SYSTEM MODEL

محاضرة الثلاثاء ٢٠٢٢/٣/٢٩

From Chapter 2 of Distributed Systems Concepts and Design,

By G. Coulouris, J. Dollimore, T. Kindberg and G. Blair

Published by Addison Wesley/Pearson Education

Topics

- Introduction
- Architectural Models
- Fundamental Models

Introduction

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

- Examples include:
    - Client-Server model
    - Peer-to-Peer model

Introduction

- Variations of client-sever model can be formed by:
  - The partition of data or replication at cooperative servers
  - The caching of data by proxy servers and clients
  - The use of mobile code and mobile agents

Introduction

- Fundamental Models deal with a more formal description of the properties that are common in all of the architectural models.
- Some of these properties in distributed systems are:
  - There is no global time in a distributed system.
  - All communication between processes is achieved by means of messages.

Introduction

- Message communication in distributed systems has the following properties:
  - Delay
  - Failure
  - Security attacks

Introduction

- Message communication issues are addressed by three models:
  - Interaction Model
    - It deals with performance and with the difficulty of setting time limits in a distributed system.
  - Failure Model
    - It attempts to give a precise specification of the faults that can be exhibited by processes and communication channels.
  - Security Model
    - It discusses possible threats to processes and communication channels, and provides a basis for analysis of threats, to design a systems that are able to resist threats.

Architectural Models

- Introduction
- System Architectures
- Variants of Client Sever Model

Architectural Models-Intro

- The architecture of a system is its structure in terms of separately specified components and their interrelationships.
  - The overall goal is to ensure that the structure will meet present and likely future demands on it.
  - Major concerns are to make the system:
    - Reliable
    - Manageable
    - Adaptable
    - Cost-effective

System Architectures

- Client-Server model
  - Most often architecture for distributed systems.
  - Client process interact with individual server processes in a separate host computers in order to access the shared resources that they manage.
  - Servers may in turn be clients of other servers.
    - E.g. a search engine can be both a server and a client: it responds to queries from browser clients and it runs web crawlers that act as clients of other web servers. (Figure 3,4)

System Architectures

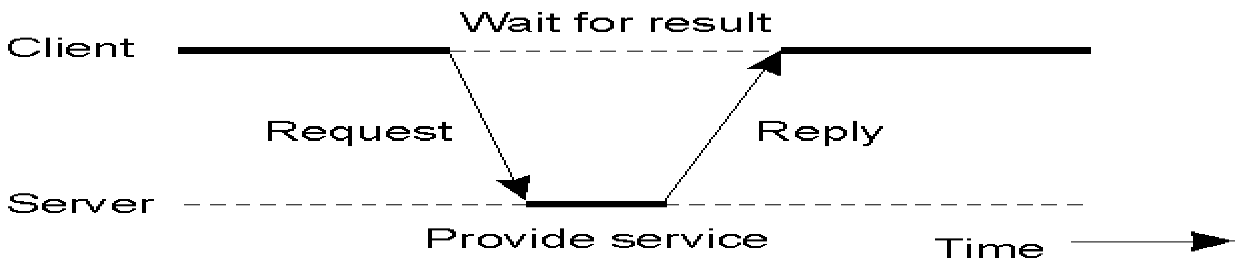


Figure 3. General interaction between a client and a server.

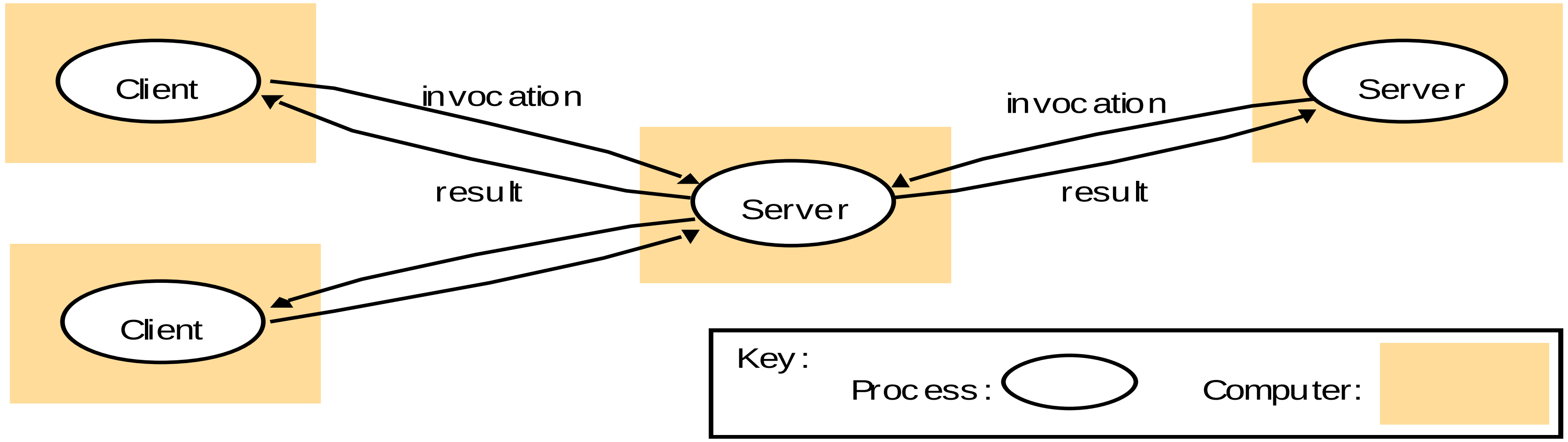


Figure 4. Clients invoke individual servers

Couloris,Dollimore, Kindberg and Blair Distributed Systems: Concepts and Design , Pearson Education

System Architectures

- Peer-to-Peer model
    - All of the processes play similar roles, interacting cooperatively as peers to perform a distributed activities or computations without any distinction between clients and servers or the computers that they run on.
    - E.g.,BitTorrent file-sharing system.

(Figure 5)

System Architectures

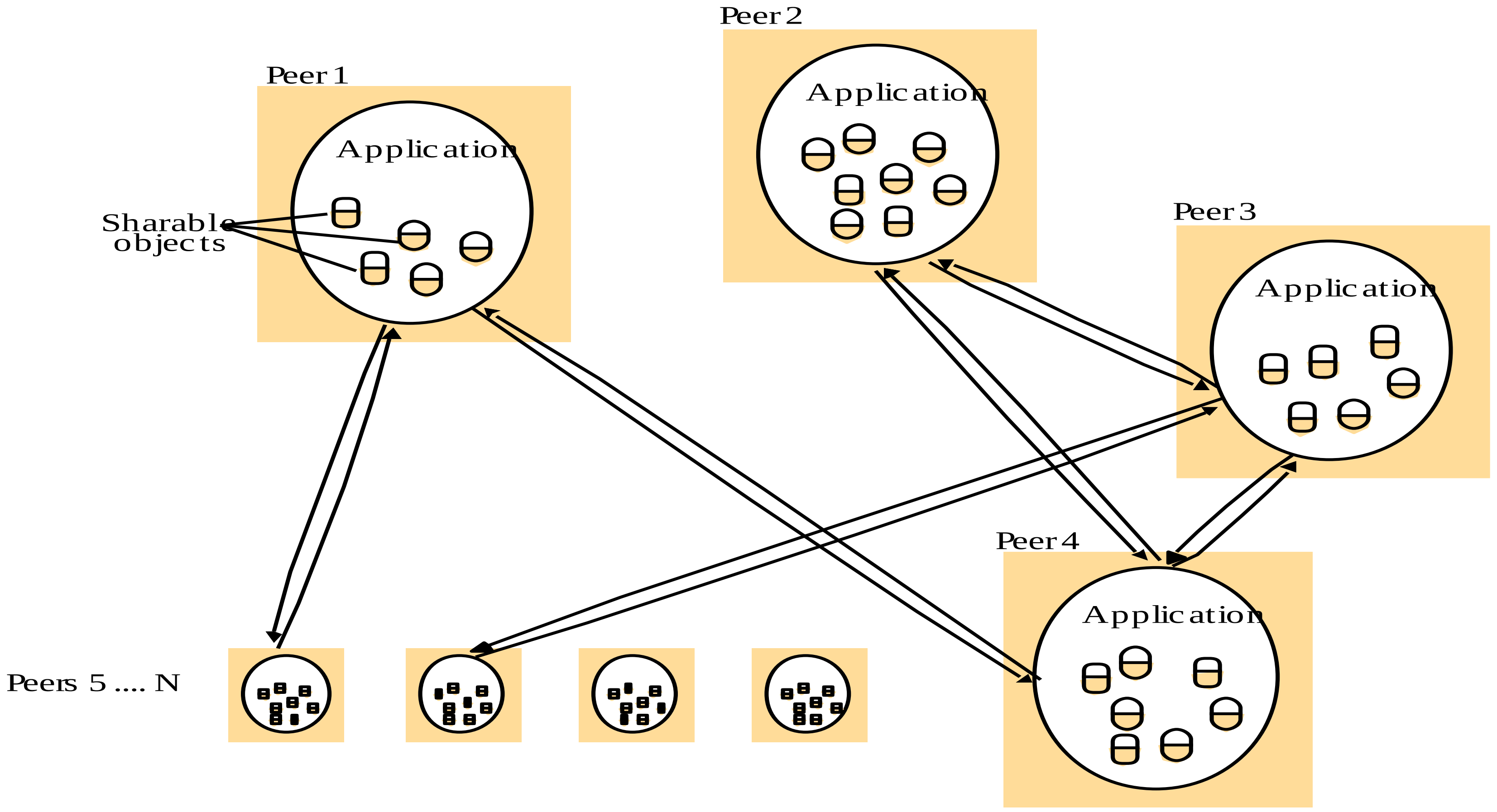


Figure 5. A distributed application based on the peer-to-peer architecture

Variants of Client Server Model

- The problem of client-server model is placing a service in a server at a single address that does not scale well beyond the capacity of computer host and bandwidth of network connections.
- To address this problem, several variations of client-server model have been proposed.
- Some of these variations are discussed in the next slide.

Variants of Client Server Model

- 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.
  - The servers may partition the set of objects on which the service is based and distribute those objects between themselves, or they may maintain replicated copies of them on several hosts.
  - E.g. cluster that can be used for search engines.

(Figure 6)

Variants of Client Server Model

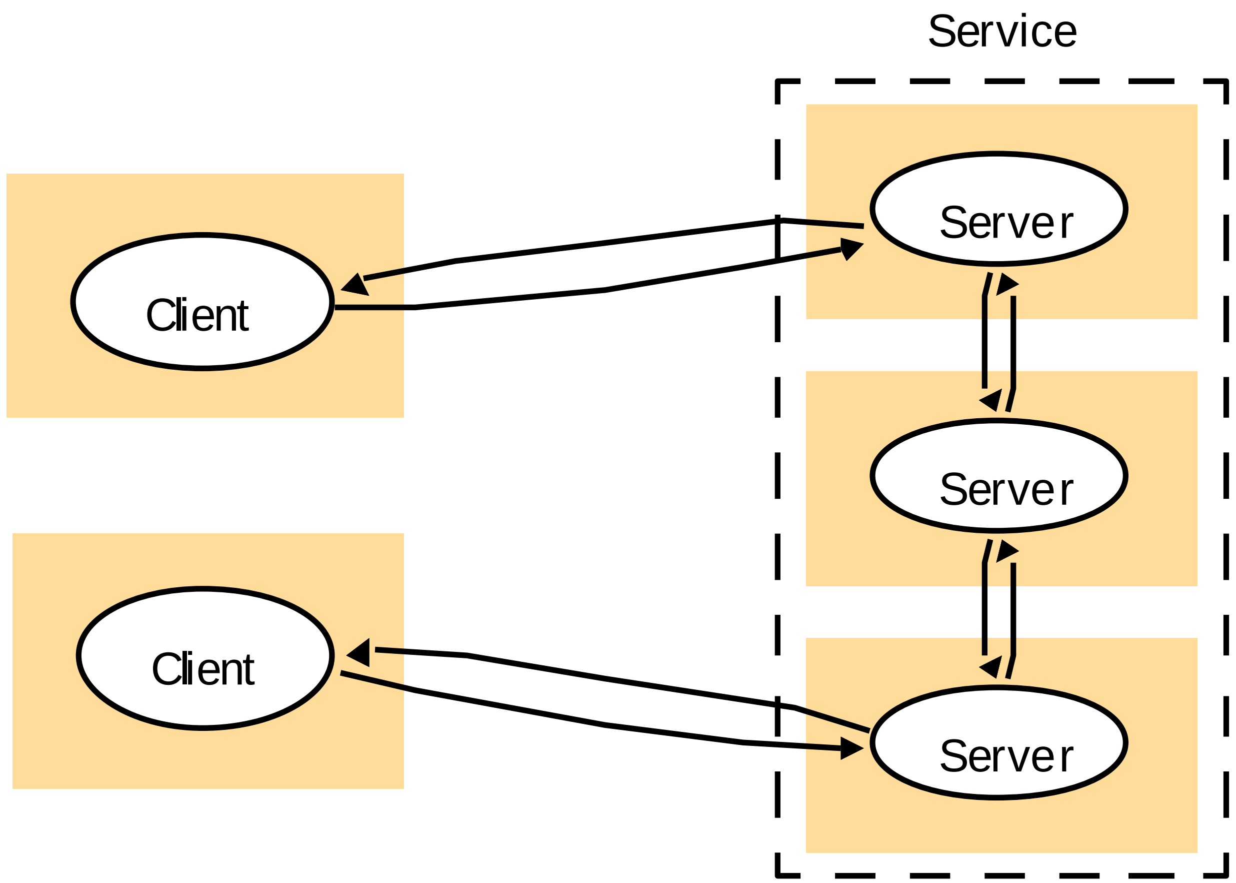


Figure 6. A service provided by multiple servers.

Variants of Client Server Model

- Proxy servers and caches
  - A cache is a store of recently used data objects.
  - When a new object is received at a computer it is added to the cache store, replacing some existing objects if necessary.
  - When an object is needed by a client process the caching service first checks the cache and supplies the object from there if an up-to-date copy is available.
  - If not, an up-to-data copy is fetched.

Variants of Client Server Model

- Caches may be co-located with each client or they may be located in a proxy server that can be shared by several clients.

(Figure 7)

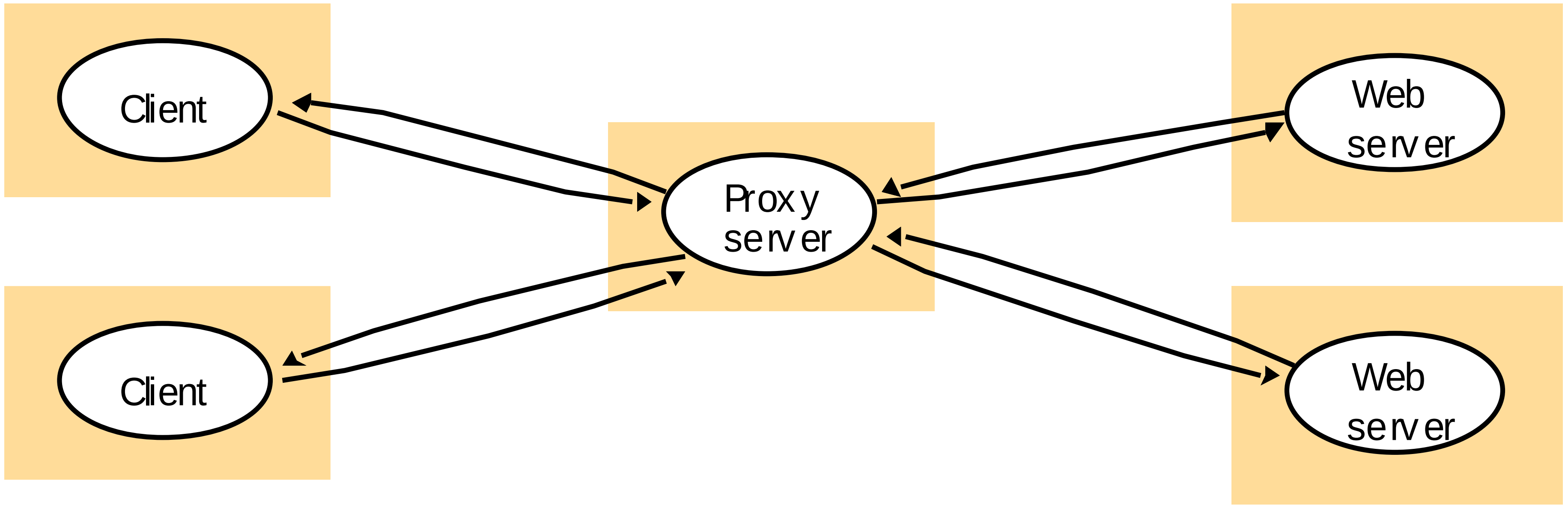


Figure 7. Web proxy server

Variants of Client Server Model

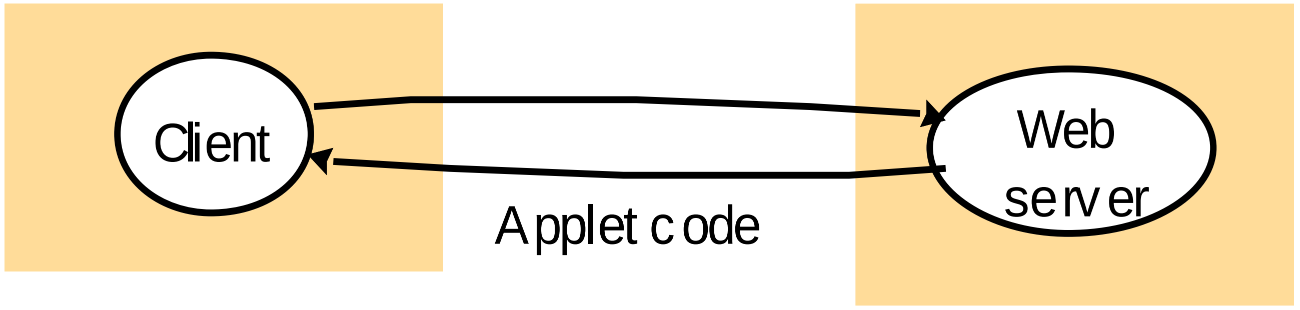
- **Mobile code:** the code that can be sent from one computer to another and run at the destination.
  - ➢ Applets are a well-known and widely used example of mobile code.

    the user running a browser selects a link to an applet whose code is stored on a web server; the code is downloaded to the browser and runs there
  - ➢ Applets downloaded to clients give good interactive response
  - ➢ Mobile codes such as Applets are a potential security threat to the local resources in the destination computer.
  - ➢ Browsers give applets limited access to local resources. For example, by providing no access to local user file system.

(Figure 8)

Variants of Client Server Model

a) client request results in the downloading of applet code



b) client interacts with the applet

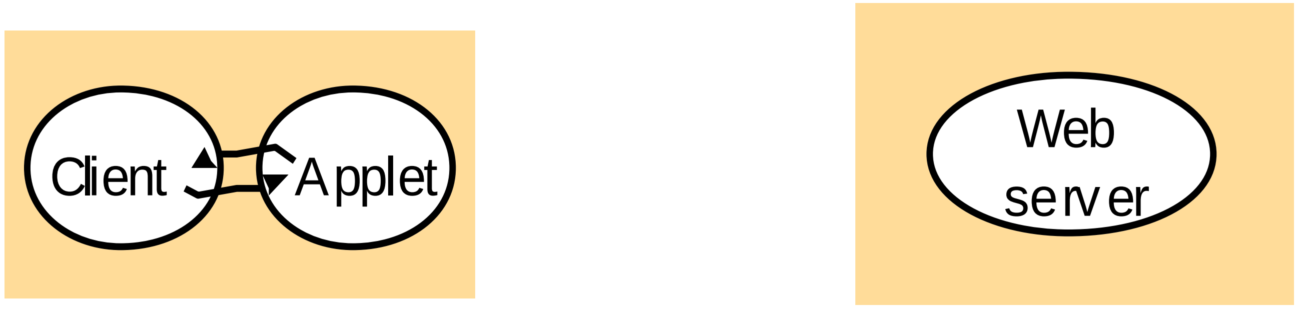


Figure 8. Web applets

Variants of Client Server Model

- Mobile agents
  - A running program (code and data) that travels from one computer to another in a network carrying out a task, usually on behalf of some other process.
  - Examples of the tasks that can be done by mobile agents are:
    - To collecting information.
    - To install and maintain software on the computers within an organization.
    - To compare the prices of products from a number of vendors.

Variants of Client Server Model

- Mobile agents are a potential security threat to the resources in computers that they visit.
- The environment receiving a mobile agent should decide on which of the local resources to be allowed to use, based on the identity of the user on whose behalf the agent is acting.
- Mobile agents themselves can be vulnerable
    - They may not be able to complete their task if they are refused access to the information they need.

Fundamental Models

- Introduction
- Interaction Model
- Failure Model
- Security Model

Fundamental Models-Intro

- Fundamental Models are concerned with a more formal description of the properties that are common in all of the architectural models.
- All architectural models are composed of processes that communicate with each other by sending messages over a computer networks.

Interaction Model

- Distributed systems are composed of many interacting processes.
    - Two significant factors affecting interacting processes in a distributed system are:
        - Communication performance is often a limiting characteristic.
        - It is impossible to maintain a single global notion of time.

Interaction Model-Communication Channels

- Performance of communication channels
  - Communication over a computer network has the performance characteristics such as:
    - Latency: The delay between the start of a message’s transmission from one process to the beginning of its receipt by another.
    - Bandwidth: The total amount of information that can be transmitted over a computer network in a given time.
      - Communication channels using the same network, have to share the available bandwidth.
    - Jitter: The variation in the time taken to deliver a series of messages.
      - It is relevant to multimedia data.
        - For example, if consecutive samples of audio data are played with differing time intervals then the sound will be badly distorted.

Interaction Model-Computer Clock

- 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.

Interaction Model-Computer Clock

- This is because computer clock drift from perfect time and their drift rates differ from one another.
- Clock drift rate refers to the relative amount that a computer clock differs from a perfect reference clock.
- Even if the clocks on all the computers in a distributed system are set to the same time initially, their clocks would eventually vary quite significantly unless corrections are applied.

Interaction Model-Variations

- Two variants of the interaction model
  - In a distributed system it is hard to set time limits on the time taken for process execution, message delivery or clock drift.
  - Two models of time assumption in distributed systems are:
    - 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.

Interaction Model-Variations

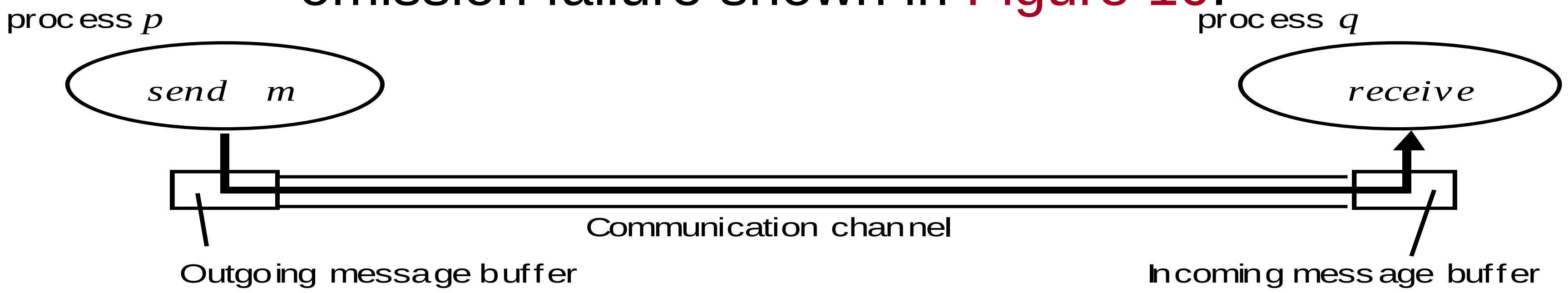
- ❖ Asynchronous distributed system
  - It has no assumption about time.
  - There is no bound on process execution speeds.
    - ❑Each step may take an arbitrary long time.
  - There is no bound on message transmission delays.
    - ❑A message may be received after an arbitrary long time.
  - There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

Failure Model

- In a distributed system both processes and communication channels may fail – That is, they may depart from what is considered to be correct or desirable behavior.
- Types of failures:
    - Omission Failures
    - Arbitrary Failures
    - Timing Failures

Failure Model

- Omission failure
  - Omission failures refer to cases when a process or communication channel fails to perform actions that it is supposed to do.
  - The chief omission failure of a process is to crash. In case of the crash, the process has halted and will not execute any further steps of its program.
  - Another type of omission failure is related to the communication which is called communication omission failure shown in Figure 10.



Failure Model

- The communication channel produces an omission failure if it does not transport a message from “p”’s outgoing message buffer to “q”’s incoming message buffer.
- This is known as “dropping messages” and is generally caused by lack of buffer space at the receiver or at an intervening gateway or by a network transmission error, detected by a checksum carried with the message data.

Failure Model

- Arbitrary failure
  - Arbitrary failure is used to describe the worst possible failure semantics, in which any type of error may occur.
    - E.g. a process may set a wrong values in its data items, or it may return a wrong value in response to an invocation.
  - Communication channel can suffer from arbitrary failures.
    - E.g. message contents may be corrupted or real messages may be delivered more than once.

Failure Model

- Timing failure
  - Timing failures are applicable in synchronized distributed systems where time limits are set on process execution time, message delivery time and clock drift rate.

Failure Model

- Masking failure
  - It is possible to construct reliable services from components that exhibit failure.
    - E.g. multiple servers that hold replicas of data can continue to provide a service when one of them crashes.
  - A service masks a failure, either by hiding it altogether or by converting it into a more acceptable type of failure.
    - E.g. checksums are used to mask corrupted messages- effectively converting an arbitrary failure into an omission failure.

Security Model

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

Security Model

- Protecting Objects
  - Access rights
    - Access rights specify who is allowed to perform the operations on a object.
      - Who is allowed to read or write its state.
  - Principal
    - Principal is the authority associated with each invocation and each result.
    - A principal may be a user or a process.
    - The invocation comes from a user and the result from a server.

Security Model

- The sever is responsible for
  - Verifying the identity of the principal (user) behind each invocation.
  - Checking that they have sufficient access rights to perform the requested operation on the particular object invoked.
  - Rejecting those that do not.

(Figure 11)

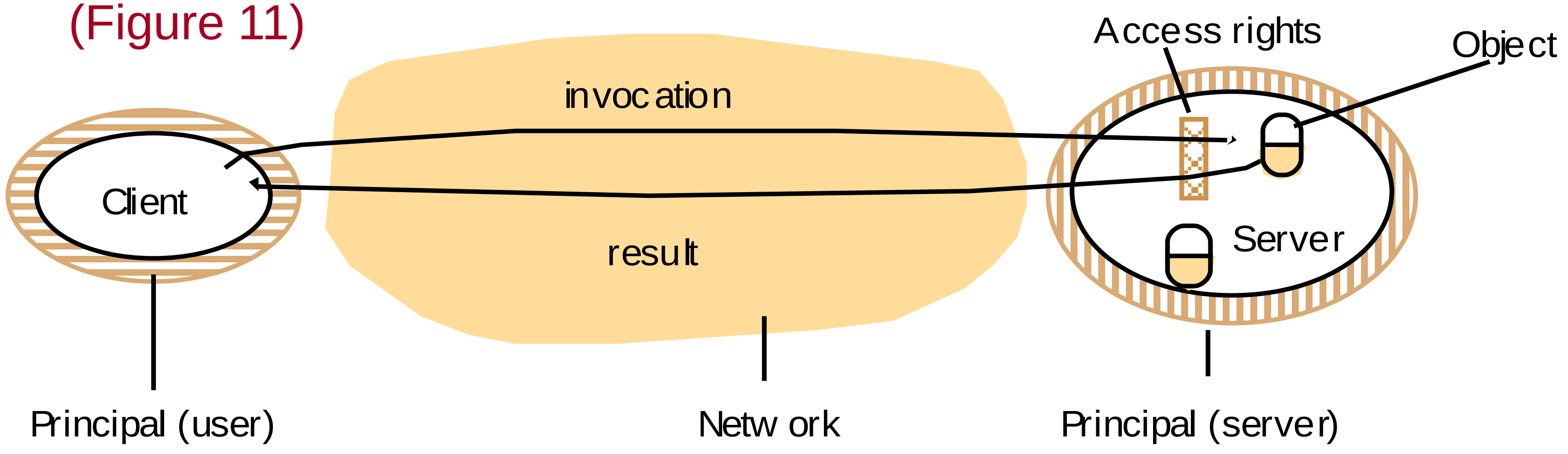


Figure 11. Objects and principals.

Security Model

- The enemy
  - To model security threats, we assume an enemy that is capable of sending any message to any process and reading or copying any message between a pair of processes.

(Figure 12)

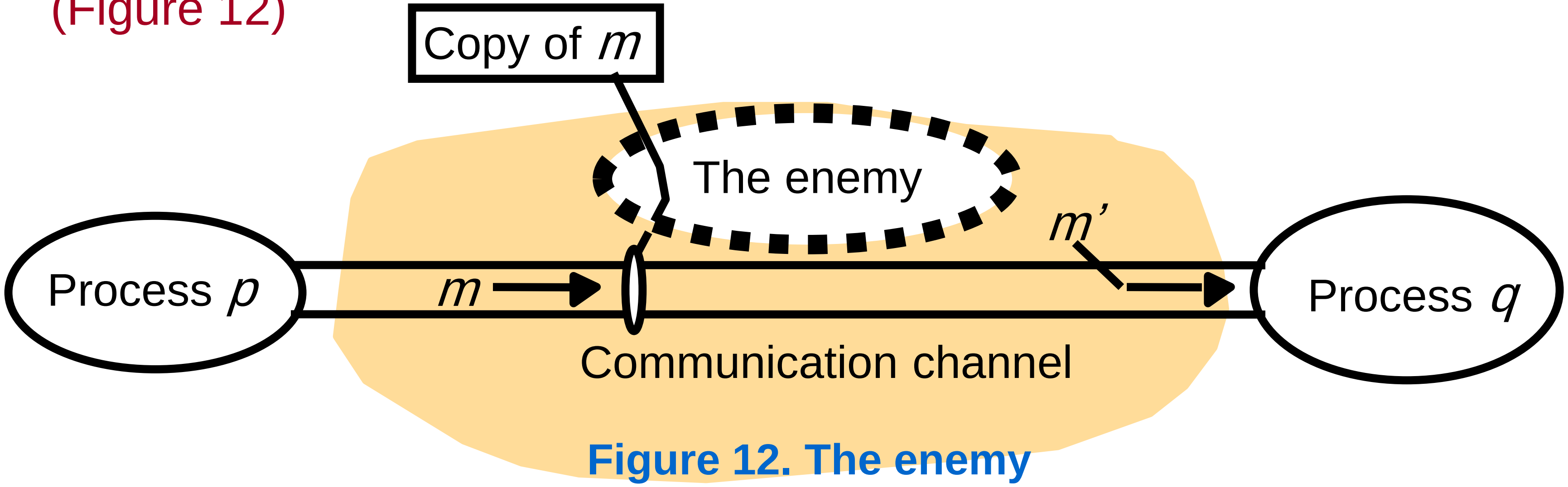


Figure 12. The enemy

Security Model

- Threats from a potential enemy are classified as:
  - Threats to processes (email), lack of knowledge of true source of a message
    - problem both to server and client side
    - example: spoofing a mail server
  - Threats to communication channels(email)
    - threat to the privacy and integrity of messages
    - can be defeated using secure channels

Security Model

- Defeating security threats
  - Secure systems are based on the following main techniques:
    - Cryptography and shared secrets
      - Cryptography is the science of keeping message secure.
      - Encryption is the process of scrambling a message in such a way as to hide its contents.
    - Authentication
      - The use of shared secrets and encryption provides the basis for the authentication of messages.