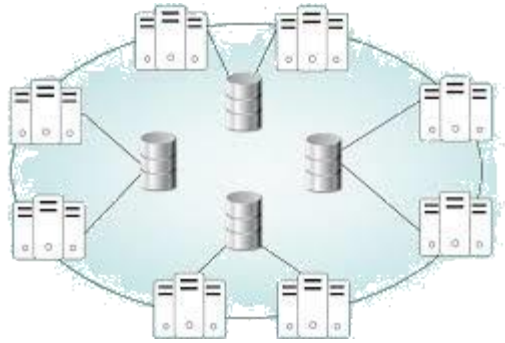
Distributed and Parallel Computer Systems

CSC 423



Lecture 4

Client-Sever Models & Fundamental Models

INSTRUCTOR

DR / AYMAN SOLIMAN

➢ Contents

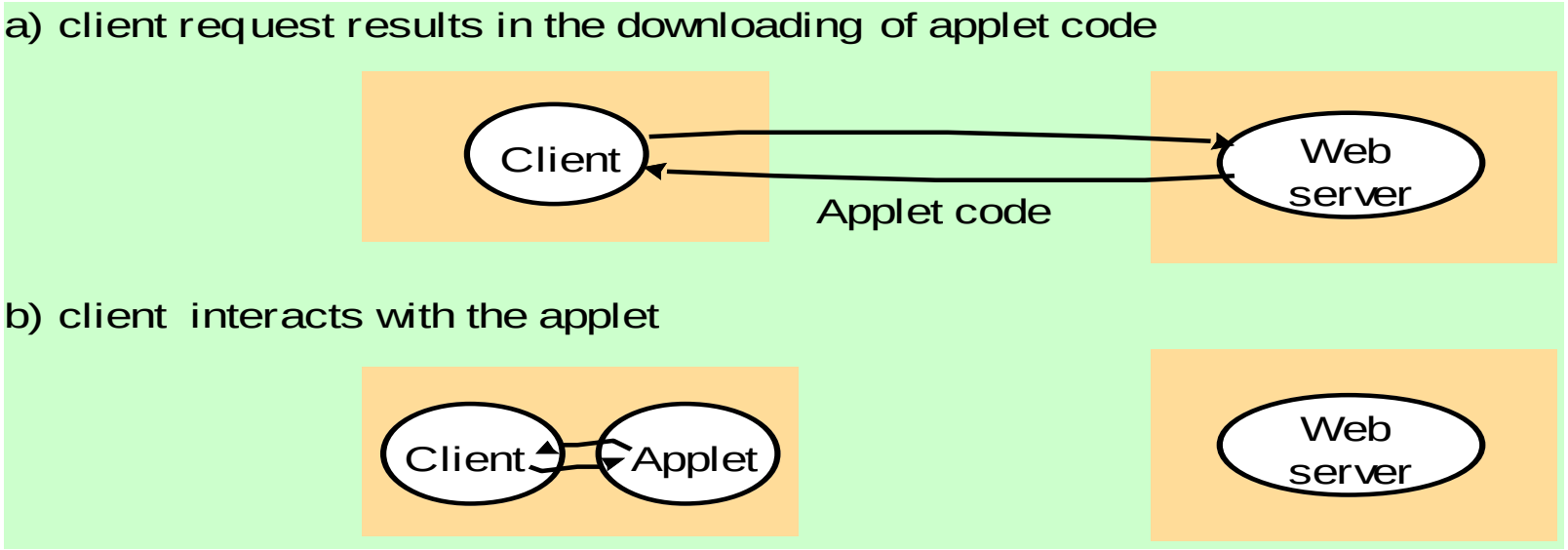
- ➢ Alternative Client-Sever models driven by:
  - Mobile code, mobile agents, network computers, thin clients, mobile devices and spontaneous networking
  - Design Challenges/Requirements
- ➢ Fundamental Models – Formal Description
  - Interaction model
  - Failure model
  - Security model
- ➢ Summary

❑ Variants of Client Server Model

- Several variations on the client-server model can be derived from the consideration of the following factors:
  - The use of mobile code and mobile agents;
  - Users need for low-cost computers with limited hardware resources that are simple to manage;
  - The requirement to add and remove mobile devices in a convenient manner.

❑ 1. Mobile code

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



- Mobile codes such as Applets are potential security threat, so the browser gives applets limited access to local resources (e.g., NO access to local/user file system).

❑ 2. Mobile Agents

- A running program (code and data) that travels from one computer to another in a network carrying out of an autonomous task.

- **Advantages**: flexibility, savings in communications cost

- Potential security threat to the resources in computers they visit. The environment receiving agent should decide which of the local resource to allow. (e.g., web servers).

- Agents themselves can be vulnerable – they may not be able to complete task if they are refused access.

❑ 3. Network Computers

- The management of application files and the maintenance of a local software based require considerable technical effort of a nature that most users are not qualified to provide.

- The network computer is a response to this problem. It downloads its operating system and any application software needed by the user from a remote file server.

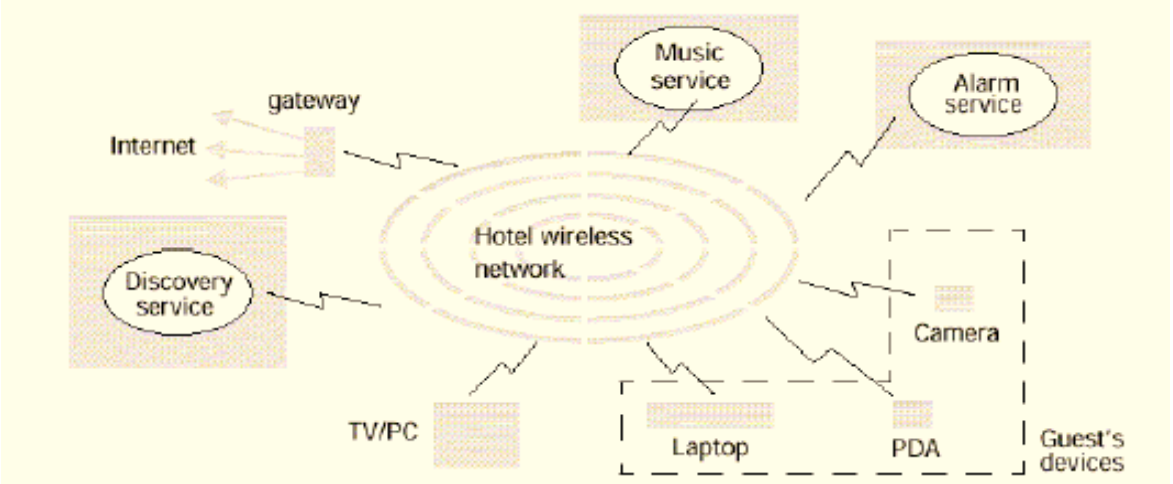
- Applications are run locally but the files are managed by a remote file server.

❑ 3. Network Computers

- Network applications such as a web browser can also be run.
- Since all the applications data and code are stored by a file server, the users may migrate from one network computer to another.
- The processor and memory capacities of a network computer can be constrained in order to reduce its cost.

❑ Mobile devices and spontaneous networking

- The world is increasingly populated by small and portal computing devices.
- Need to provide discovery services:
  - (1) registration service to enable servers to publish their services and
  - (2) look up service to allow clients to discover services that meet their requirements.



➢ Contents

- Alternative Client-Sever models driven by:
  - Mobile code, mobile agents, network computers, thin clients, mobile devices and spontaneous networking
  - Design Challenges/Requirements
- Fundamental Models – Formal Description
  - Interaction model
  - Failure model
  - Security model
- Summary

❑ Design Requirements/Challenges of Distributed Systems

- **Performance Issues**
  - Responsiveness
    - Support interactive clients
      - ✓ Use caching and replication
  - Throughput (Concurrency)
  - Load balancing and timeliness
- **Quality of Service:**
  - Reliability
  - Security
  - Adaptive performance.
- **Dependability issues:**
  - Correctness, and fault tolerance
  - Dependable applications continue to work in the presence of faults in hardware, software, and networks.

➢ Contents

- Alternative Client-Sever models driven by:
  - Mobile code, mobile agents, network computers, thin clients, mobile devices and spontaneous networking
  - Design Challenges/Requirements
- Fundamental Models – Formal Description
  - Interaction model
  - Failure model
  - Security model
- Summary

❑ Fundamental Models

- Fundamental Models are concerned with the main entities in the system,
- and how they interact,
- and the characteristics that affect their individual and collective behavior.
- All Architectural Models composed of processes that communicate with each other by sending messages over a computer networks.

❑ Fundamental Models

- Models addressing time synchronization, message delays, failures, security issues are addressed as:

1) Interaction Model – deals with performance and the difficulty of setting of time limits in a distributed system.

2) Failure Model – specification of the faults that can be exhibited by processes

3) Secure Model – discusses possible threats to processes and communication channels.

❑ 1- Interaction Model

- Distributed systems composed of many processes interact like:
  - Multiple server processes may cooperate with one another to provide service like domain name service.
  - A set of peer processes may cooperate with one another to achieve a common goal, for example voice\video conferencing system.

❑ 1- Interaction Model

- The processes interact by passing messages, resulting in:
  - Communication (information flow)
  - Coordination (synchronization and ordering of activities) between processes.

- Two significant factors :
  - Communication performance.
  - It is impossible to maintain a single global notion of time.

❑ 1- Interaction Model: Performance of Communication Channel

- The communication channel realized in a variety of ways:
  - Streams
  - Simple message passing over a network.

❑ 1- Interaction Model: Performance of Communication Channel

- Communication over a computer network has performance characteristics:
  - Latency:
    - A delay between the start of a message's transmission from one process to the beginning of receipt by another.
  - Bandwidth:
    - The total amount of information that can be transmitted over in a given time.
  - Jitter:
    - The variation in the time taken to deliver a series of messages. It is very relevant to multimedia data.

❑ 1- Interaction Model: Computer clocks and timing events

- Each computer in a DS has its own internal clock, which can be used by local processes to obtain the value of the current time.

- However, 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.

❑ 1- Interaction Model: Computer clocks and timing events

- Even if the clocks on all the computers in a DS are set to the same time initially, their clocks would eventually vary quite significantly unless corrections are applied.
  - There are several techniques to correct time on computer clocks.
  - For example, computers may use radio receivers to get readings from GPS (Global Positioning System).

❑ 1- Interaction Model: Two variants of the interaction model

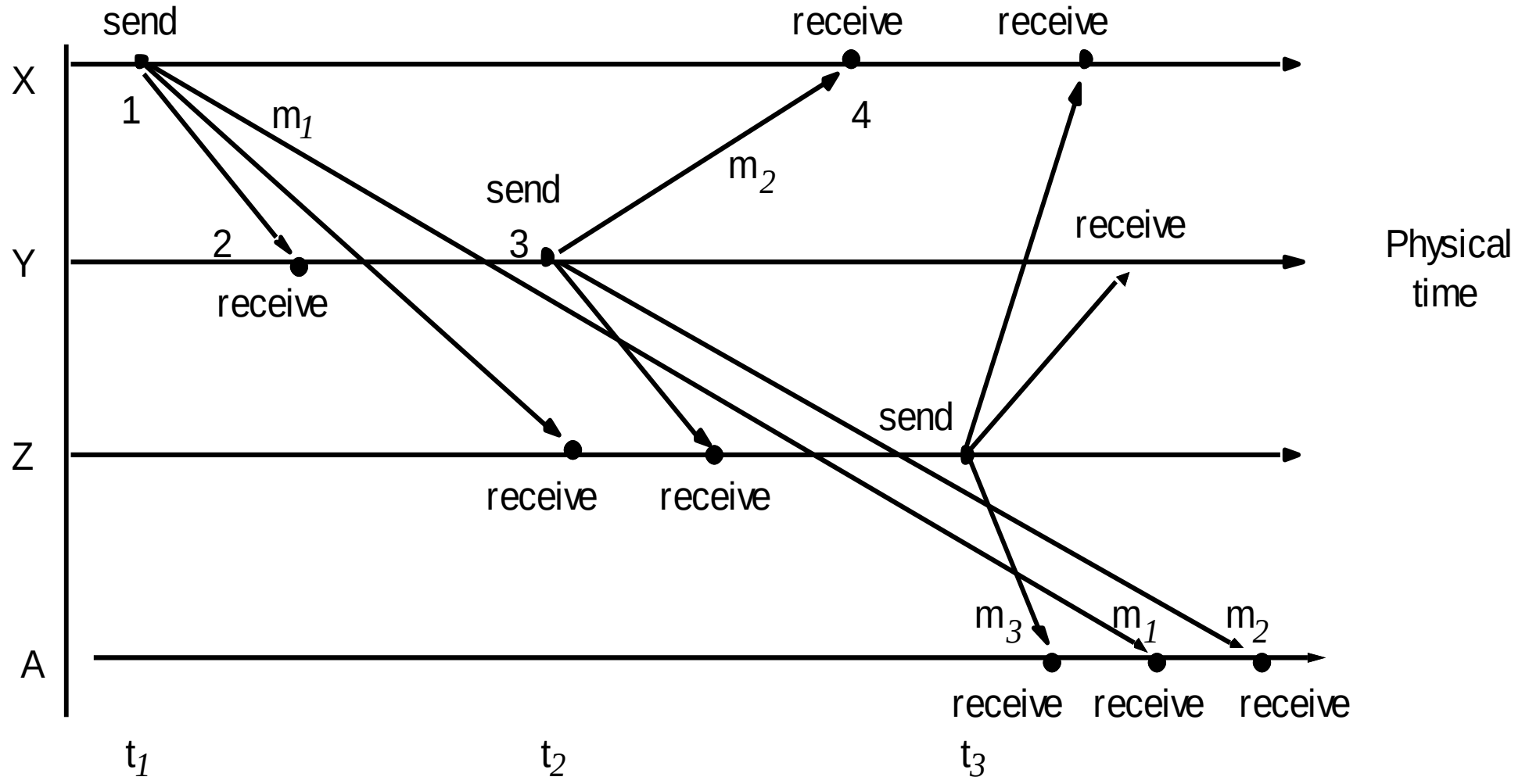
- Synchronous DS – hard to achieve the following:
  - The time take to execute a step of a process has know lower and upper bounds.
  - Each message transmitted over a channel is received within a known bounded time.
  - Each process has a local clock whose drift rate from real time has known bound.
  - Use time out to detect the failure of a process.

- Asynchronous DS: There is NO bounds on the following:
  - Process execution speeds
  - Message transmission delays
  - Clock drift rates.

❑ 1- Interaction Model: Event Ordering

- The execution of a system can be described in terms of events and their ordering
- In many DS applications we are interested in knowing whether an event occurred before, after, or concurrently with another event at another processes.
- Consider a mailing list with users X, Y, Z, and A.

❑ Real-time ordering of events



❑ Inbox of User A looks like:

From	Subject
Z	Re: Meeting
X	Meeting
Y	Re: Meeting

- Due to independent delivery in message delivery, message may be delivered in different order.
- If messages m1, m2, m3 carry their time t1, t2, t3, then they can be displayed to users accordingly to their time ordering.
- BUT clocks cannot be synchronized perfectly across a distributed system (e.g: logical ordering)

❑ 2- Failure Model

- In a DS, both processes and communication channels may fail .

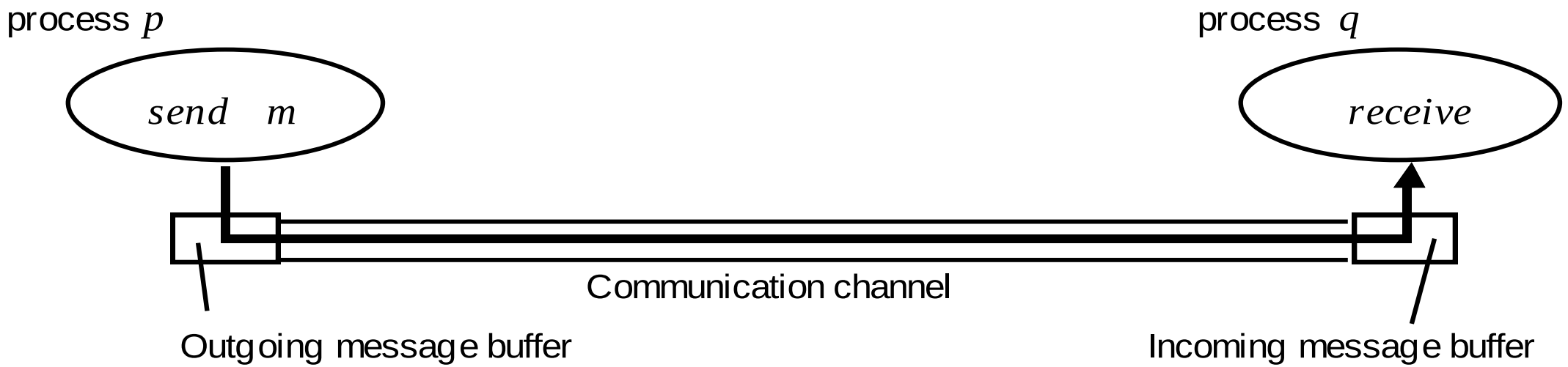
- Types of failures:

1) Omission Failure (فشل الإغفال)

2) Arbitrary Failure (فشل اعتباطي)

3) Timing Failure (فشل زمنى)

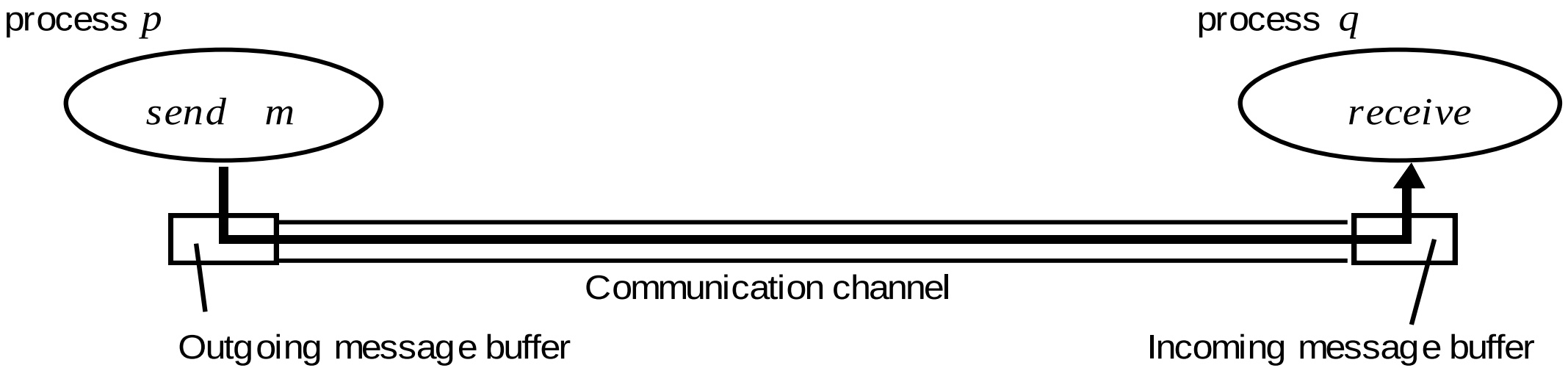
❑ Processes and channels



- Communication channel produces an omission failure if it does not transport a message from outgoing message buffer to incoming message buffer.

known as "dropping messages"

❑ Processes and channels



- "dropping messages" is generally caused by lack of buffer space at the receiver or a network transmission error.

❑ 1) and 2) 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, a process may stop or take an incorrect step.

❑ 3) Timing failures

- It is applicable in synchronous distributed system where time limit is set on process execution time.

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

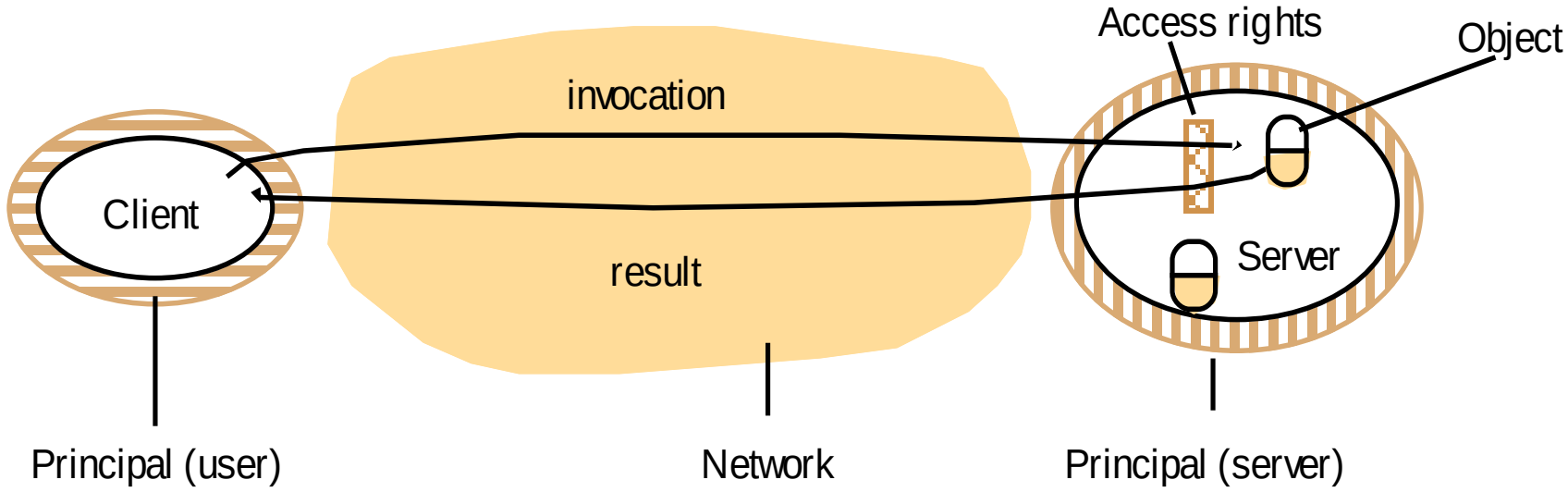
- It is possible to construct reliable services from components that exhibit failures.
  - For example, multiple servers that hold replicas of data can continue to provide a service when one of them crashes.

- can mask the failure of the components on which it depends:
  - Checksums are used to mask corrupted messages.

❑ 3- Security Model

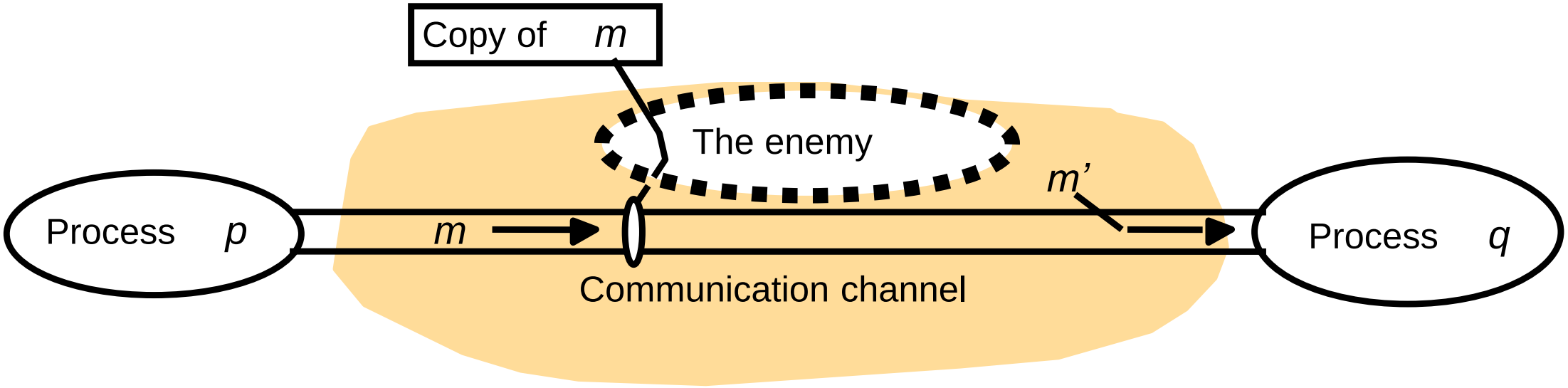
- The security of a DS can be achieved by:
  - Securing the processes and the channels used in their interactions
  - Protecting the objects that they encapsulate against unauthorized access.

❑ Protecting Objects: Objects and principals



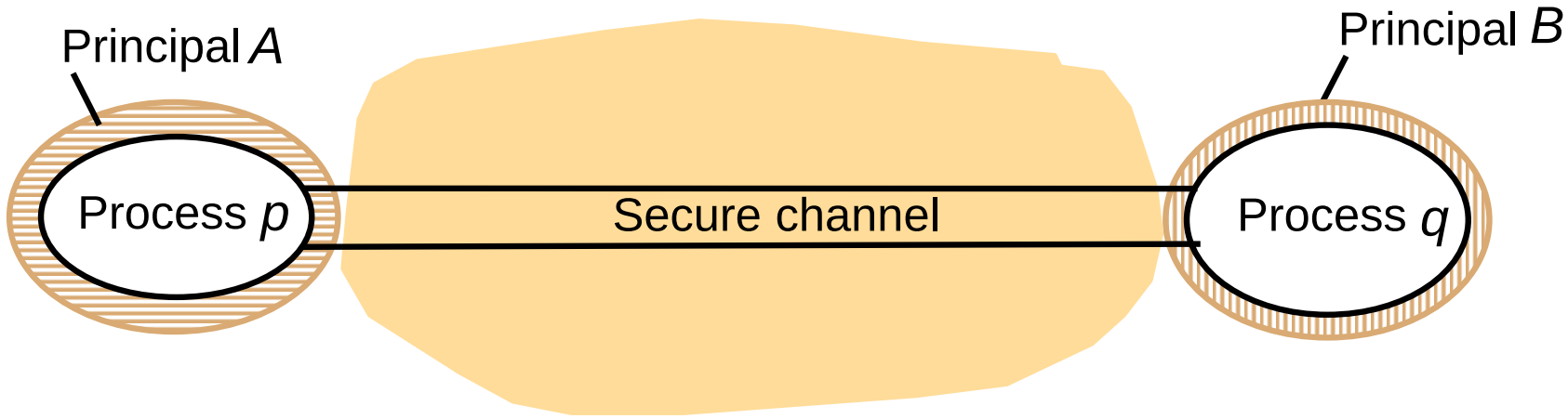
- Use "access rights" that define who is allowed to perform operation on an object.
- The server should verify the identity of the (user) and checking that they have sufficient access rights to perform the requested operation on the particular object, rejecting those who do not.

❑ The enemy



- An enemy can send any process or reading/copying message between a pair of processes
- Threats form a potential enemy: threats to processes, threats to communication channels, and denial of service.

❑ Defeating security threats: Secure channels



- Encryption and authentication are used to build secure channels.
- Each of the processes knows the identity of the principal on whose behalf the other process is executing and can check their access rights before performing an operation.

➢ Contents

- ➢ Alternative Client-Sever models driven by:
  - Mobile code, mobile agents, network computers, thin clients, mobile devices and spontaneous networking
  - Design Challenges/Requirements
- ➢ Fundamental Models – Formal Description
  - Interaction model
  - Failure model
  - Security model
- ➢ Summary

❑ Summary

- Most DSs are arranged accordingly to one of a variety of architectural models:
  - Client-Server
    - Clients and a Single Sever, Multiple Servers, Proxy Servers with Cache, Peer Model
  - Alternative Client-Sever models driven by:
    - Mobile code, mobile agents, network computers, mobile devices and spontaneous networking
- Fundamental Models – formal description
  - Interaction, failure, and Security models.

Thank you