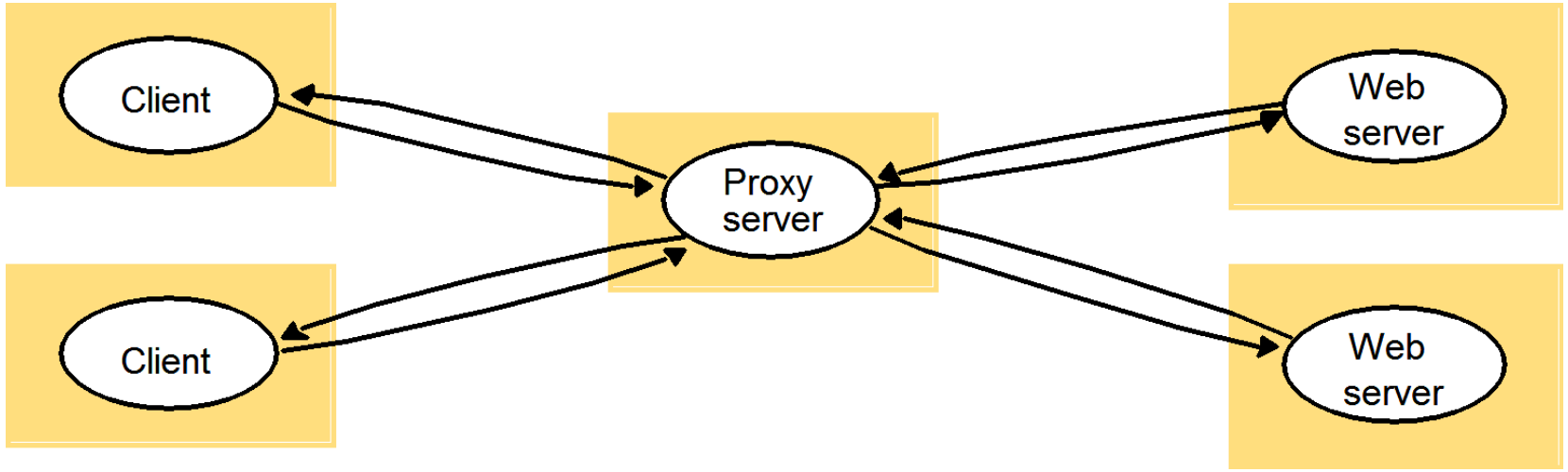
## ❑ Service provided by multiple servers

- Services may be implemented as several server processes in separate host computers.
- May provides multiple consistent copies of data in processes running in different computers.
- **Example**: applications such as Google, parallel databases Oracle



## ❑ Proxy Servers & Caches

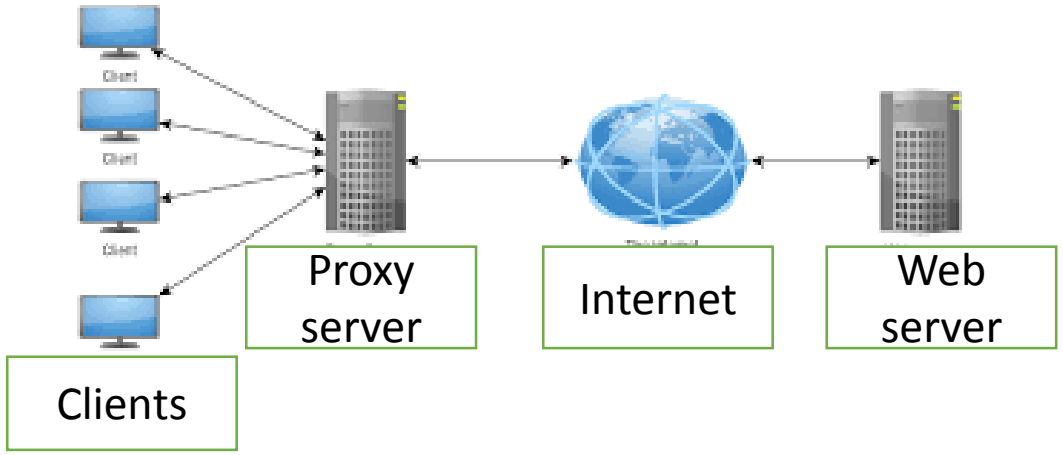
- Cache is a store of recently used data objects that is closer than the objects themselves. When a new object is received at a computer it is added to the cache store, replacing some existing objects if necessary




- The purpose of proxy servers is to increase availability and performance

## ❑ Proxy Servers & Caches

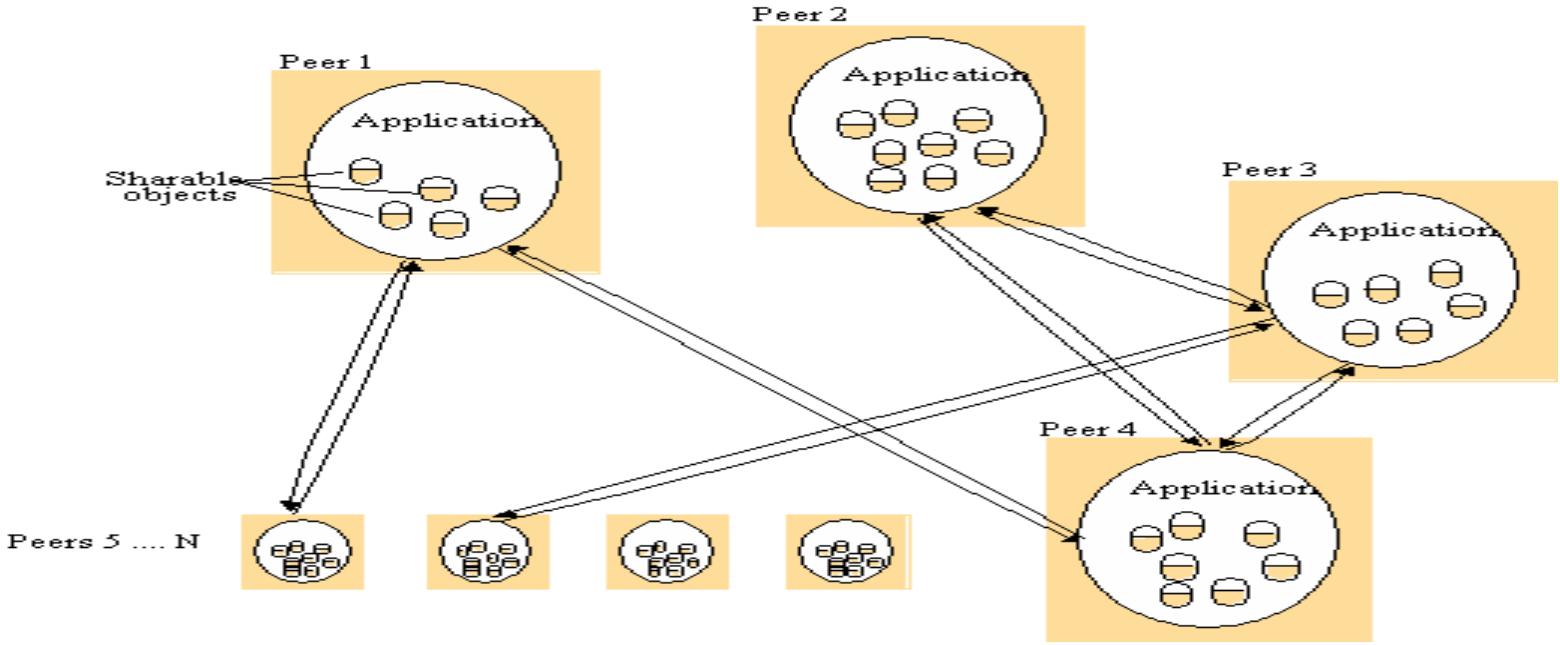
- This reduces the load on the web servers and improves the performance for end users by reducing the time taken for a dynamic request.




- Web browsers maintain a cache of recently visited web pages and other web resources in the client's local file system

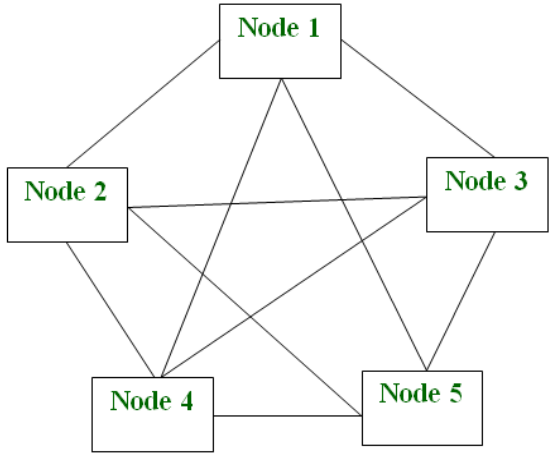
## ❑ Peer Processes

- All the processes play similar roles, interacting cooperatively as peers to perform a distributed activities or computations without distinction between clients and servers.
- Elimination of server processes.



## ❑ Peer Processes

- Can be used very effectively… for "swarm".
- No central point of failure (reliable)




P2P Architecture

- No central point of control (difficult to deny service for adversaries)
- Some peers will typically contribute more than others (super-peer)

Thank you