System Models

Y. V. Lokeswari AP / CSE

Reference: George Coulouris, Jean Dollimore and Tim Kindberg, “Distributed Systems Concepts and Design”, Fifth Edition, Pearson Education, 2012

System Models

- Physical Model
- Architectural Model
- Fundamental Model

System Models

1. **Physical Model**

   Describes system as set of computers and their interconnecting network.

2. **Architectural Model**

   Describes system in terms of computational task performed by its computational elements.

3. **Fundamental Model**

   1. **Interaction Model**

      Sequencing communication between elements of systems

   2. **Failure Model**

      Ways in which system may fail to operate correctly

   3. **Security Model**

      How to protect system against attempt to interfere with correct operation or steal its data.

Physical Model

1. **Physical Model**

   A representation of the underlying hardware elements of a distributed system that abstracts away from specific details of the computer and networking technologies.

1. **Early Distributed Systems**
   - 10 and 100 nodes interconnected by a local area network, with limited Internet connectivity, share local printers and file servers

2. **Internet-scale Distributed systems**
   - An extensible set of nodes interconnected by a network of networks (the Internet)
   - Involves only desktops and computers (Static)

3. **Contemporary Distributed systems**

   Nodes are laptops, smart phones (Nodes move from one location to another)

Architectural Model

2. Architectural Model

It describes computational and communication tasks.

1. Architectural Elements

- Communicating Entities
- Communicating paradigms
- Roles and Responsibilities
- Placement

2. Architectural Patterns

- Layered Architecture
- Tiered Architecture
- AJAX
- Thin Clients
- Proxy , Brokerage and Reflection

3. Middleware platform

- Categories of Middleware

Architectural Model

1. Architectural Elements

1. Communicating Entities

1. **Processes**
2. **Nodes**
3. **Threads**
4. **Objects (Accessed through Interfaces)**
5. **Components (Accessed through Interfaces and assumptions made on other components to fulfill its function).**
6. **Web services (S/W application that interacts with other software agents using XML messages and Internet protocols)**

2. Communicating Paradigm

1. **Interprocess communication**
2. **Remote Invocation**
    1. **Request-Reply Protocols**
    2. **Remote Procedure calls (RPC)**
    3. **Remote Method Invocation (RMI)**
3. **Indirect Communication (Space and Time Uncoupling)**
    1. **Group Communication (One – to –many, Each group is addressed by a Group Identifier)**
    2. **Publish-Subscribe Systems (Producers distribute information to large number of similar consumers)**
    3. **Message Queues (Producer process send messages to Queue and consumer process access messages from queues)**
    4. **Tuple Spaces (Processes place arbitrary items (tuples) in tuple space and other processes can read or remove tuples)**
    5. **Distributed Shared Memory (Abstraction for sharing data between processes)**

Architectural Model

1. Architectural Elements

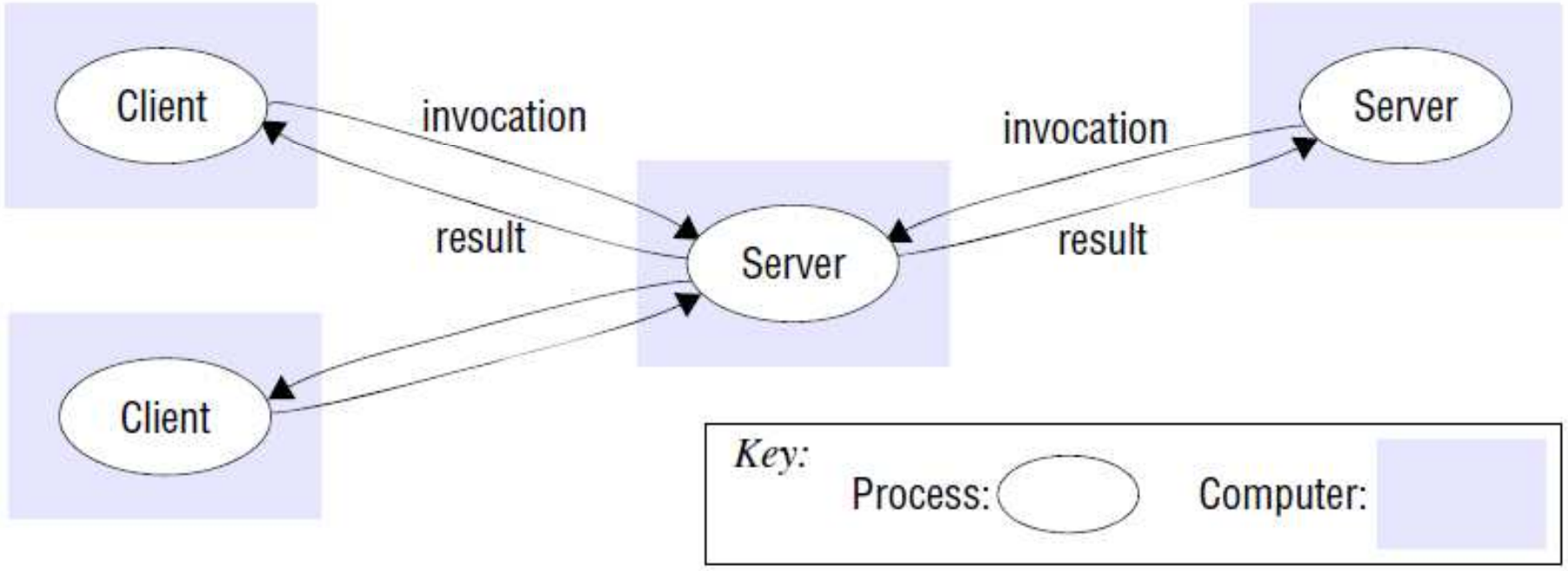
Communicating entities (what is communicating)		*Communication paradigms (how they communicate)*		
System-oriented entities	*Problem-oriented entities*	*Interprocess communication*	*Remote invocation*	*Indirect communication*
Nodes	Objects	Message passing	Request-reply	Group communication
Processes	Components	Sockets	RPC	Publish-subscribe
	Web services	Multicast	RMI	Message queues
				Tuple spaces
				DSM

Architectural Model

1. Architectural Elements

3. Roles and Responsibilities

1. **Process takes roles as either client or server**
2. **Servers may in turn be clients of other servers.**

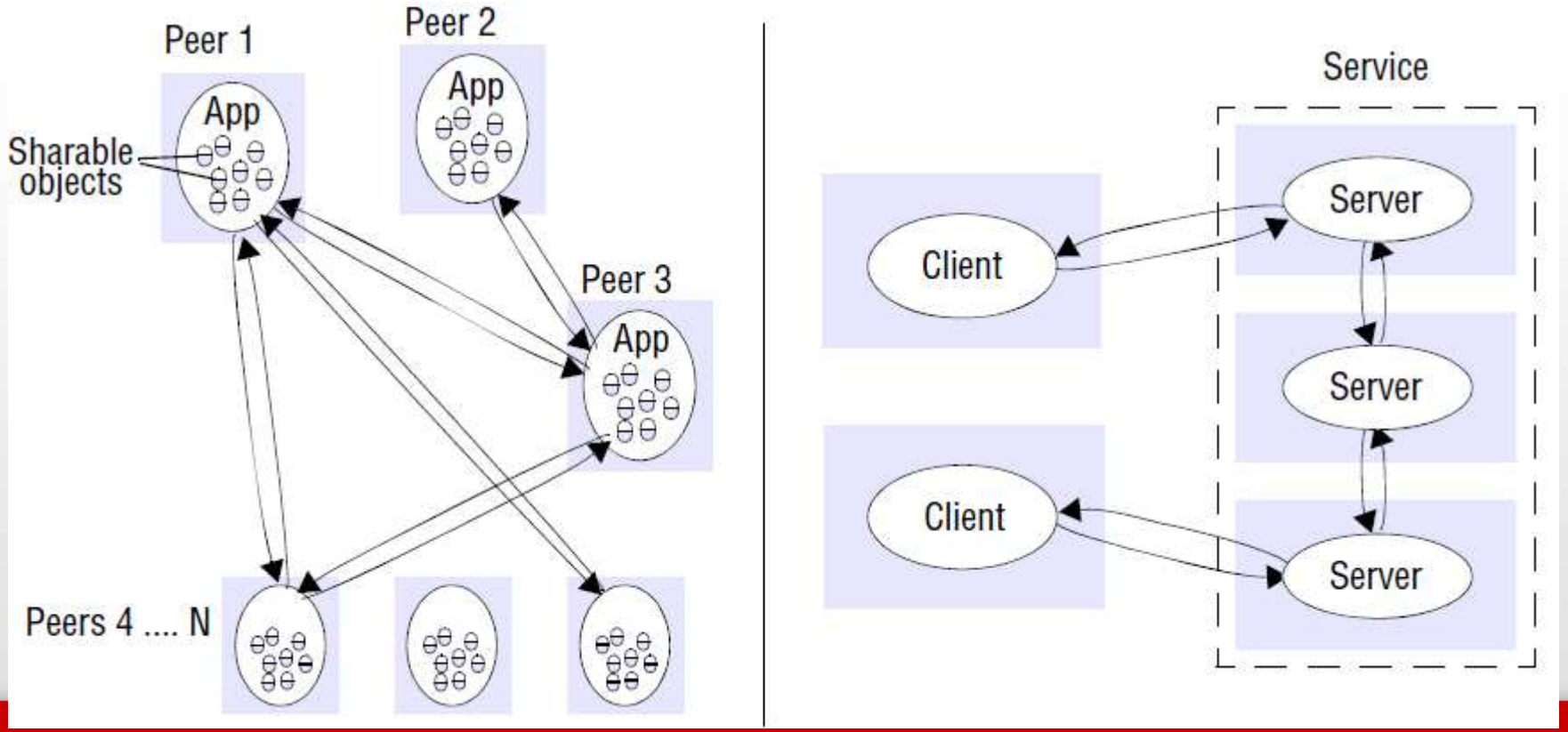


Architectural Model

1. Architectural Elements

3. Roles and Responsibilities

1. **Peer – to –peers : All processes involved in task , interact cooperatively as peers.**
2. **Service provided by multiple servers**



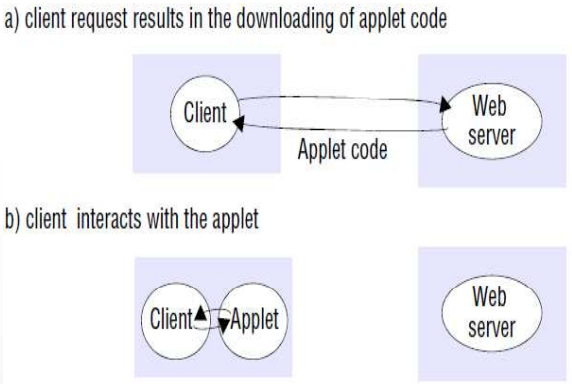
Architectural Model

1. Architectural Elements

4. Placement

How entities (objects, services) map on the underlying physical distributed infrastructure.

- Placement needs to take into account the following
- The patterns of communication between entities,
- The reliability of given machines and their current loading,
- The quality of communication between different machines



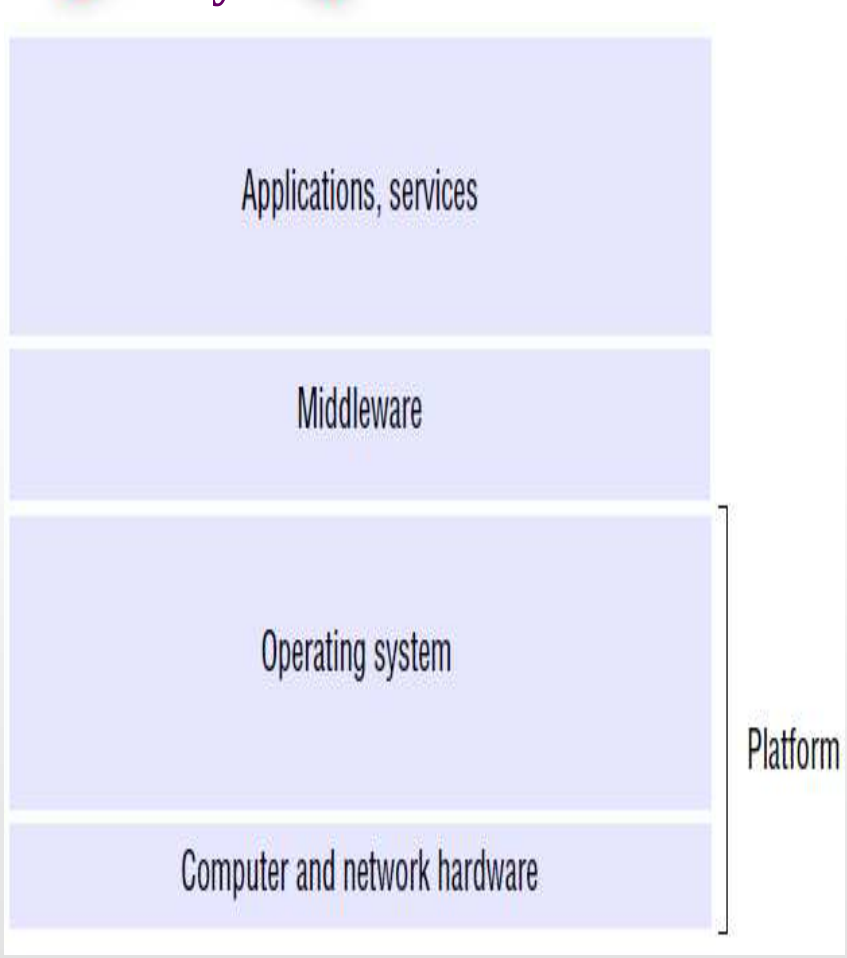
Issues in Placement:

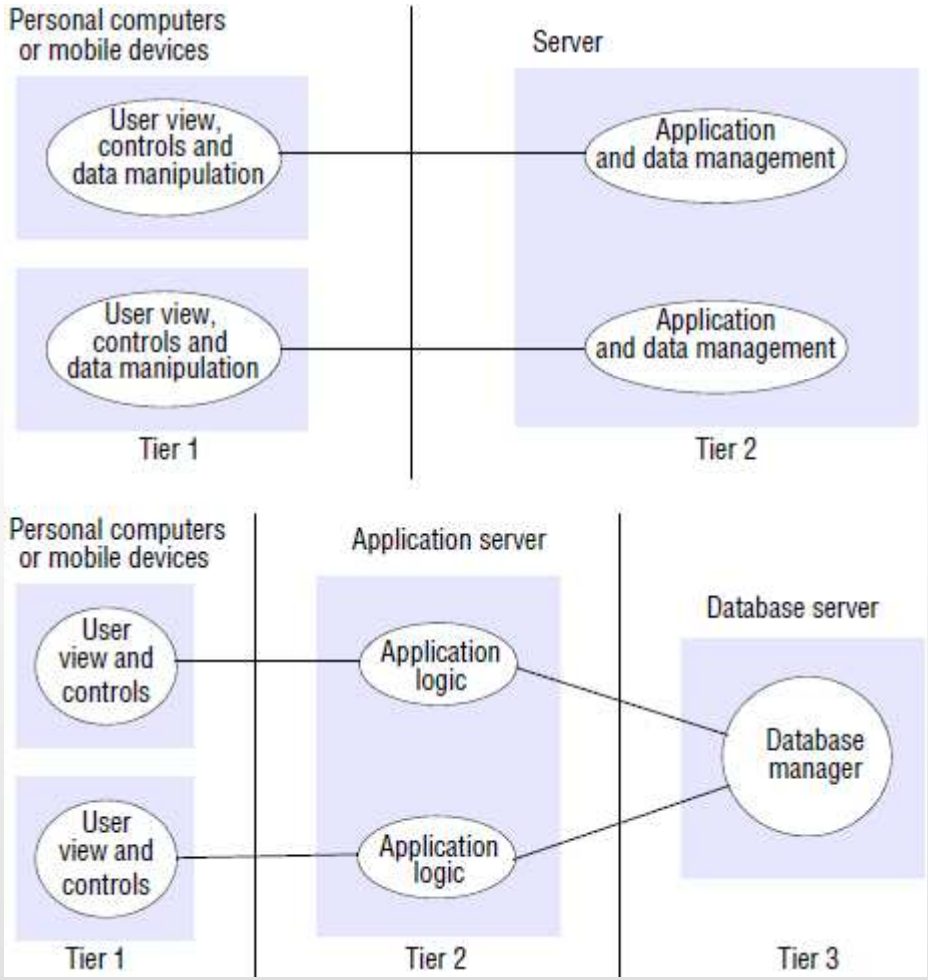
- Mapping of services to multiple servers
  - Services may be implemented as several server processes in separate host computers interacting as necessary to provide a service to client processes
- Caching
  - cache is a store of recently used data objects that is closer to one client or a particular set of clients
- Mobile code
  - Browser downloads code from Web server and executes it (Eg: Applets)
- Mobile agents
  - a running program (including both code and data) that travels from one computer to another in a network carrying out a task on someone's behalf

Architectural Model

2. Architectural Patterns

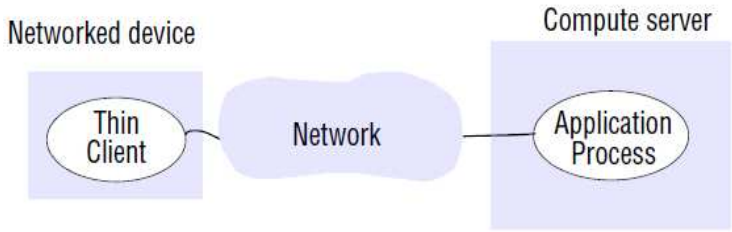
1. Layered Architecture Vs Tiered Architecture





Architectural Model

2. Architectural Patterns



Java Script: It allows UI and application logic to be programmed and executed in a browser window

AJAX

Enable major interactive web applications to be developed and deployed. It allows current page to be displayed with new values of application.

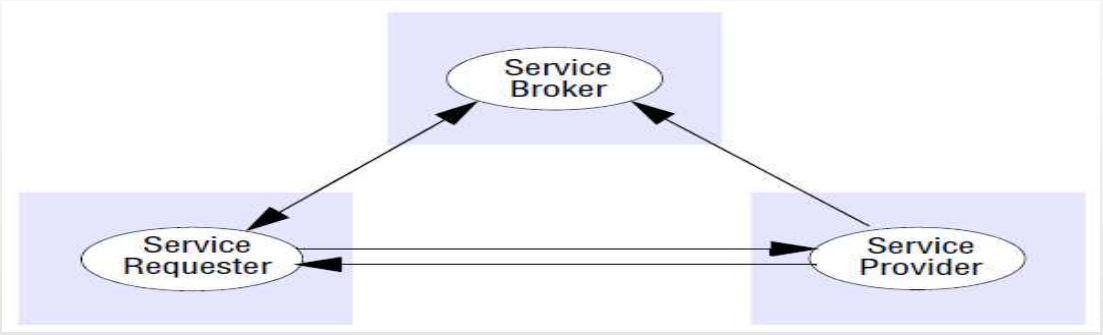
Thin Clients

It depends heavily on another computer (server) to fulfill its computational roles

Proxy

Designed to support location transparency in Remote procedure calls.

Web Services



Reflections:

Introspection – Dynamic discovery of properties of system

Intercession - Ability to dynamically modify structure or behavior.

Architectural Model

3. Middleware Solutions

Abstract over heterogeneity in the underlying infrastructure

Major categories:	*Subcategory*	*Example systems*
Distributed objects (Chapters 5, 8)	Standard	RM-ODP
	Platform	CORBA
	Platform	Java RMI
Distributed components (Chapter 8)	Lightweight components	Fractal
	Lightweight components	OpenCOM
	Application servers	SUN EJB
	Application servers	CORBA Component Model
	Application servers	JBoss
Publish-subscribe systems (Chapter 6)	-	CORBA Event Service
	-	Scribe
	-	JMS
Message queues (Chapter 6)	-	Websphere MQ
	-	JMS
Web services (Chapter 9)	Web services	Apache Axis
	Grid services	The Globus Toolkit
Peer-to-peer (Chapter 10)	Routing overlays	Pastry
	Routing overlays	Tapestry
	Application-specific	Squirrel
	Application-specific	OceanStore
	Application-specific	Ivy
	Application-specific	Gnutella

Fundamental Model

3. Fundamental Model

It describes about interaction, failure and security models

1. Interaction Model

- Multiple server processes may cooperate with one another to provide a service
- A set of peer processes may cooperate with one another to achieve a common goal.
- **Performance of communicating channels**
  - **Latency** (delay b/w start of message transmission and beginning of its receipt)
  - **Bandwidth** (Amount of information transmitted in given time)
  - **Jitter** (Variation in time to deliver consecutive audio video data)
- **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**.
  - Therefore two **processes running** on **different computers** can each associate **timestamps** with their **events**
  - The term **clock drift** rate refers to the **rate** at which a **computer clock deviates** from a perfect **reference clock.**

Fundamental Model

3. Fundamental Model

1. Interaction Model

Two variants of Interaction Model

Synchronous distributed systems (Bound on time)

- **Process execution speeds :** The time to execute each step of a process has known lower and upper bounds.
- **Message transmission delays :** Each message transmitted over a channel is received within a known bounded time.
- **Clock drift rates :** Each process has a local clock whose drift rate from real time has a known bound.

Asynchronous distributed systems (No Bound on time)

- **Process execution speeds** : may take in picoseconds or for long time
- **Message transmission delays** : Message delivered in negligible time or for long time.
- **Clock drift rates** : Drift of clock is arbitrary.

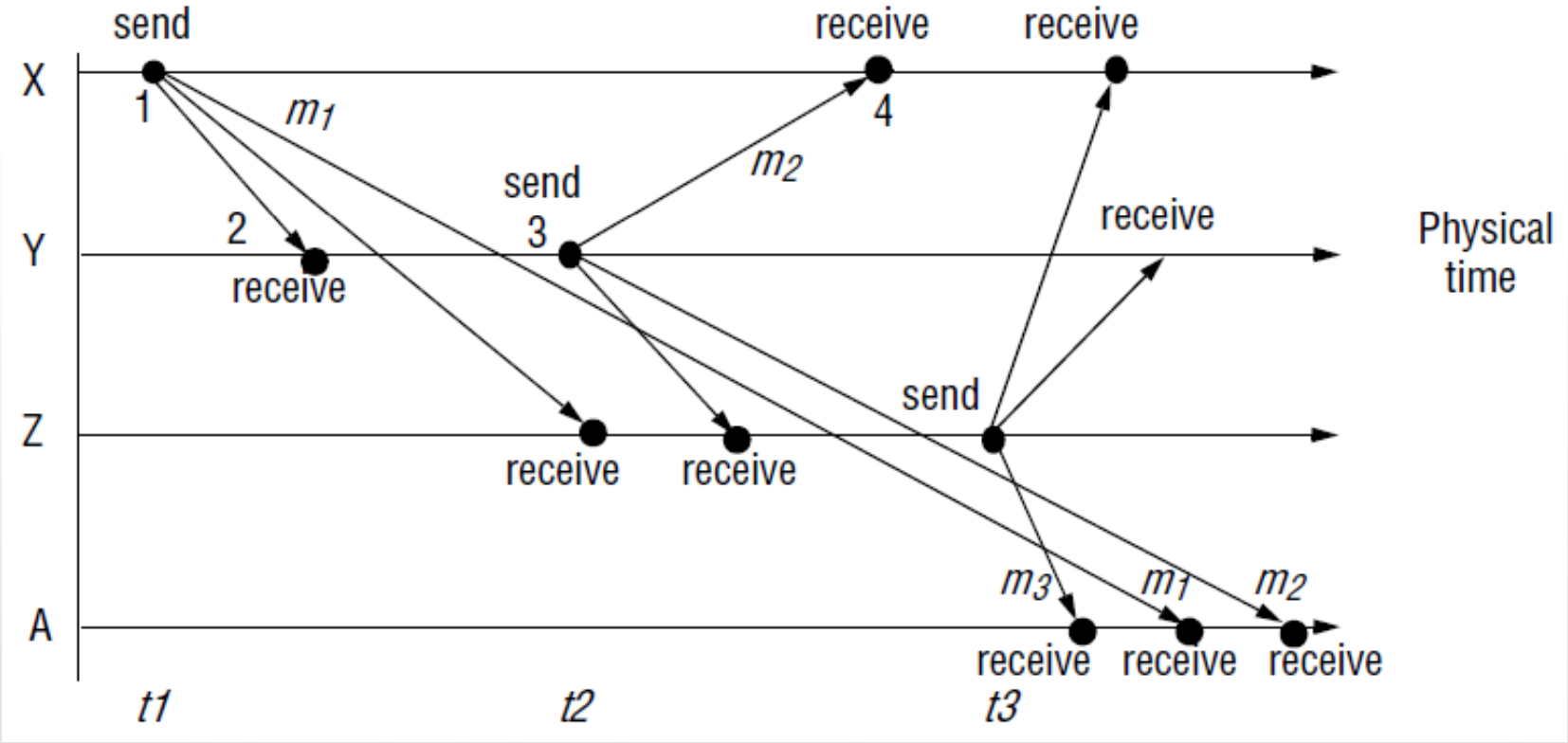
Fundamental Model

3. Fundamental Model

1. Interaction Model

Event Ordering

		Inbox:
Item	*From*	*Subject*
23	Z	Re: Meeting
24	X	Meeting
25	Y	Re: Meeting



X sends *m1 before Y receives m1;* *Y sends m2 before X receives m2* ; Y receives *m1 before sending m2*

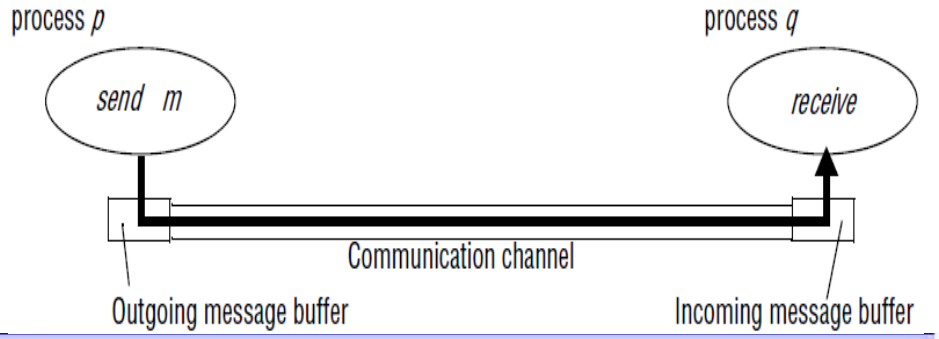
Fundamental Model

3. Fundamental Model

2. Failure Model

This defines the ways in which failure may occur.

Omission Failures 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* operation 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 or commit omissions; a process may stop or take an incorrect step.

Fundamental Model

3. Fundamental Model

2. Failure Model

Timing Failure

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 Failure

- Multiple servers that hold replicas of data
- Can enable a new service to be designed to mask the failure of the components.
- A service masks a failure either by hiding it altogether or by converting it into a more acceptable type of failure

Fundamental Model

3. Fundamental Model

2. Failure Model

Reliability of one-to-one communication

Reliable communication is defined as

- **Validity**: Any message in the outgoing message buffer is eventually delivered to the incoming message buffer.
- **Integrity**: The message received is identical to one sent, and no messages are delivered twice.

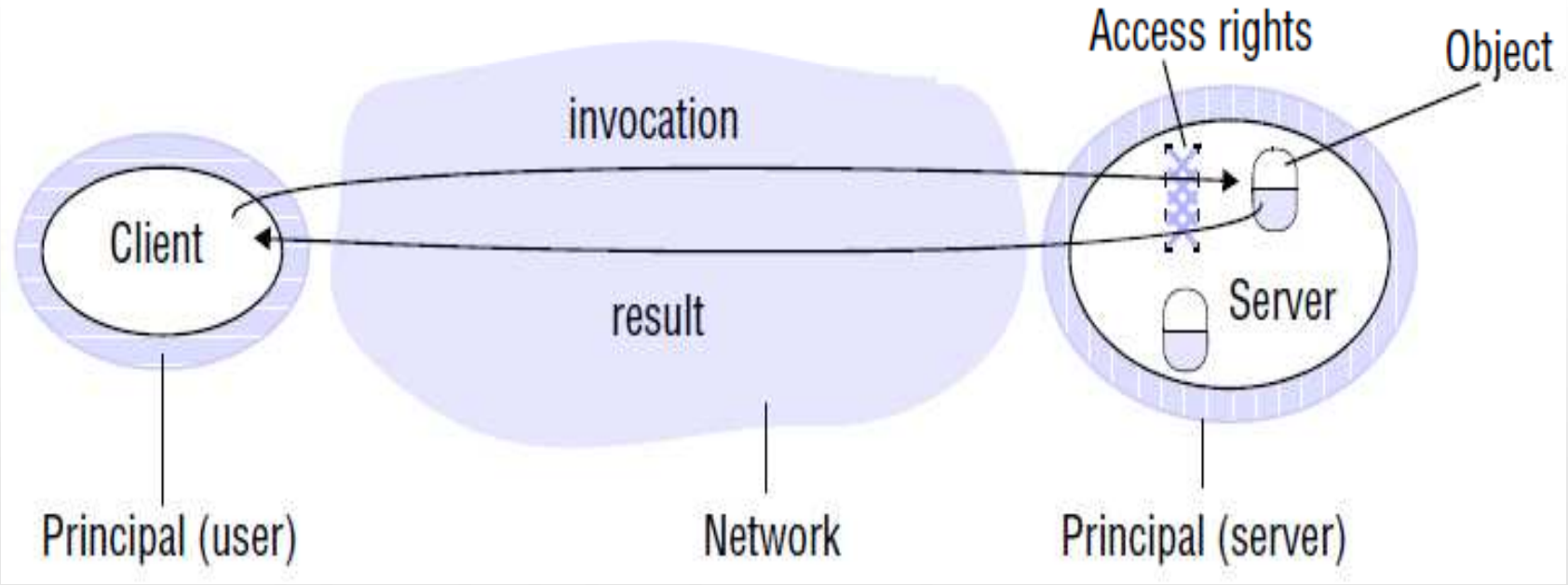
Fundamental Model

3. Fundamental Model

3. Security Model

- securing the processes and the channels
- Protecting the objects that they encapsulate against unauthorized access

Protecting objects

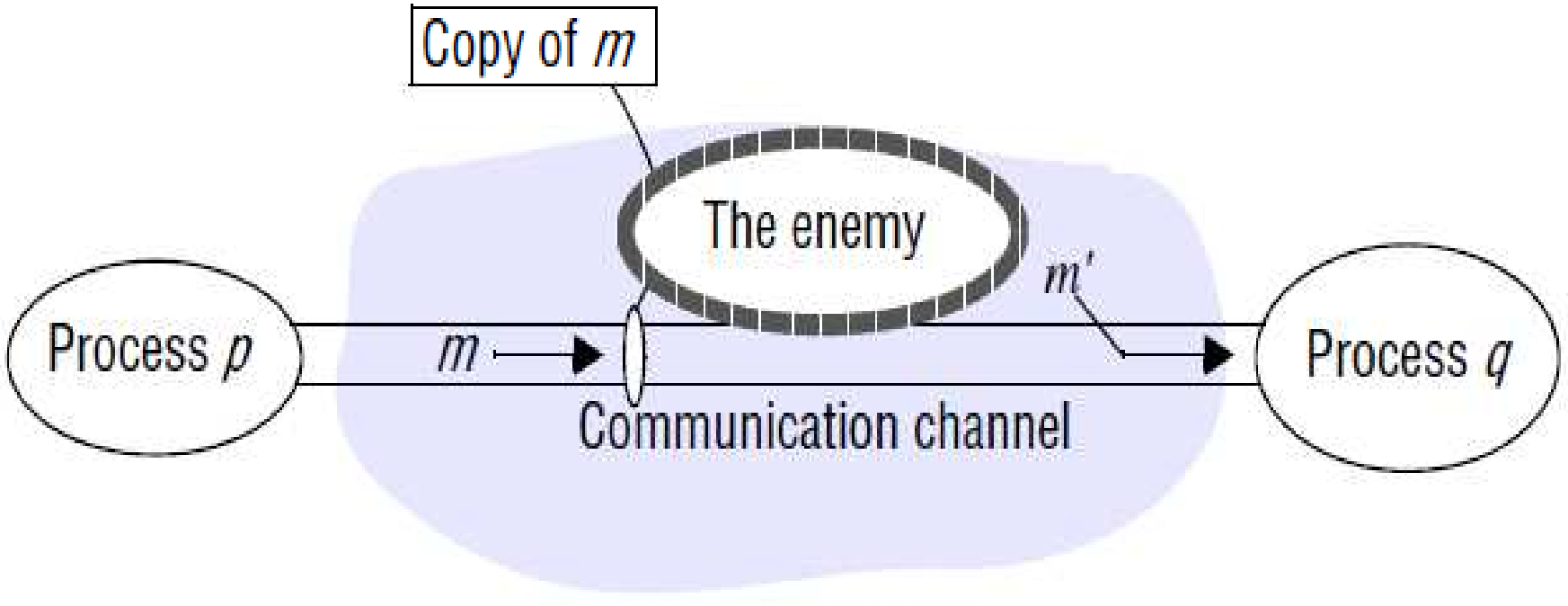


Fundamental Model

3. Fundamental Model

3. Security Model

Enemy : is capable of sending any message to any process and reading or copying any message sent between a pair of processes



Fundamental Model

3. Fundamental Model

3. Security Model

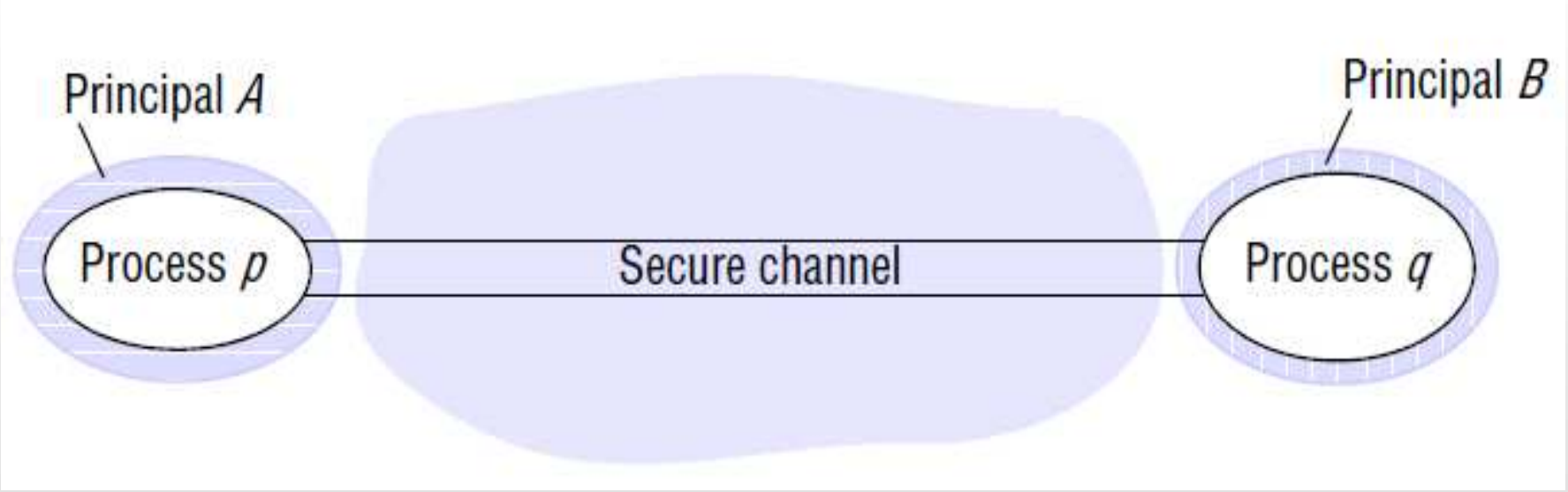
- **Enemy** : Causes threats to **Processes and Communication Channels**
- **Threats to Processes**
  - **Servers**
  - **Clients**
- **Threats to Communication Channels**
  - **Copy alter or inject messages that travel across network**
  - **Save copies of message and replay them at later time.**
- **Denial of Service**
- **Mobile code**

Fundamental Model

3. Fundamental Model

3. Security Model

- **Defeating Security Threats**
  - **Cryptography**
  - **Authentication**
  - **Secure Channels (Protocols like VPN and SSL)**



System Models

- Physical Model
- Architectural Model
- Fundamental Model

Thank You