Chapter 2: System Models

From **Coulouris, Dollimore, Kindberg and Blair**
Distributed Systems: Concepts and Design

Outlines

- Generations of Distributed Systems
- Physical Model
- Architectural Model
- Fundamental Model

Figure 2.1
Generations of distributed systems

Distributed systems:	*Early*	*Internet-scale*	*Contemporary*
Scale	Small	Large	Ultra-large
Heterogeneity	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
Openness	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
Quality of service	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

Physical Model

- The **physical model** is a representation of the underlying HW elements of a DS that abstracts away from specific details of the computer and networking technologies.

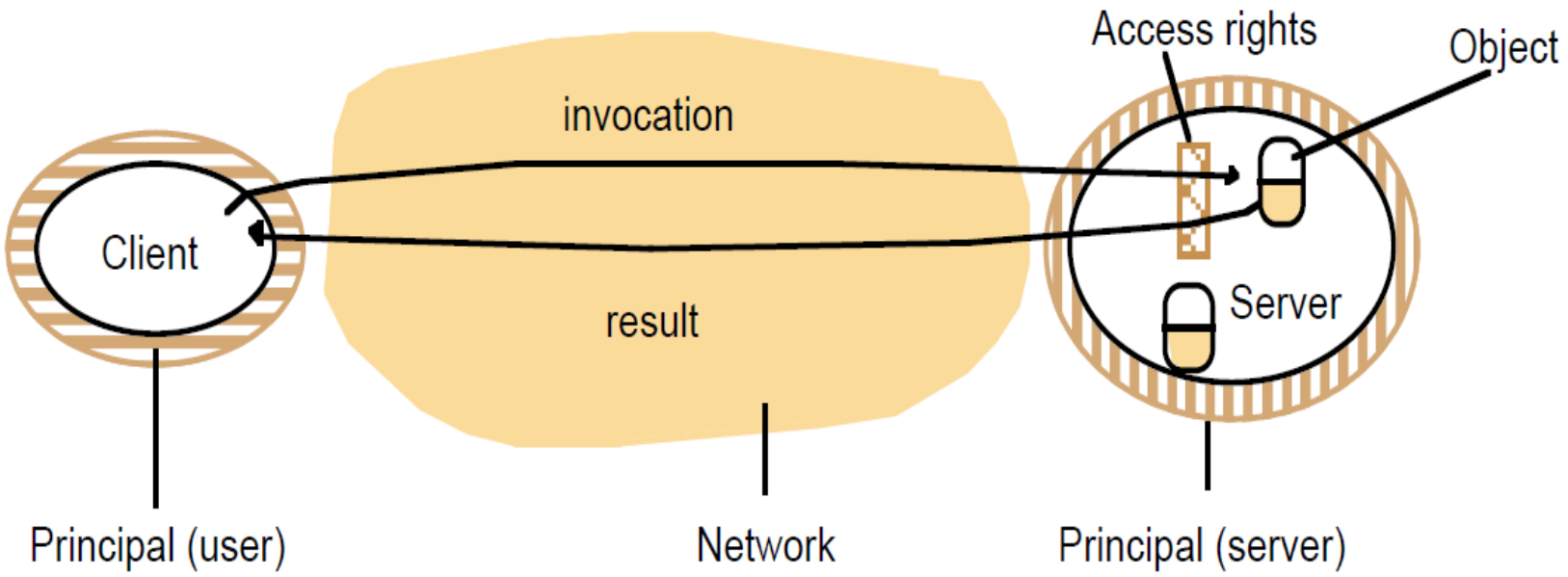
- The **DS** connects desktops, laptops, mobile computing devices, ubiquitous (everywhere) computing systems (including embedded systems), and cloud computing systems using different network topologies.

Architectural Model

- The **architectural model** describes the system in terms of computational and communication tasks performed by the computational elements and their interrelationship.
- **Communicating entities:**

1. **Processes.**
2. **Objects:** Uses object-oriented approach (encapsulation). A computation consists of a number of interacting objects representing units of decomposition of a given problem domain. Objects are accessed through interfaces. Interface Definition Language (IDL) is needed to provide specifications of each method definition of an object.
3. **Web services:** Closely related to objects. Web services are more integrated to WWW.

Figure 2.17
Objects and principals



Architectural Model

- **Communication paradigms (models):**

1. **Inter-process communication:** Massage-passing, sockets, and multicasting.
2. **Remote invocation (call):** Request-reply, Remote Procedure Call (RPC), Java Remote Method Invocation (RMI).
3. **Indirect Communication:** Group communication, Publish-subscribe, Message queues, Tuple spaces, Distributed Shared Memory (DSM).

Architectural Model

Publish-Subscribe Systems:

- One to many style.
- Receiver (Subscriber) declares interest in some event.
- Sender (Publisher) publish events of a given interest.
- Events are sent to matching subscribers.

Tuple Spaces:

- Sender places arbitrary data of interest called tuples in a persistent space.
- Receiver can manipulate the tuple by choosing to read it or even delete it.

Queue Systems:

- Point to point style.
- Sender sends messages to a queue.
- Receiver consumes messages from the queue.

Distributed Shared Memory (DSM):

- Sender and receivers can manipulate data in shared memory.
- Synchronization issues must be taken care of.

Figure 2.2
Communicating entities and communication paradigms

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

Roles and responsibilities:

1. **Client-server.**

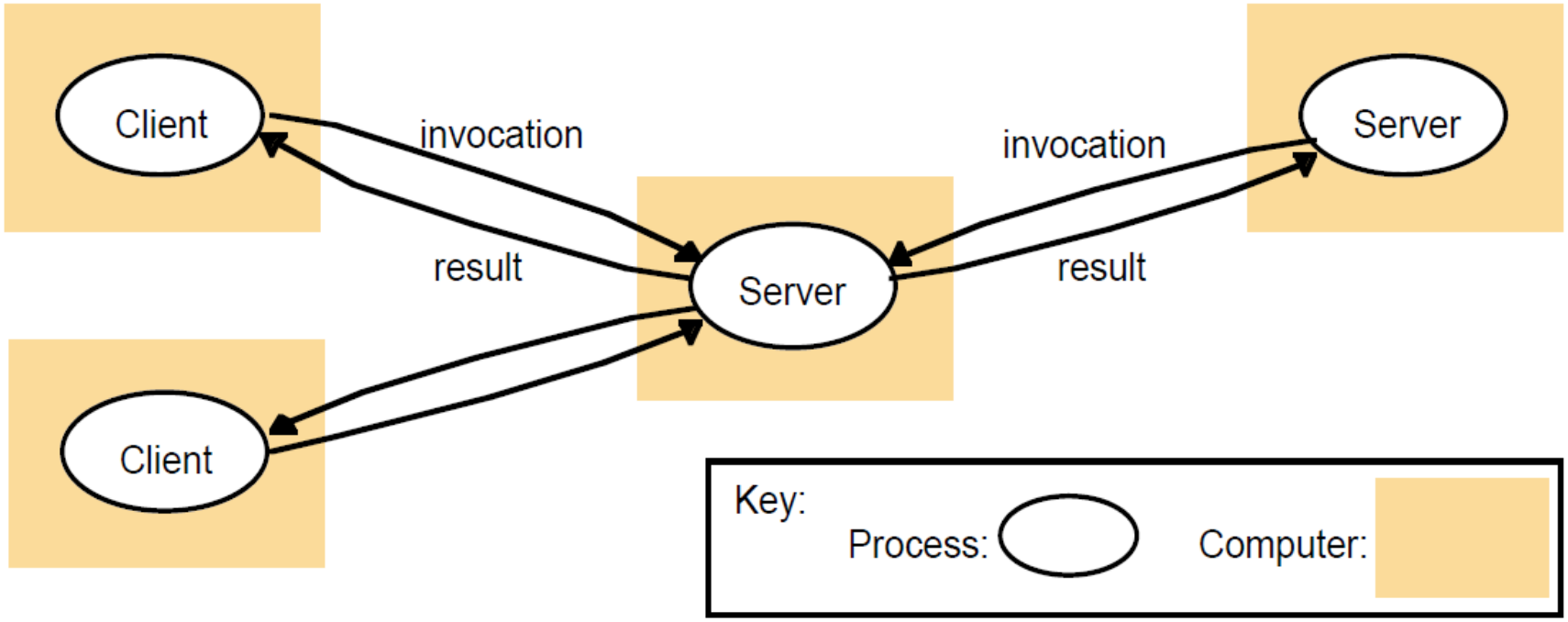
2. **Peer-to-peer.**

Architectural Model

1. **Client-server:** Processes take the role of being either a client or a server. In particular, client processes interact with individual server processes in potentially separate host computers in order to access the shared resources they manage. Servers may be clients to other servers **(Centralized resource sharing).**

- **Example-1: A web server** is a client of a local file server that manages the files in which the web pages are stored.
- **Example-2: Search engines** enable users to look up summaries of information available on web pages at sites throughout the Internet. These summaries are made by programs called **web crawlers**, which run in the background at a search engine site using **HTTP requests** to access web servers throughout the Internet. A **search engine** is 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 2.3
Clients invoke individual servers

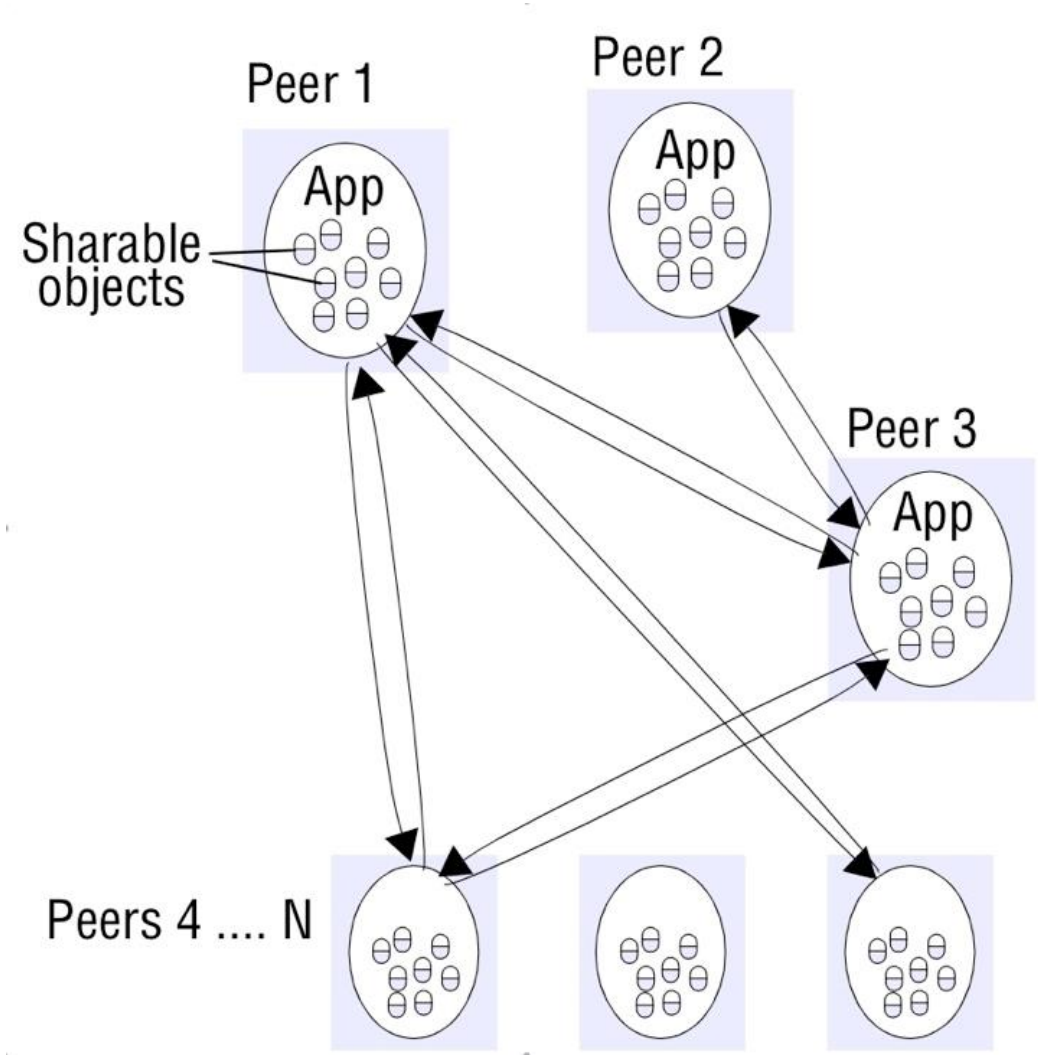


Architectural Model

2. **Peer-to-peer:** All processes involved in the task of being client or server. They play similar roles. P2P system can scale up easily **(Decentralized resource sharing).**

- All participating processes run the same program and offer the same set of interfaces to each other.
- **Example: Peer-to-peer application** where applications are composed of large numbers of peer processes running on separate computers and the pattern of communication between them depends entirely on application requirements. A large number of data objects are shared, an individual computer holds only a small part of the application database, and the storage, processing and communication loads for access to objects are distributed across many computers and network links (See Fig. 2.4a).

Figure 2.4a
Peer-to-peer architecture (Example: Peer-to-peer application)



Architectural Model

Placement:

Mapping services (objects) to the underlying physical DS.

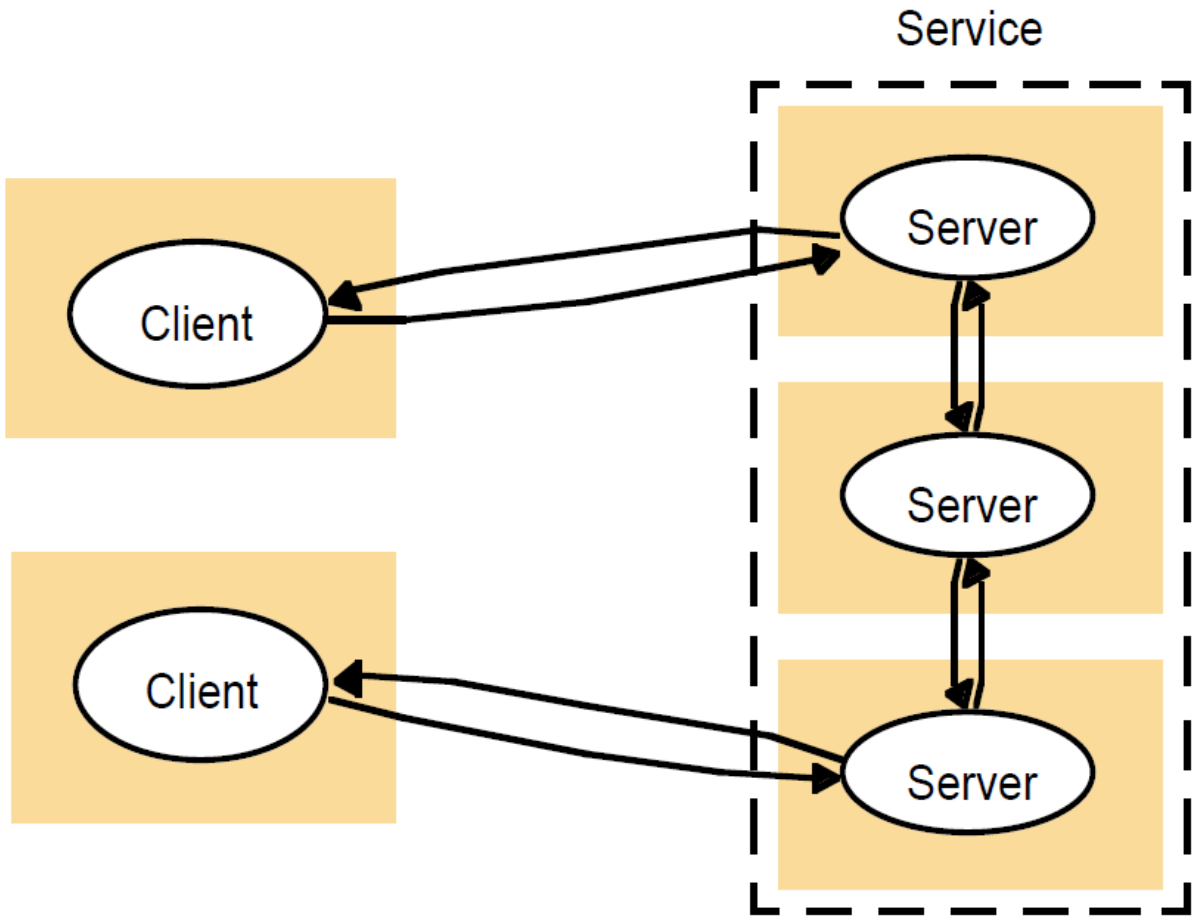
Placement Strategies:

1. **Mapping of services into multiple servers.**
2. **Caching.**
3. **Mobile code.**
4. **Mobile agents.**

Architectural Model

1. **Mapping of services into multiple servers** by partitioning set of services (objects) and distributing them among the servers or by replicating them among the servers (Fig. 2.4b).

Figure 2.4b
A service provided by multiple servers

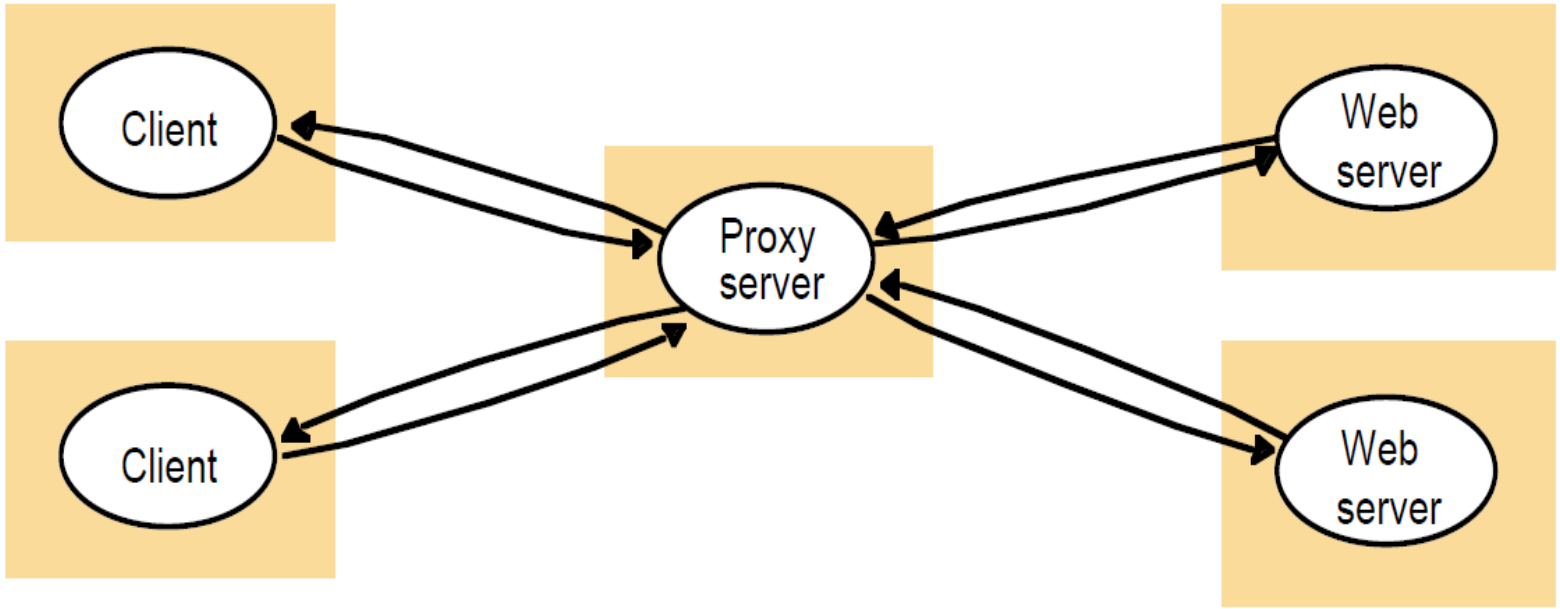


Architectural Model

2. **Caching:** Storing recently used data objects in a local cache (buffer).

- **Example:** Web proxy servers (Fig. 2.5) provide a shared cache of web resources for the client machines at a site or across several sites. The purpose of proxy servers is to increase the availability and performance of the service by reducing the load on the wide area network and web servers.

Figure 2.5
Web proxy server (Caching)



Architectural Model

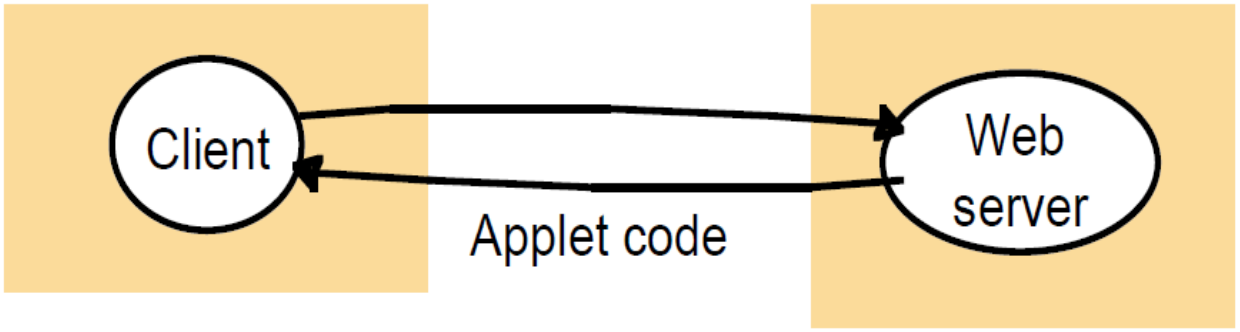
3. **Mobile code:** Example of mobile code is applets.

- 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 (See Fig. 2.6).

- An advantage of running the downloaded code locally is that it can give good interactive response since it does not suffer from the delays or variability of bandwidth associated with network communication.

Figure 2.6
Web applets

a) client request results in the downloading of applet code



b) client interacts with the applet

Architectural Model

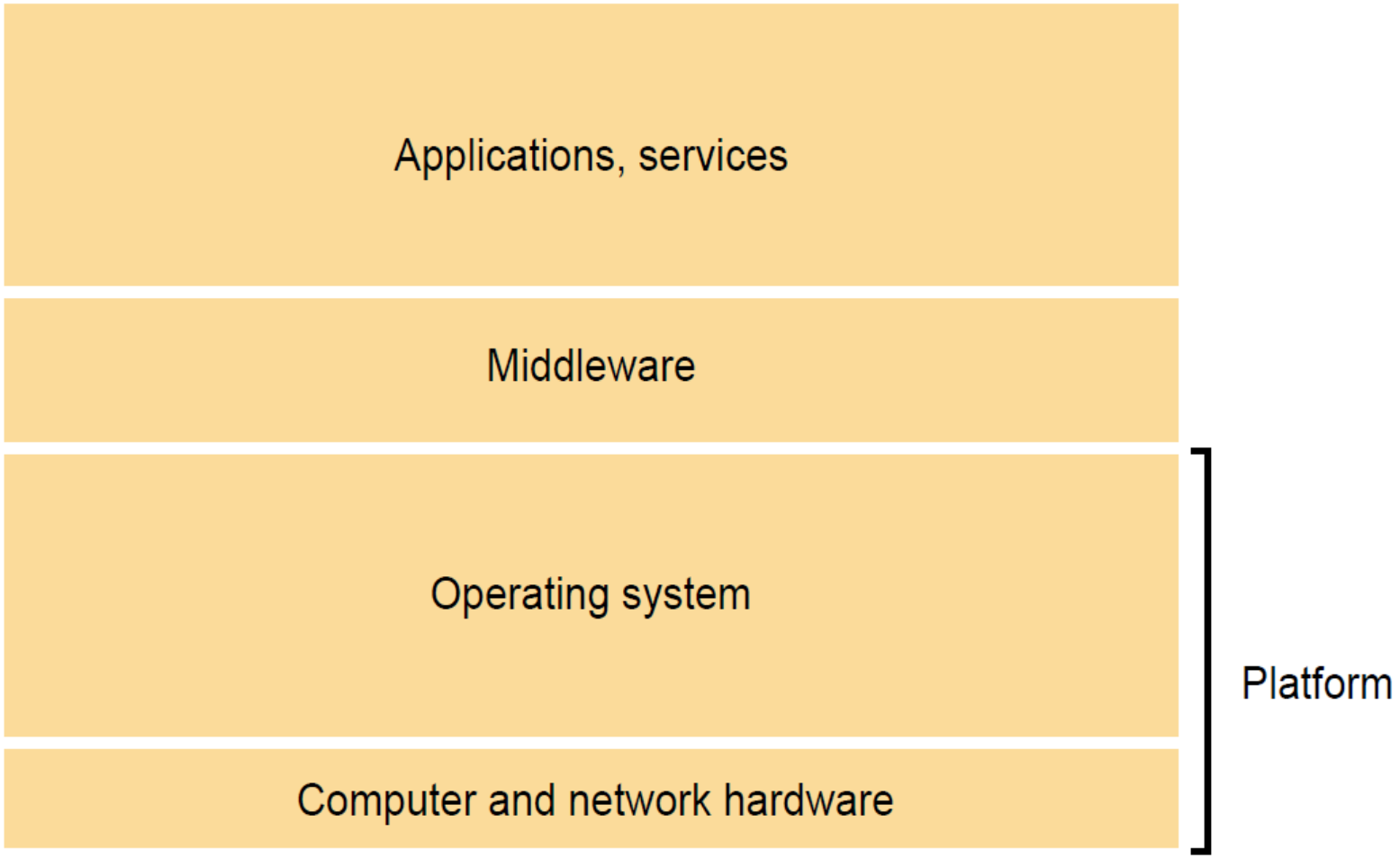
4. **Mobile agents:** A mobile agent is a running program (including both code and data) that travels from one computer to another in a network carrying out a task on someone's behalf, such as collecting information, and eventually returning with the results.

Architectural Patterns

Layering:

- Complex systems are partitioned into a number of layers, with a given layer making use of the services offered by the layers below. This abstraction makes a higher layer unaware of the lower layers' implementation details.
- Platform consists of the lowest level HW and OS layers. Middleware is used to mask heterogeneity (Fig. 2.7).

Figure 2.7
Software and hardware service layers in distributed systems



Architectural Patterns

Tiered Architecture:

- Organizing the functionalities of a given layer and placing each one into an appropriate server (physical node).
- It could be two-tiered, three-tiered, or n-tiered architecture (Fig. 2.8).

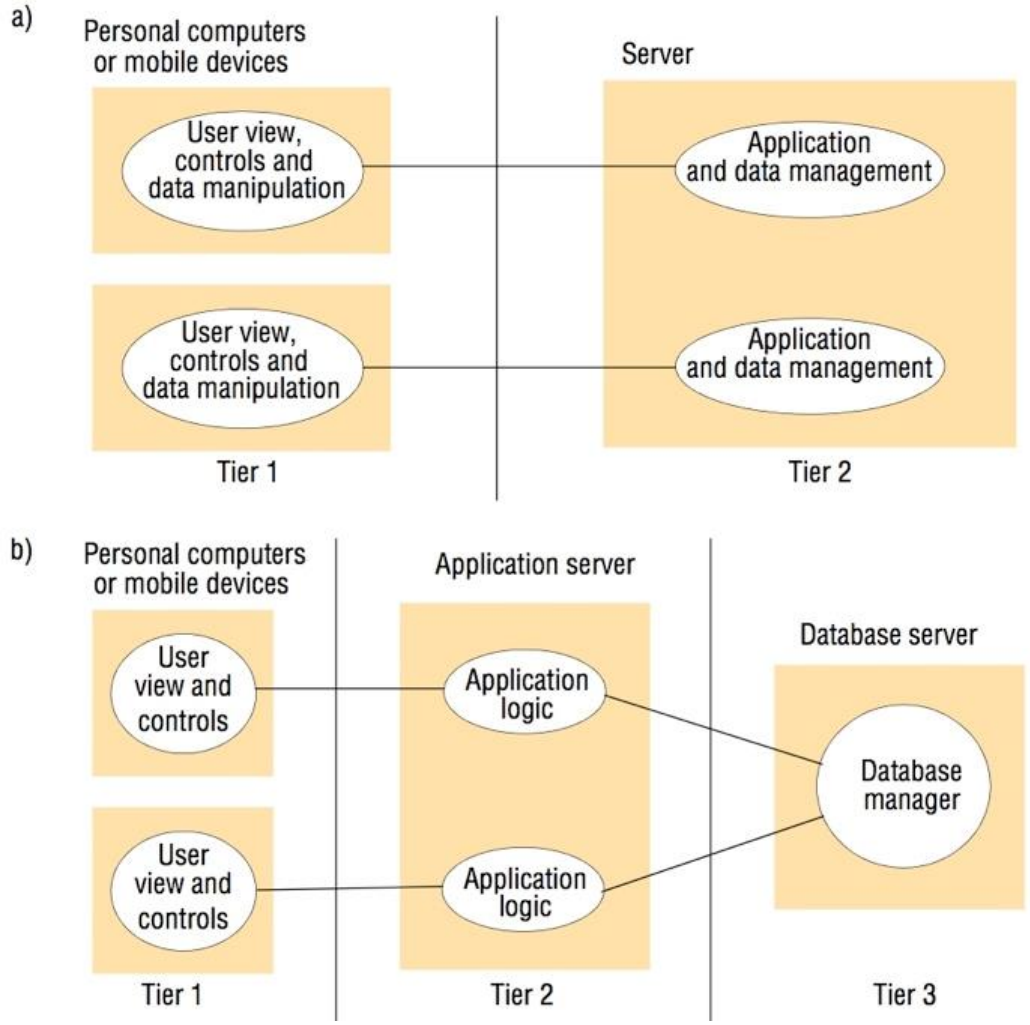
- **Applications** are decomposed into **three functions:**

1. Presentation logic is concerned with handling user interaction and updating the view of the application as presented to the user.

2. Application logic (algorithmic) or (business) is concerned with the detailed application-specific processing associated with the application.

3. Data logic is concerned with the persistent storage of the application, typically in a database management system.

Figure 2.8
Two-tier and three-tier architectures

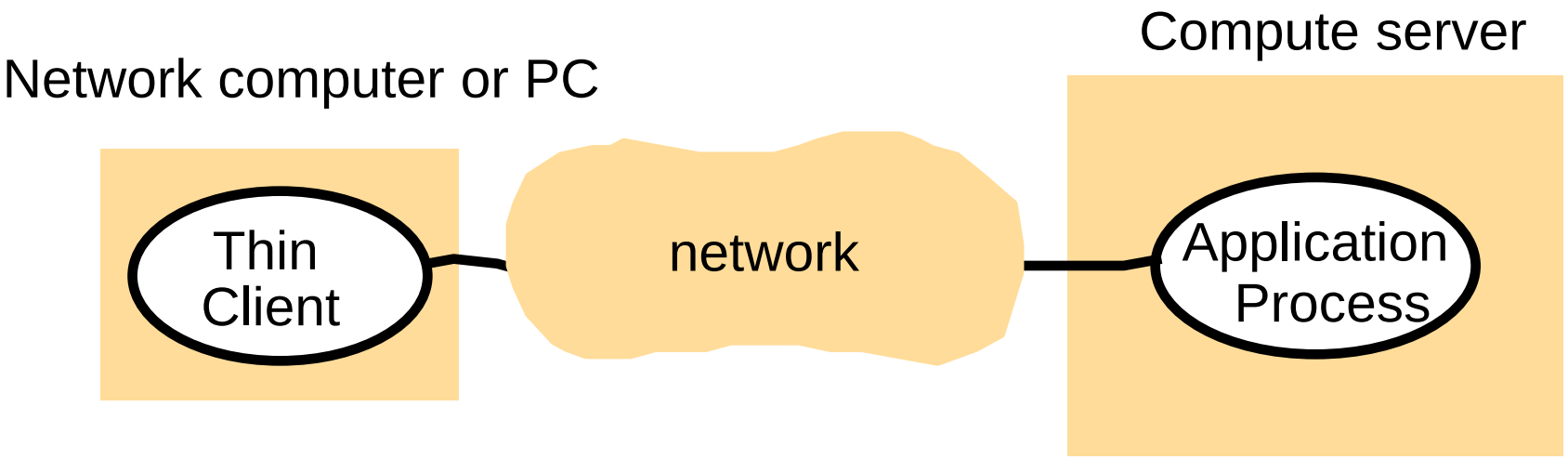


Architectural Patterns

Thin Clients

- Enabling access to sophisticated networked services, provided for example by a cloud solution.
- The term **thin client** refers to a software layer that supports a window-based user interface that is local to the user while executing application programs or accessing services on a remote computer.
- **Figure 2.10** illustrates a thin client accessing a compute server over the Internet.
- The **advantage** of this approach is that potentially simple local devices (smart phones) can be enhanced with networked services and capabilities.
- The **drawback** of the thin client architecture is in highly interactive graphical activities such as image processing, the delays experienced by users are increased due to both network and operating system latencies.

Figure 2.10
Thin clients and compute servers



Thin Clients Examples

- **PutTTY** is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP (Secure Copy Protocol), SSH (Secure Shell), Telnet, rlogin, and raw socket connection. It can also connect to a serial port. The name "PuTTY" has no official meaning.

- **WinSCP (Windows Secure Copy)** is a free and open-source SSH File Transfer Protocol (SFTP), File Transfer Protocol (FTP), and secure copy protocol (SCP) client for Microsoft Windows. Its main function is secure file transfer between a local computer and a remote server.

- **Zoom Desktop Client.**

Fundamental Models

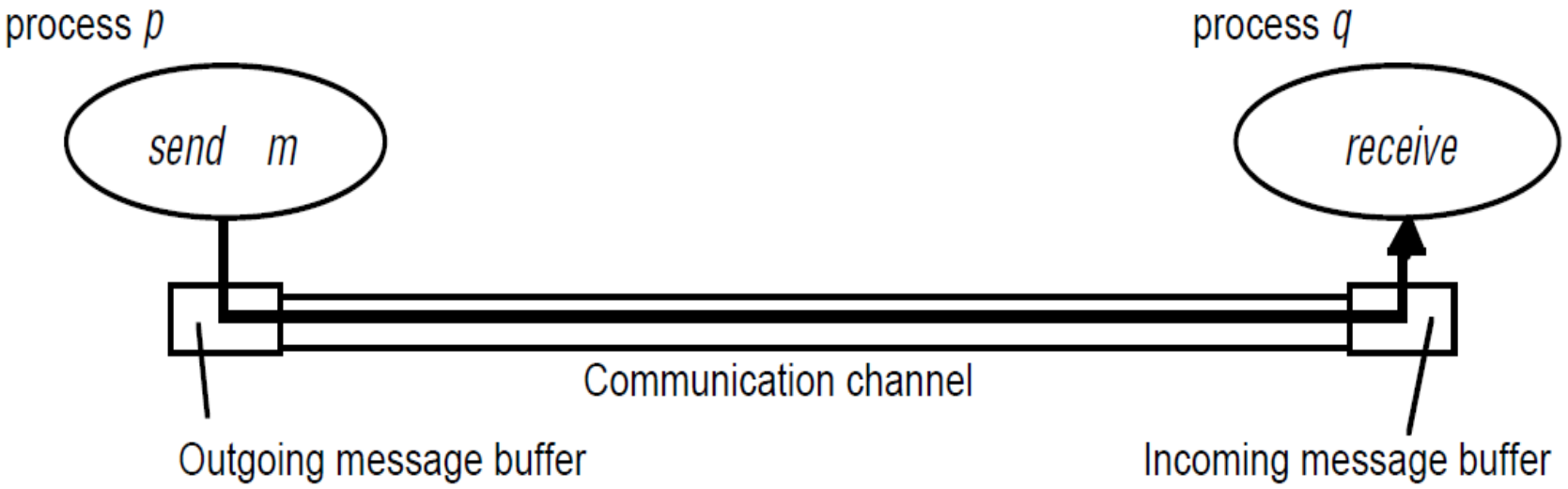
- Models based on the fundamental properties that allow us to be more specific about their characteristics as well as their failures and security risks that may carry.

Fundamental Models

Interaction:

- Computation occurs within processes that interact with each other through massage passing.
- Network delays result in coordination limitations.
- The interaction model must reflect the facts that communication takes place with delays that are of considerable durations, and the difficulty of maintaining same access time to the different DS resources.

Figure 2.14
Processes and channels



Fundamental Models

Failure:

- DS is threatened whenever a failure occurs.
- This model classifies faults and provides basis for the analysis of their potential effects as well as for the design of a DS that is able to tolerate those faults.

Security:

- DS openness exposes them to attacks from internal and external agents.
- This model analyzes and classifies the possible attacks and is used to design a DS that is able to resists the attacks.

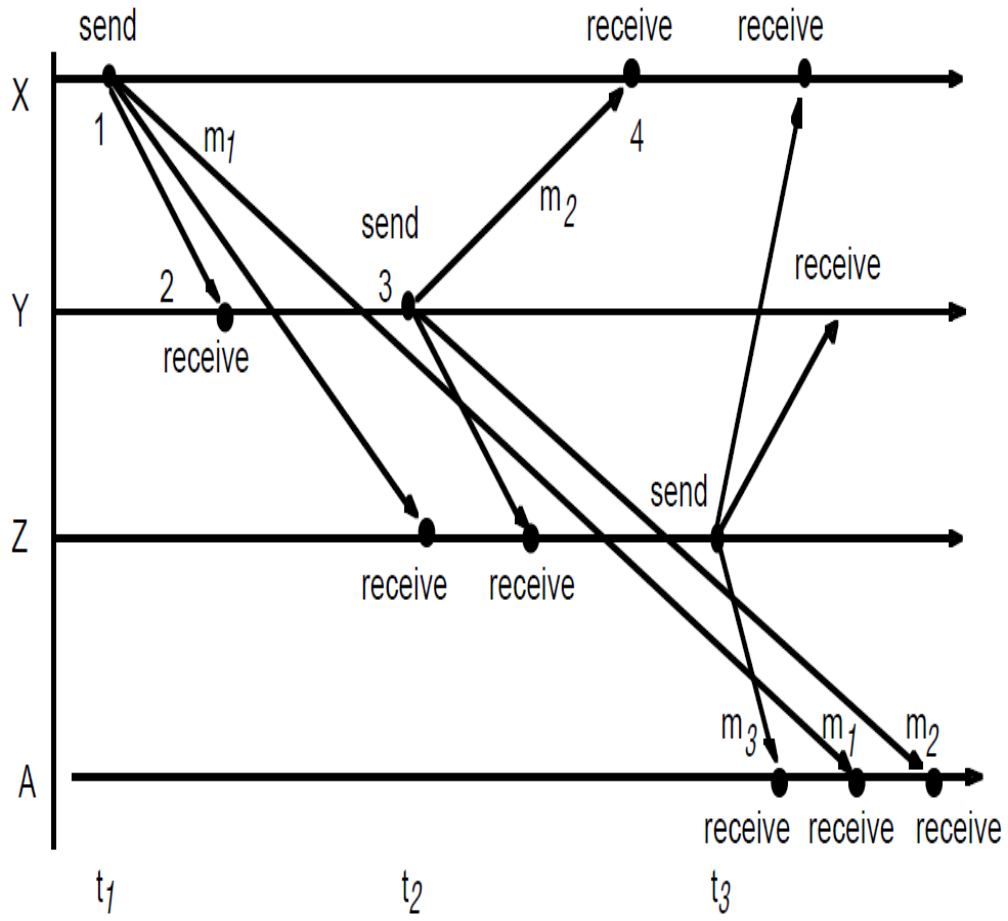
Figure 2.15
Omission (disregard) and arbitrary failures

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 (show) arbitrary behavior: It may send/transmit arbitrary messages at arbitrary times, commit omissions (perform deletion); a process may stop or take an incorrect step.

Figure 2.11
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.

Figure 2.13
Real-time ordering of events

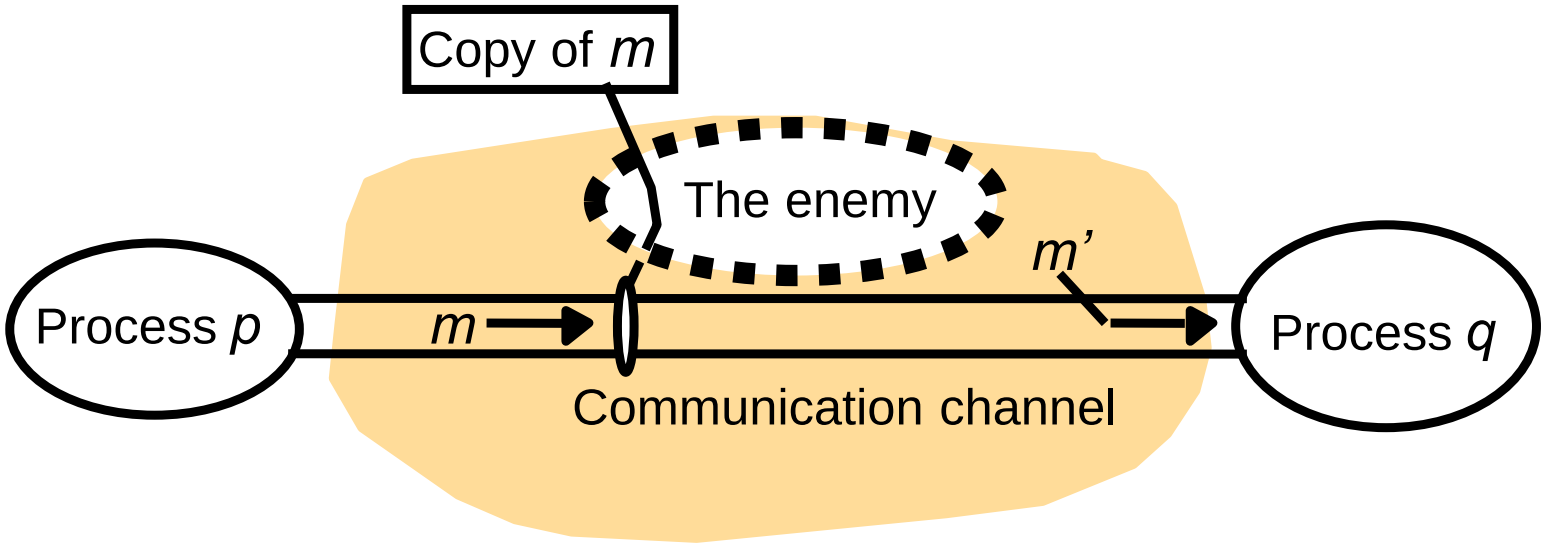


- X sends 'Hi' to Y, Z, and A.
- When Y, Z, A receive it, they send 'Re: Hi' to all other nodes.

- Time delays might cause messages to arrive in wrong order.

- Look at 'A'. It received messages as:
- Re: Hi
  Hi
  Re: Hi

Figure 2.18
The enemy



- Enemy intercepts message *m* and replace it with his own copy ***m'***.
- Message ***m'*** might be harmful or contains false information.

Figure 2.19
Secure channels