System Models

System models

Purpose

- illustrate/describe **common properties** and **design choices** for distributed system in a single descriptive model.

System Models

Two types of models

1. Architectural Models
2. Fundamental Models

Architectural model is concerned with the placement if its components and the relationships **between** them.

Fundamental models are concerned **with a** more formal description of the properties that are common in all of the **architectural models**.

2.2 Architectural System Model

- An architectural model of a distributed system is concerned with the placement of its parts and the relationships between them.
- **Examples**
  - Client-server
  - Peer-to-peer
- **Interaction Model**
  - Deals with performance and the difficulty to set time limits (e.g., in message delivery).
- **Failure Model**
  - Gives a precise specification of the faults of the processes and the links.
  - Defines reliable communication and correct processes.

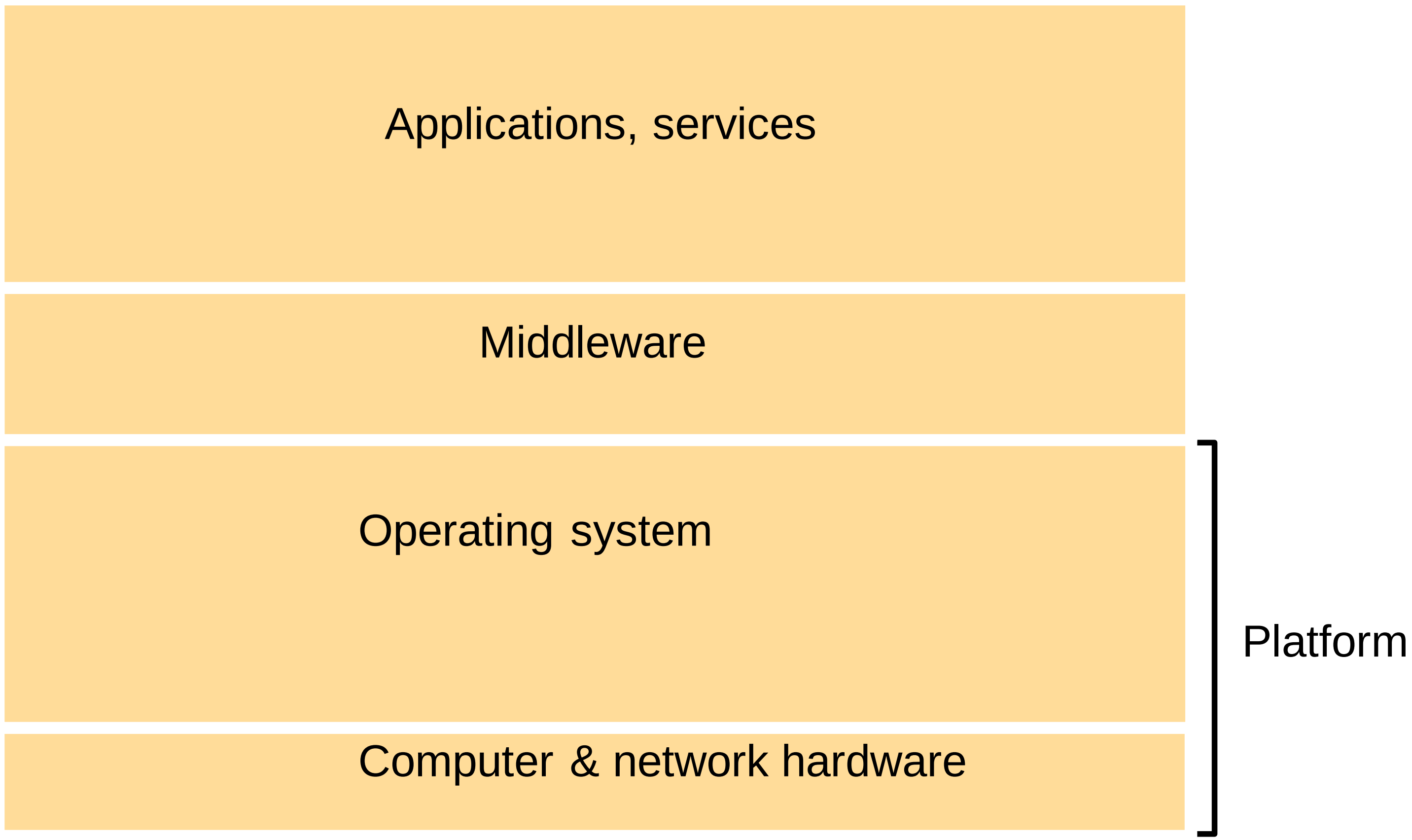
Architectural Models

- The architecture abstracts the functions of the individual components of the distributed system.
    - ensure that the structure will meet present and likely future demands
    - make the system reliable, manageable, adaptable, and cost-effective
- Classification of processes
    - Servers, clients, peers
    - Identifies responsibilities, helps to assess their workloads, determines the impact of failures in each of them.

Software Layers

- **Software architecture** refers to services offered and requested between processes located in the same or different computers.
  - structuring of software as layers or modules
  - Service layers
- Distributed service
  - One or more server processes
  - Client processes
- Platform -> the lowest level hardware and software layers
  - Examples: Intel x86/Windows, Intel x86/Solaris, PowerPC/MAC OS, Intel x86/Linux
  - Lowest level layers that provide servicesto other higher layers
- Middleware
  - masks heterogeneity & provides a convenient programming model
  - Provides useful building blocks:
    - Remote method invocation, communication between a group of processes, notification of events, partitioning, placement and retrieval of data or objects, replication, transmission of multimedia data in real time

2.2.1 Software and hardware service layers in DS



Applications, services

Middleware

Operating system

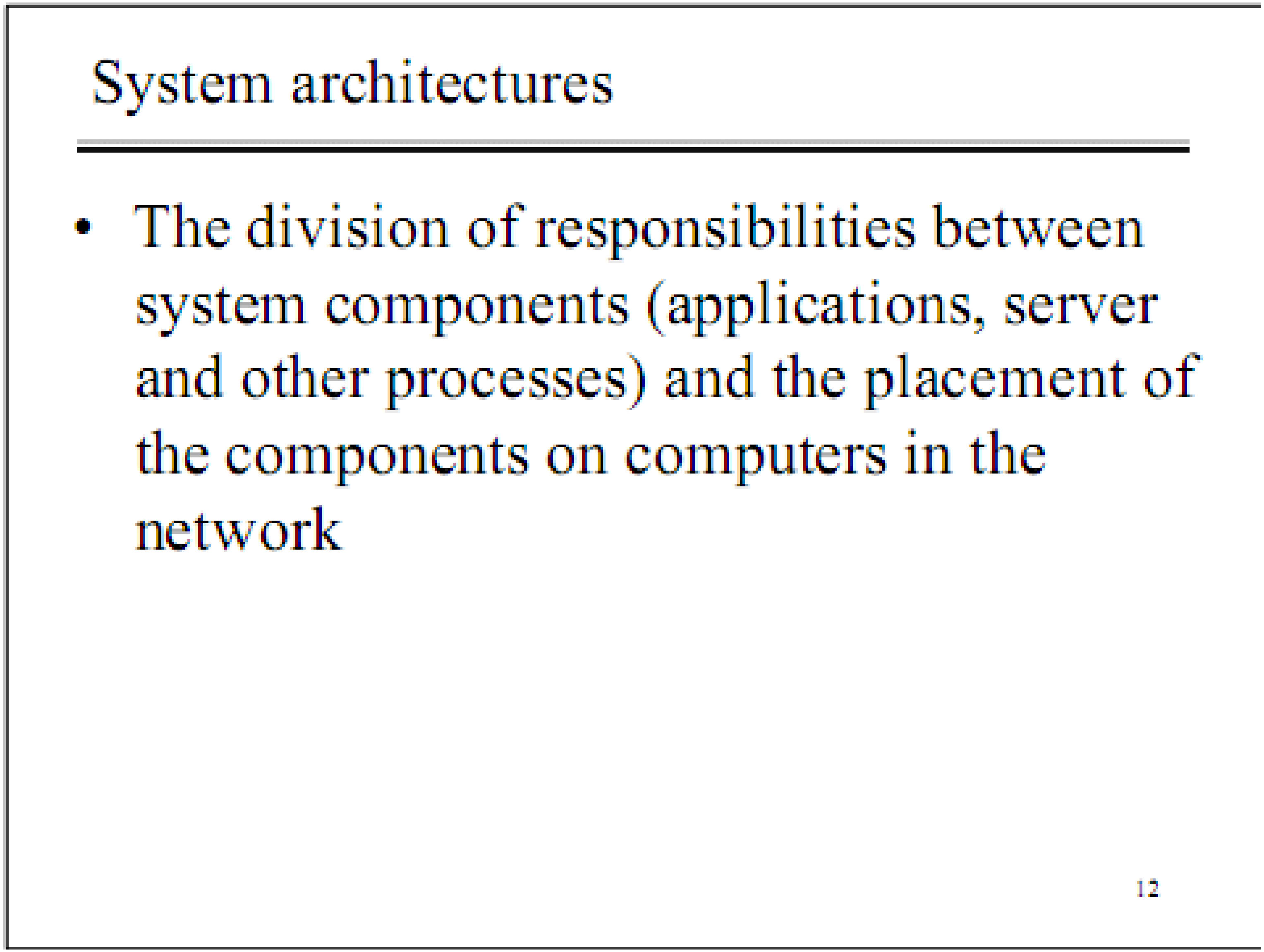
Computer & network hardware

Platform

Middleware

- Eg CORBA,
- Java RMI,
- DCOM

2.2.2 System Architectures



System architectures

- The division of responsibilities between system components (applications, server and other processes) and the placement of the components on computers in the network

System Architectures

1. Client- server model
2. Peer processes

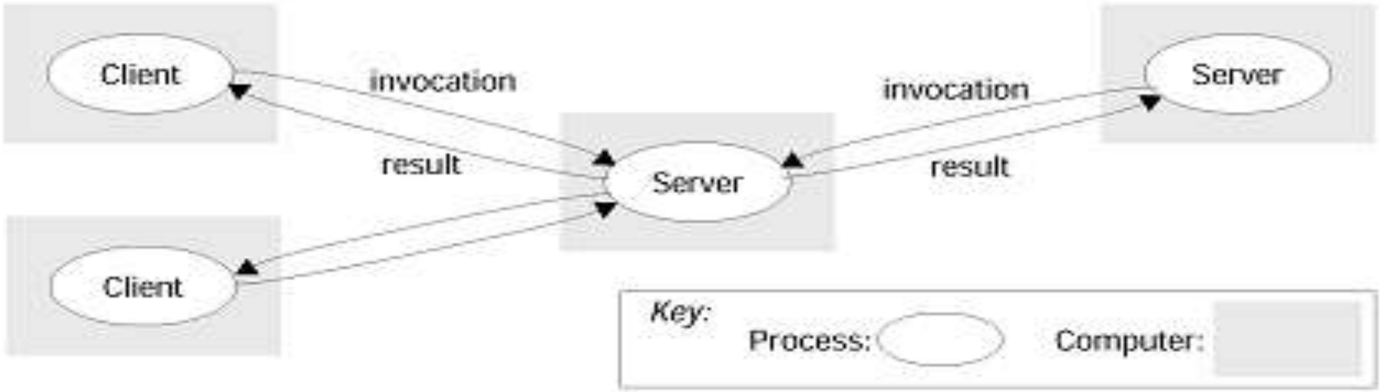
1. Client- server model

- The system is structured as a set of processes, called servers, that offer services to the
  users , called clients.
- The client-server model is usually based on a simple request/reply protocol, implemented with send/receive primitives or RPC or RMI.
- The client processes interact with individual server processes in
  separate host computers in order to access the shared resource that
  they manage
- The server does the work and returns a result or error code if the work could not be performed.
- A server can itself request services from other servers, thus the server itself acts like a client.
- Eg: Web Server is often a client of Local file server, Apache Database Server is client of HTTP server

Client-server model

- Servers can be clients, and clients can be servers.

Clients invoke individual servers

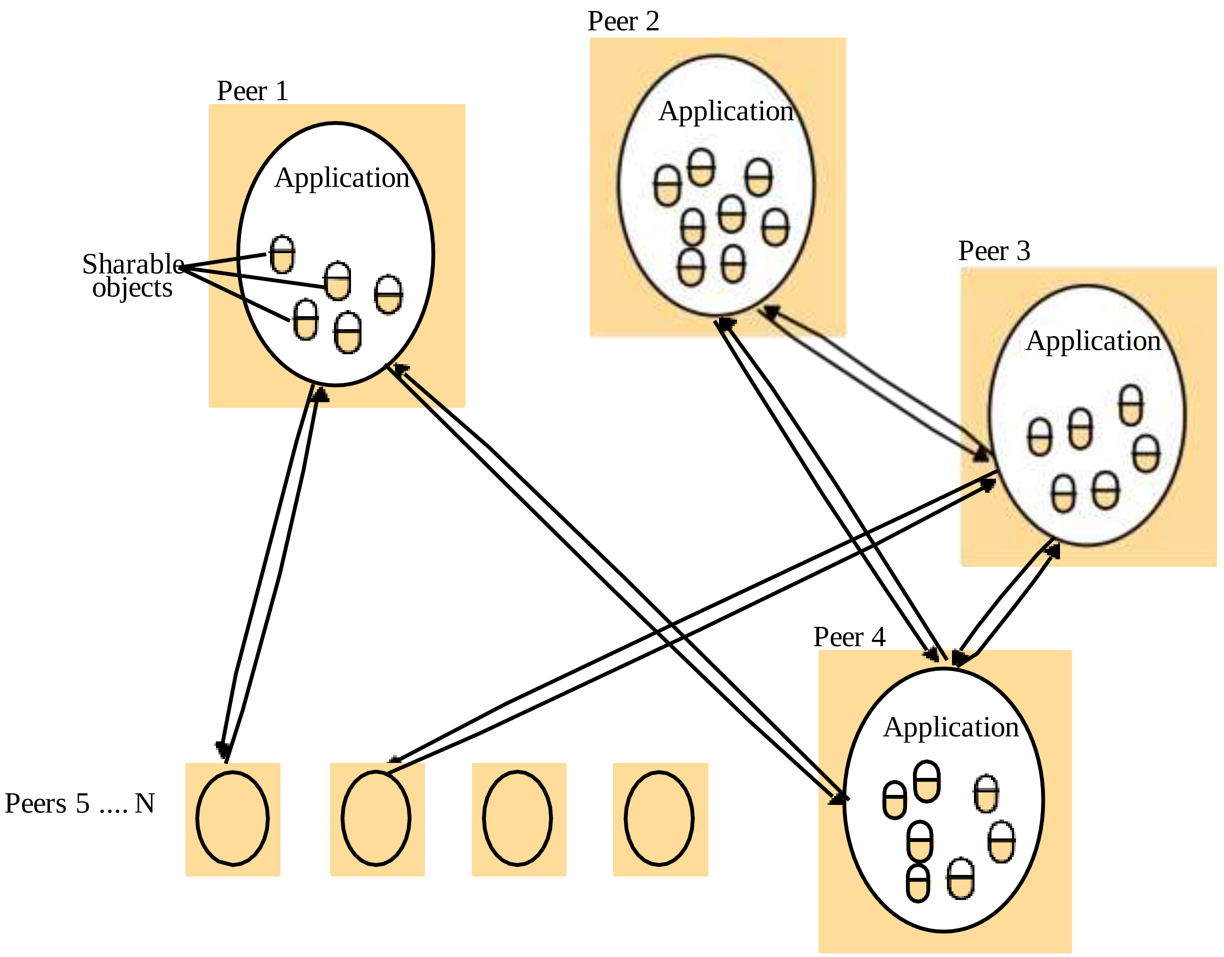


4. Peer processes

- All processes(objects) play similar roles **without distinction between** client or servers.
- The aim of peer to peer architecture is to exploit the resources in a large no. of participating computers for fulfillment of given task or activity.
- A **large number of data objects are shared**; any individual computers hold only a small part of the application database.
- **It distributes shared resources widely**.
- **It shares** computing and communication loads.

Ex Napster application for sharing digital Music files

A distributed application based on peer Processes



Variations on the client-server model

Including :-

1 Services provided by multiple servers

2 Proxy Servers and Caches

3 Mobile Code

4 Mobile Agents

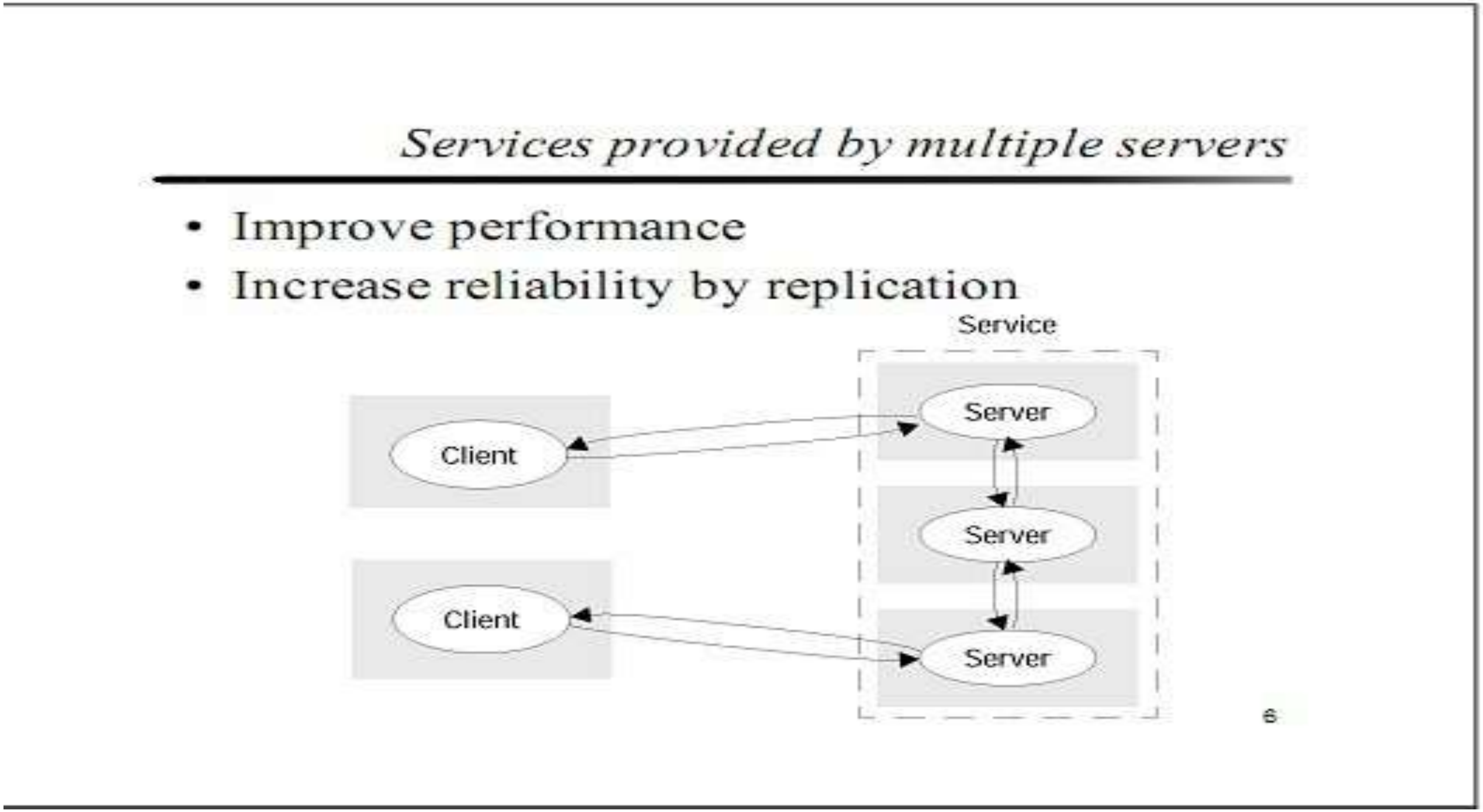
5 Network Computers

6 Thin Clients

7 Mobile devices and spontaneous networking

1. Services provided by multiple servers

- Services may be implemented **as several server processes** in separate host computers interacting as necessary to provide a service to client processes.



Services provided by multiple servers

- Improve performance
- Increase reliability by replication

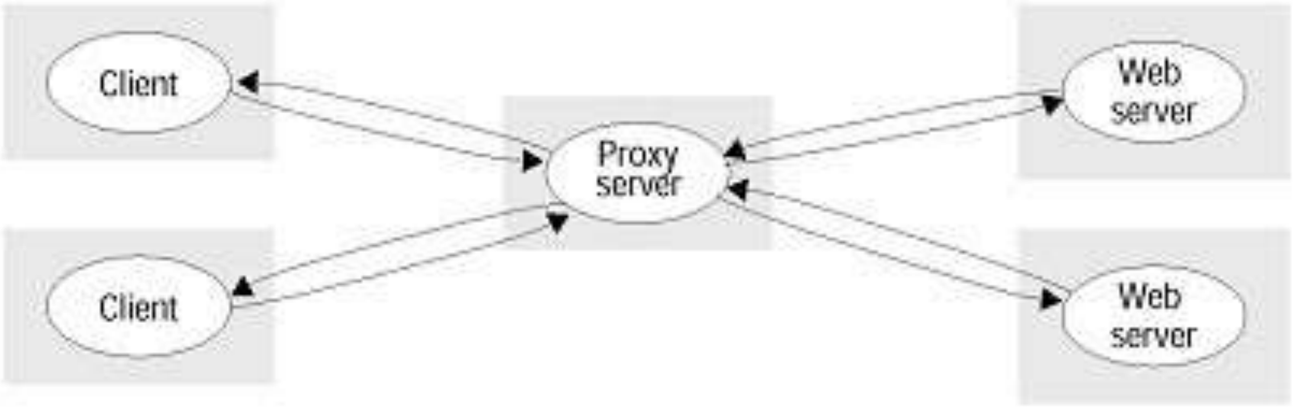
2. Proxy Servers and Caches

Proxy Servers

- proxy servers are used to **increase availability and performance of the services** by reducing the load on the network and web-server.
- Proxy server provides **copies(replications) of resources** which are managed by other server.

Proxy servers and cache

- Proxy servers are used to increase availability and performance of the service by reducing the load on the network and web-server



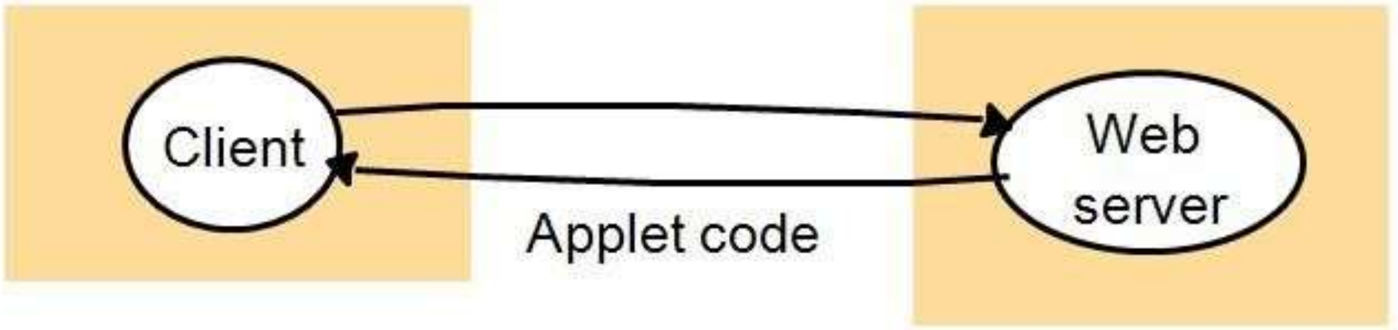
Cache

- A store of recently used data objects that is closer to the client process than those remote objects.
- When an object is needed by a client process the caching service checks the cache and supplies the objects from there in case of an up-to-date copy is available.
- Purpose : is to increase performance and availability **by avoiding** **frequent accesses to remote servers.**

3. Mobile code

- It is used to refer to code that can be sent from one computer to another and run at the destination.
- Example : java applets

client request results in the downloading of applet code



<u>**Step : 1-**</u>-The user running a browser selects a link to an applets whose code is stored on a web server.

- The code is downloaded to the browser and runs there.

client interacts with the applet

Client
Applet
Web server

Step:2-Client interacts with the applet.

Advantages & Disadvantages

Advantages:

1. Remote invocations are replaced by local ones.
2. Good interactive response.
3. Does not suffer from the delays.

Disadvantages:

- Security threat to the local resources in the destination computer.

4 Mobile Agents

- Mobile agent is a running program that travels from one computer to another **carrying out a task to someone's behalf**, such as collecting information, eventually returning with the results.(eg: Google form)
- Mobile agent is a complete program(including both code & data) **that can work independently.**
- Mobile agent **can invoke local resources/data.**

Advantages:

- Reduce communication cost and time by replacing remote invocation with localones.

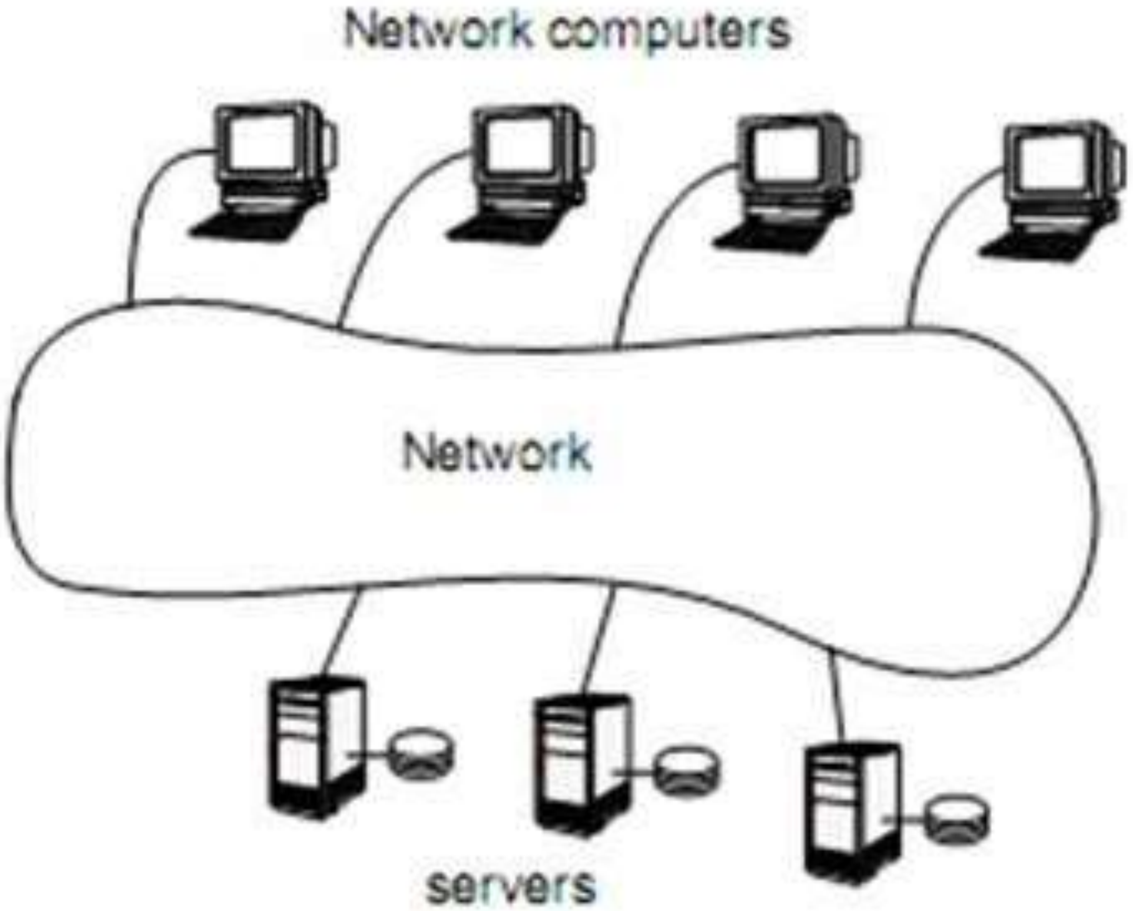
Disadvantages:

1. Limited applicability
2. Security threat ofthe visited sites resources

5 Network Computers

- Network computers **do not store locally** operating system or application code.
- **Downloads** its operating system and any applications needed by the user **from a remote file server**.
- **Applications run locally** but files are managed by a remote file server.
- **Users can migrate** from one network computer to another.
- Its processor and memory capacities **can be restricted to reduce its cost.**

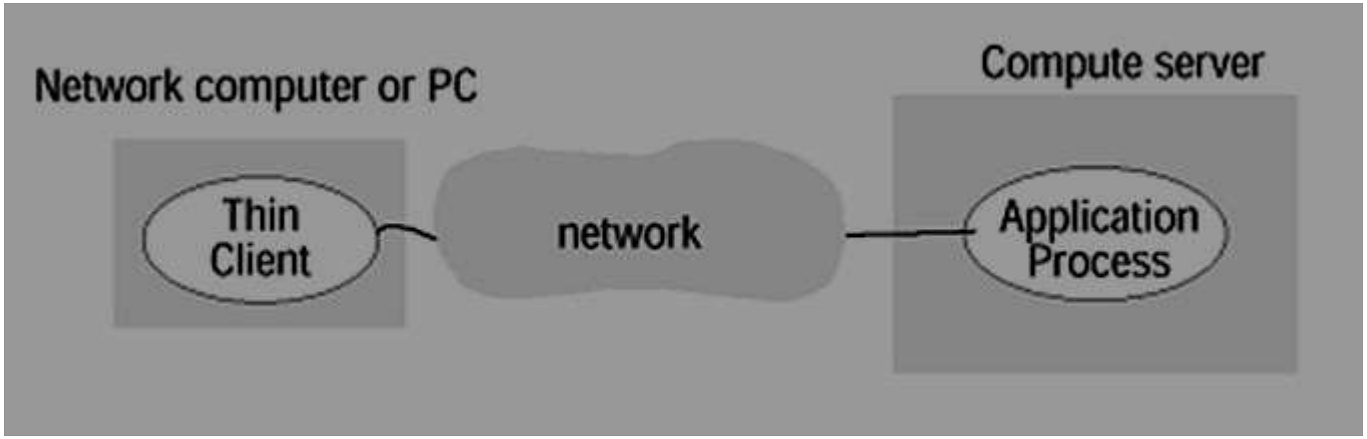
Network Computers



Advantages

- The network computers can be **simpler**, with limited capacity, it does not need even a local hard disk.
- User can **login from any computers**.
- **No user effort** for **software management** or **administration**.

6. Thin Clients



- Thin clients do not download code (operating system or application) from the server to run it locally.
- All code is run on the server, in parallel for several clients.
- The thin client only runs the user interface.
- It is same as network computer scheme but instead of downloading the application code into the user’s computer, it runs them on a server machine, compute server.

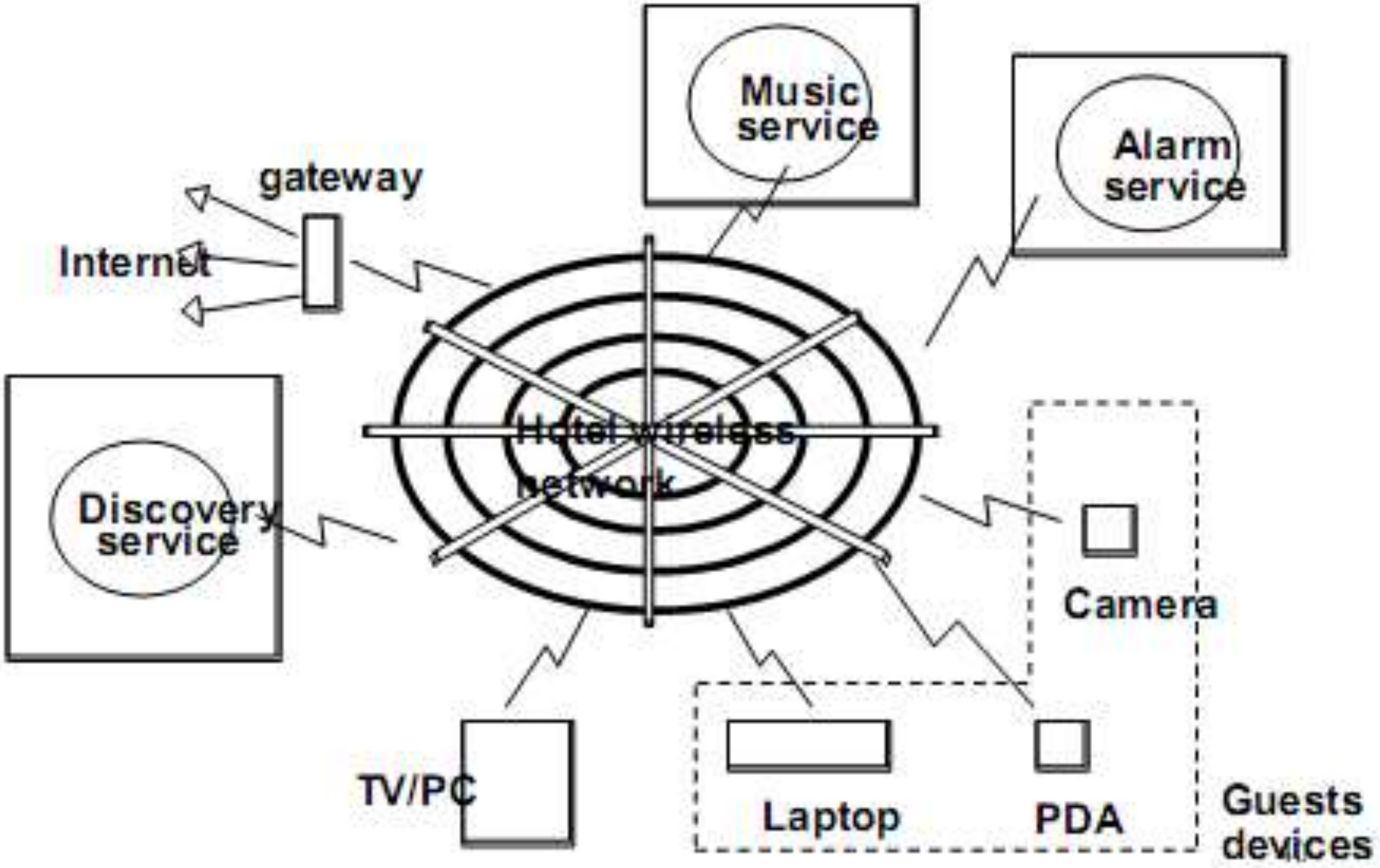
Disadvantages:

High latencies:- increasing of the delays in highly interactive graphical applications.

7. Mobile Devices and Spontaneous Networking

- **Mobile Devices** : Mobile devices are hardware computing components that move between physical locations.
- **Spontaneous Network**: integrate mobile devices and other devices into a given network.

Spontaneous networking in a hotel



Advantages & Disadvantages

Advantages:

1. Easy connection to a local network.
2. Easy integration with local services.

Disadvantages:

3. Limited connectivity
4. Less Security and privacy

2.2.5. Design Requirements for Distributed Architectures

Including

1. Performance issues
2. Quality of services
3. Use of caching & replication
4. Dependability issues

Design Requirements for Distributed Architectures

1.Performance Issues

- Responsiveness
  - The speed of a remote invocation depends on:
    - The load and performance of the server and network
    - Delays in all the software components (client and server operation systems and middleware, code of the process that implements the service)
  - Transfer of data is slow
- Throughput
  - Rate at which computational work is done
  - Fairness
- Balancing of Computational Loads
  - Applets remove load from the server
  - Use several computers to host a single service

Design Requirements for Distributed Architectures

- 2.Quality of Service
  - Reliability & Security
    - ability of a system to perform and maintain its function in every circumstance
  - Performance
    - Ability to meet Timeliness guarantees
  - Adaptability
    - the ability of a system to adapt itself efficiently and fast to changed circumstances

Design Requirements for Distributed Architectures

3.Use of Caching and Replication

- Cached copies of resources should be kept up-to-date when the resource at a server is updated.
- A variety of cache-coherency protocols are used to suit different applications.

Web Caching Protocol

- Web browsers and proxy servers cache responses to client requests from web servers
- The cache-consistency protocol can provide browsers with fresh copies of the resources held by the web server, but for performance reasons the freshness condition can be relaxed.
- A browser or proxy can validate a datum with the server.
- Web servers assign approximate expiry times to their resources
- The expiry time of the resource and the current time at the server are attached to the response.

Design Requirements for Distributed Architectures

4. Dependability Issues

1) Correctness

2) Fault Tolerance: Dependable applications should continue to function correctly in the presence of faults in hardware, software and networks.

- Reliability is achieved through Redundancy
  - Multiple computers – multiple communication paths
  - Several replicas of a data item
  - But redundancy is costlier.

3) Security

- Safety (Confidentiality) – absence of incorrect behavior
- Integrity – absence of improper system alteration
- Availability – readiness for correct service

2.3 Fundamental Models

- A model contains only the essential ingredients needed to understand and reason about some aspects of a system's behavior.
- A system model has to address the following:
  - What are the main entities in the system?
  - How do they interact?
  - What are the characteristics that affect their individual and collective behavior?
- Purpose
  - Make explicit all the relevant assumptions about the system we are modeling
  - Make generalizations concerning what is possible or impossible, given those assumptions.
    - General purpose algorithms
    - Desirable properties
- Interaction
  - Communication takes place with delays
  - Maintaining the same notion of time across all nodes of a distributed system is difficilt.
- Failure
- Security

Types of FUNDAMENTAL MODELS:

Interaction: Computation occurs within processes; the processes interact by passing messages, resulting in communication (information flow) and coordination (synchronization and ordering of activities) between processes.

Failure**:** The correct operation of a distributed system is threatened whenever a fault occurs in any of the computers on which it runs (including software faults) or in the network that connects them. Our model defines and classifies the faults.

Security: The security model defines and classifies the type of attacks that may take place, providing a basis for the analysis of threats to a system and for the design of systems that are able to resist them.

Interaction Model

- Multiple server processes may cooperate with one another to provide a service;
- the examples include , the Domain Name System, which partitions and replicates its data at servers throughout the Internet,
- and Sun’s Network Information Service, which keeps replicated copies of password files at several servers in a local area network.

Contd…

Factors affecting interacting processes are,

- Communication performance
- No global notion of time

Communication over computer network has the following **performance characteristics,**

- Latency(Time taken by first string of bits to reach its destination+delay in accessing n/w+ time taken by os communication services at both ends)
- Bandwidth(Amount of data that can be sent in unit time)
- Jitter(Variation in packet delivery)

Computer clocks and timing events

- Each computer in a distributed system has its own internal clock, which can be used by local processes to obtain the value of the current time.
- Two processes running on different computers can associate timestamp with their events.
- Even if two processes read their clock at the same time, their local clocks may supply different time. This is because computer clock drift from perfect time and their drift rates differ from one another.
- Two variations of interaction model are,

  Synchronous Distributed system and

  Asynchronous Distributed systems

Synchronous distributed systems

- It has a strong assumption of time
- The time to execute each step of a process has known lower and upper bounds.
- Each message transmitted over a channel is received within a known bounded time.
- Each process has a local clock whose drift rate from real time has a known bound.

Asynchronous distributed system

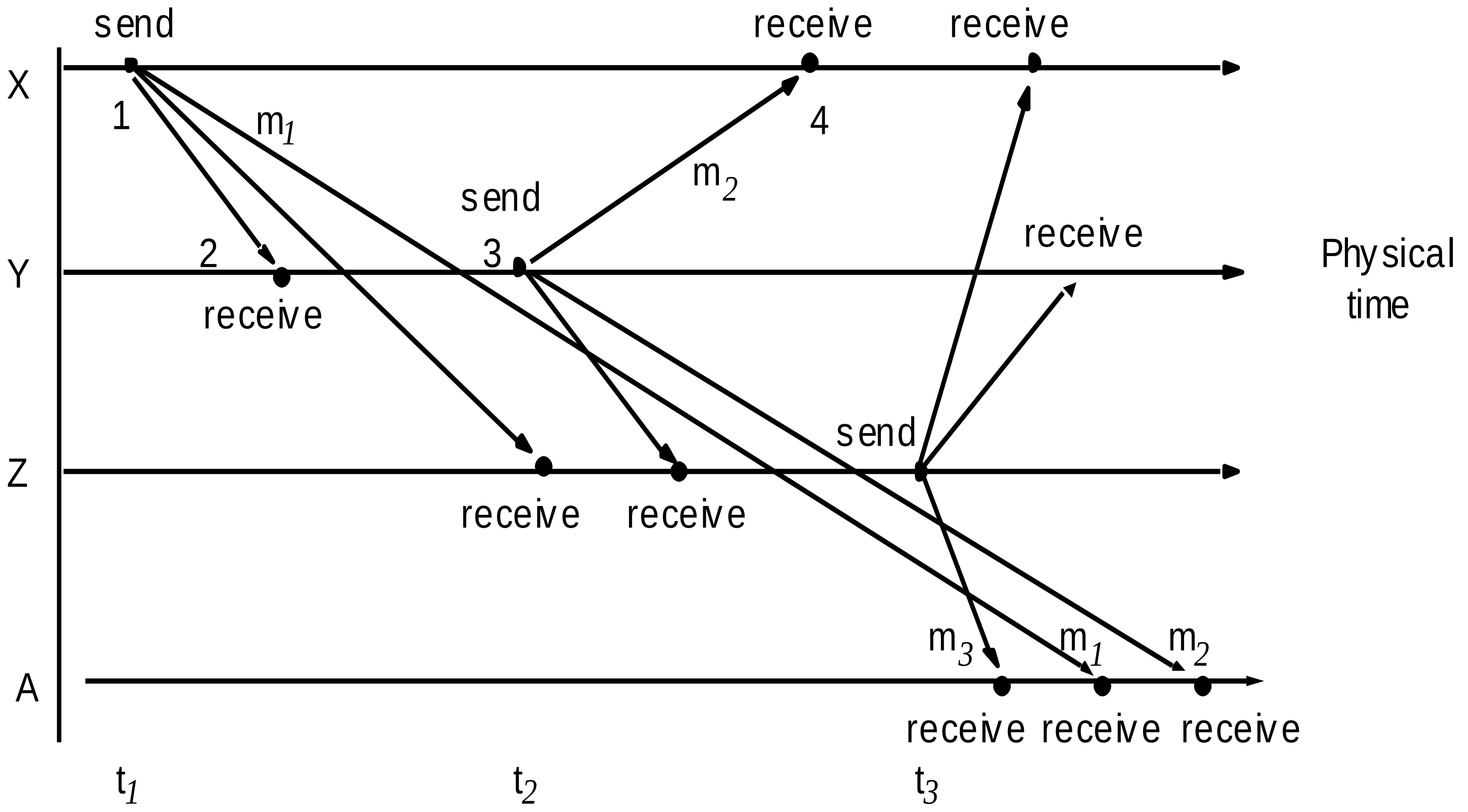
- It has no assumption about time.
- There is no bound on process execution speeds.
- There is no bound on message transmission delays.A message may be received after an arbitrary long time.
- There is no bound on clock drift rates. The drift rate of a clock is arbitrary.

Event Ordering

- In many cases, we are interested in knowing whether an event (sending or receiving a message) at one process occurred before, after, or concurrently with another event at another process.
- The execution of a system can be described in terms of events and their ordering despite the lack of accurate clocks.
- For example, consider a mailing list with users X, Y, Z, and A.

1. User X sends a message with the subject Meeting.
2. Users Y and Z reply by sending a message with the subject RE: Meeting

Figure: Real-time ordering of events



Contd..

Since clocks cannot be synchronized perfectly across distributed system, Lamport proposed a model of logical time that provides ordering among events in a distributed system.

e.g. in the previous example we know that the message is received after it was sent. Hence a logical order can be derived here,

x sends m1 before y receives m1

Y sends m2(reply) before x receives m2(reply).

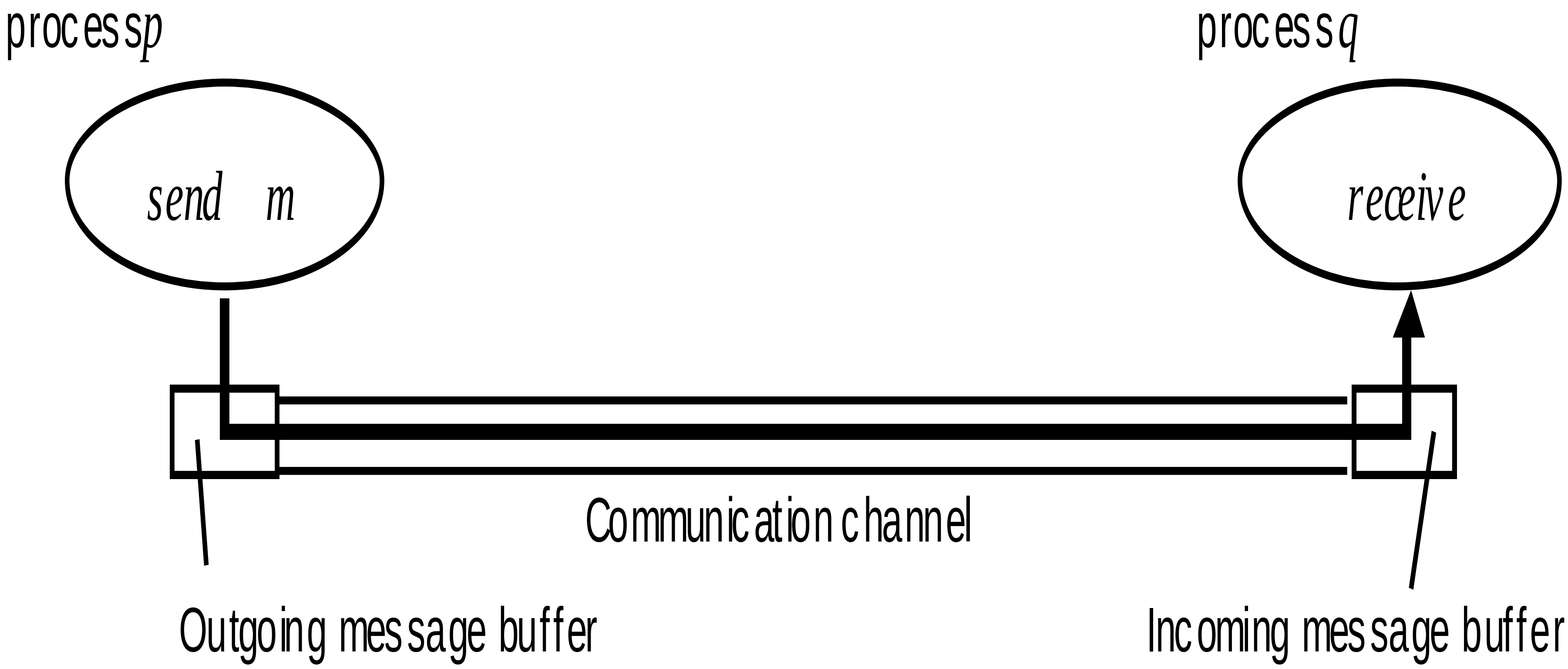
We also know that reply are sent only after receiving message.

Hence, we say that "Y receives m1 before sending m2".

Failure model

- In a distributed system both processes and communication channels may fail.
- Failure model defines the types of failure,
- Omission failure
- Arbitrary Failure
- Timing Failure

Figure Processes and Channels



Omission and arbitrary failures

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a*send*, but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Timing failures

Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

Masking Failures – Reliability of one-to-one Communication

- A service masks a failure, either by hiding it all together or by converting it into a more acceptable type of failure.
  - Checksums are used to mask corrupting messages -> a corrupted message is handled as a missing message
  - Message omission failures can be hidden by re- transmitting messages.
- The term reliable communication is defined in terms of validity and integrity as follows:
  - Validity: any message in the outgoing buffer is eventually delivered to the incoming message buffer
  - Integrity: the message received is identical to one sent, and no messages are delivered twice.

Security Model

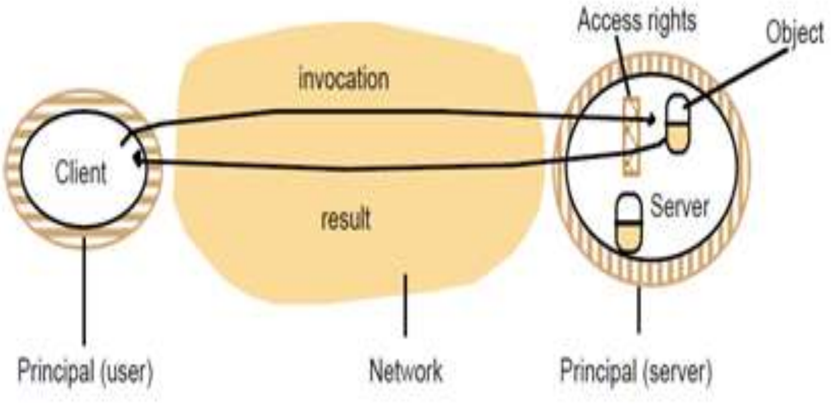
The security of a distributed system can be achieved by securing the processes and the channels used for their interactions and by protecting the objects that they encapsulate against unauthorized access

Protecting Objects: AS shown in the diagram the server manages a collection of objects on behalf of some users. Users can run client programs that send invocations to the servers to perform operations on the objects.
The server carries out the operations specified in each invocation and sends result to client.

Objects are intended to be used in different ways by different users. To support this access rights specify who is allowed to perform the operations of an object .Ex read or write its state

Protecting Objects(cont..)

Figure 2.17
Objects and principals



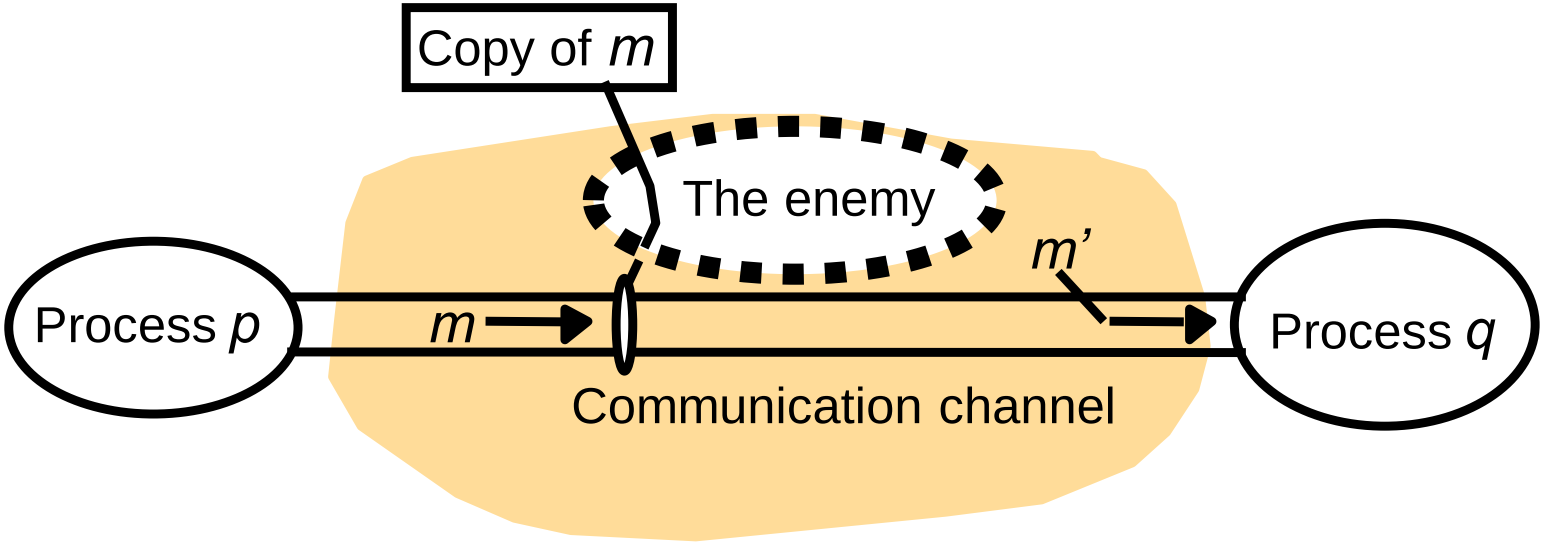
Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5
© Pearson Education 2012

Security Model(cont..)

The enemy

The enemy is capable of sending any message to any process and reading or copying any message between a pair of processes as shown in diagram
Such attacks can be made simply by using a computer connected to a network to run a program that reads network messages addressed to other computers on the network or a program that make false requests to services and purporting to come from authorized users

Figure 2.18
The enemy



Security Model (Cont..)

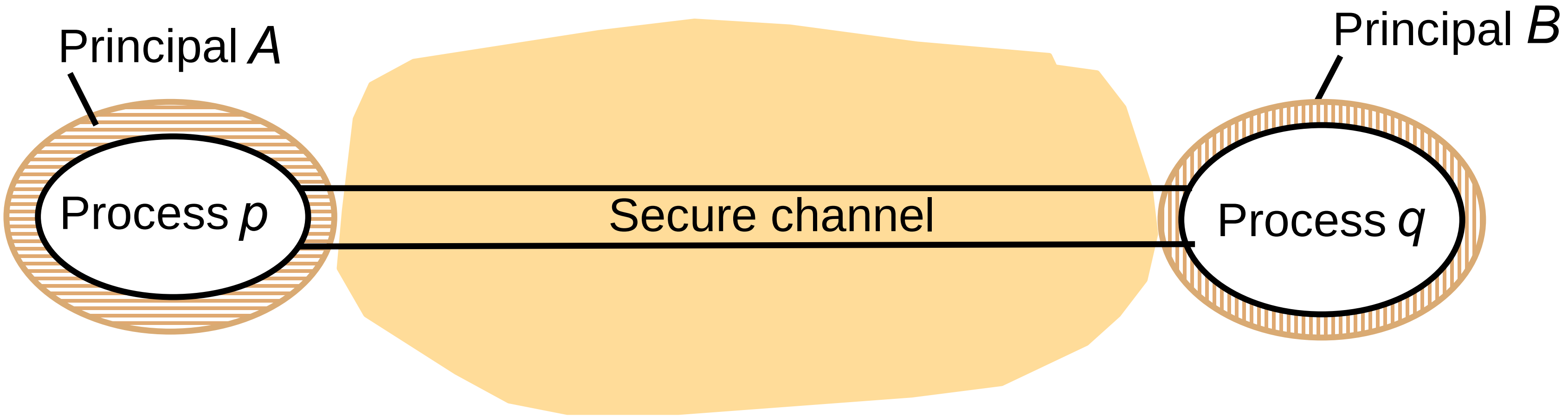
Secure Channels :Encryption and Authentication are used to build secure channels as a service layer on top of existing communication services. It has the following properties

•Each of the processes knows identity of the principal on whose behalf the other process is executing

•A secure channel ensures privacy and integrity

•Each message includes physical or logical time stamp to prevent messages from being replayed or reordered

Figure 2.19
Secure channels


Principal *A*
Principal *B*
Process *p*
Secure channel
Process *q*