LECTURE 17

Fundamental Models

Interfaces and Objects

- Interface definitions specify the set of functions available for invocation in a server (or peer) processes.
- In OO paradigm, distributed processes can be constructed as objects whose methods can be accessed by remote invocation (COBRA approach).
- Many objects can be encapsulated in server or peer processes.
- Number, types, and locations (in some implementation) of objects may change dynamically as system activates require.
- Therefore, new services can be instantiated and immediately be made available for invocation.

The Message Passing Paradigm

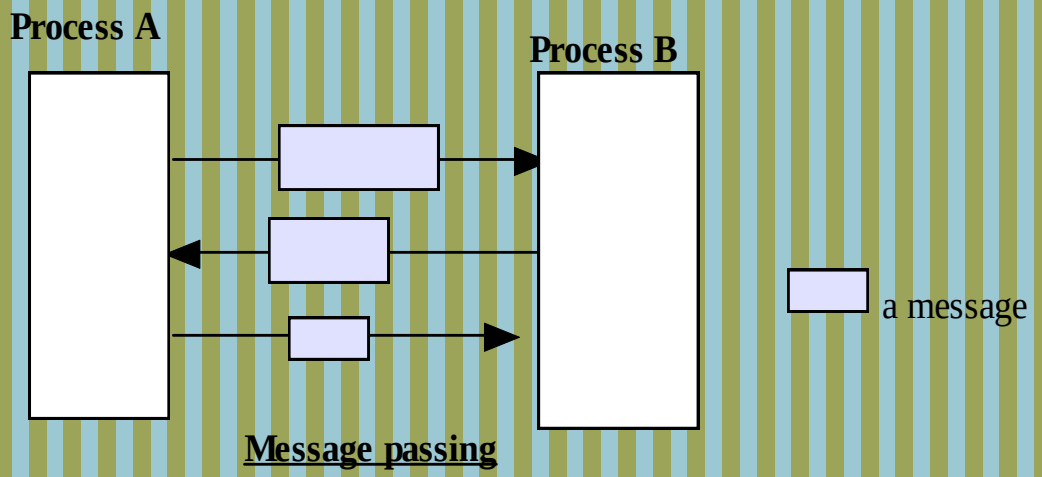
Message passing is the most fundamental paradigm for distributed applications.

- **A process sends a message representing a request.**
- **The message is delivered to a receiver, which processes the request, and sends a message in response.**
- **In turn, the reply may trigger a further request, which leads to a subsequent reply, and so forth. The message-**

The Message Passing Paradigm - 2

Message passing is the most fundamental paradigm for distributed applications.

- **A process sends a message representing a request.**
- **The message is delivered to a receiver, which processes the request, and sends a message in response.**
- **In turn, the reply may trigger a further request, which leads to a subsequent reply, and so forth. -**



The Message Passing Paradigm - 3

- **The basic operations required to support the basic message passing paradigm are *send*, and *receive*.**
- **For connection-oriented communication, the operations *connect* and *disconnect* are also required.**
- **With the abstraction provided by this model, the interconnected processes perform input and output to each other, in a manner similar to file I/O. The I/O operations encapsulate the detail of network communication at the operating-system level.**
- **The socket application programming interface is based on this paradigm.**
  - **http://java.sun.com/products/jdk/1.2/docs/api/index.html**
  - **http://www.sockets.com/**

Architectures Design Requirements

- **Performance Issues:**
  - Considered under the following factors:
    - Responsiveness:
      - Fast and consistent response time is important for the users of interactive applications.
      - Response speed is determined by the load and performance of the server and the network and the delay in all the involved software components.
      - System must be composed of relatively few software layers and small quantities of transferred data to achieve good response times.
    - Throughput:
      - The rate at which work is done for all users in a distributed system.
    - Load balancing:
      - Enable applications and service processes to proceed concurrently without competing for the same resources.
      - Exploit (مأثرة)available processing resources.

Architectures Design Requirements

- **Quality of Service:**
  - Main system properties that affect the service quality are:
    - Reliability: related to failure fundamental model (discussed later).
    - Performance: ability to meet timeliness guarantees.
    - Security: related to security fundamental model (discussed later).
    - Adaptability: ability to meet changing resource availability and system configurations.
- **Dependability issues:**
  - A requirement in most application domains.
  - Achieved by:
    - Fault tolerance: continuing to function in the presence of failures.
    - Security: locate sensitive data only in secure computers.
    - Correctness of distributed concurrent programs: research topic.

Fundamental Models
(Interaction Model)

- Distributed systems consists of multiple interacting processes with private set of data that can access.
- Distributed processes behavior is described by *distributed algorithms*.
- Distributed algorithms define the steps to be taken by each process in the system including the transmission of messages between them.
- Transmitted messages transfer information between these processes and coordinate their ordering and synchronization activities.

Fundamental Models
(Interaction Model)

- Interacting processes in a distributed system are affected by two significant factors:
  1. Performance of communication channels: is characterized by:
     - **Latency**: delay between sending and receipt of a message including
       - Network access time.
       - Time for first bit transmitted through a network to reach its destination.
       - Processing time within the sending and receiving processes.
     - **Throughput**: number of units (e.g., packets) delivered per time unit.
     - **Bandwidth**: total amount of information transmitted per time unit.
     - **Jitter**: variation in the time taken to deliver multiple messages of the same type (relevant to multimedia data).

Fundamental Models
(Interaction Model)

2. Computer clocks:
   - Each computer in a distributed system has its own internal clock to supply the value of the current time to local processes.
   - Therefore, two processes running on different computers read their clocks at the same time may take different time values.
   - *Clock drift rate* refers to the relative amount a computer clock differs from a perfect reference clock.
   - Several approaches to correcting the times on computer clocks are proposed.
   - Clock corrections can be made by sending messages, from a computer has an accurate time to other computers, which will still be affected by network delays.

Fundamental Models
(Interaction Model)

- Setting time limits for process execution, as message delivery, in a distributed system is hard.
- Two opposing extreme positions provide a pair of simple interaction models:
  - *Synchronous* distributed systems:
    - A system in which the following bounds are defined:
      - 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 perfect time has a known bound.
    - Easier to handle, but determining realistic bounds can be hard or impossible.

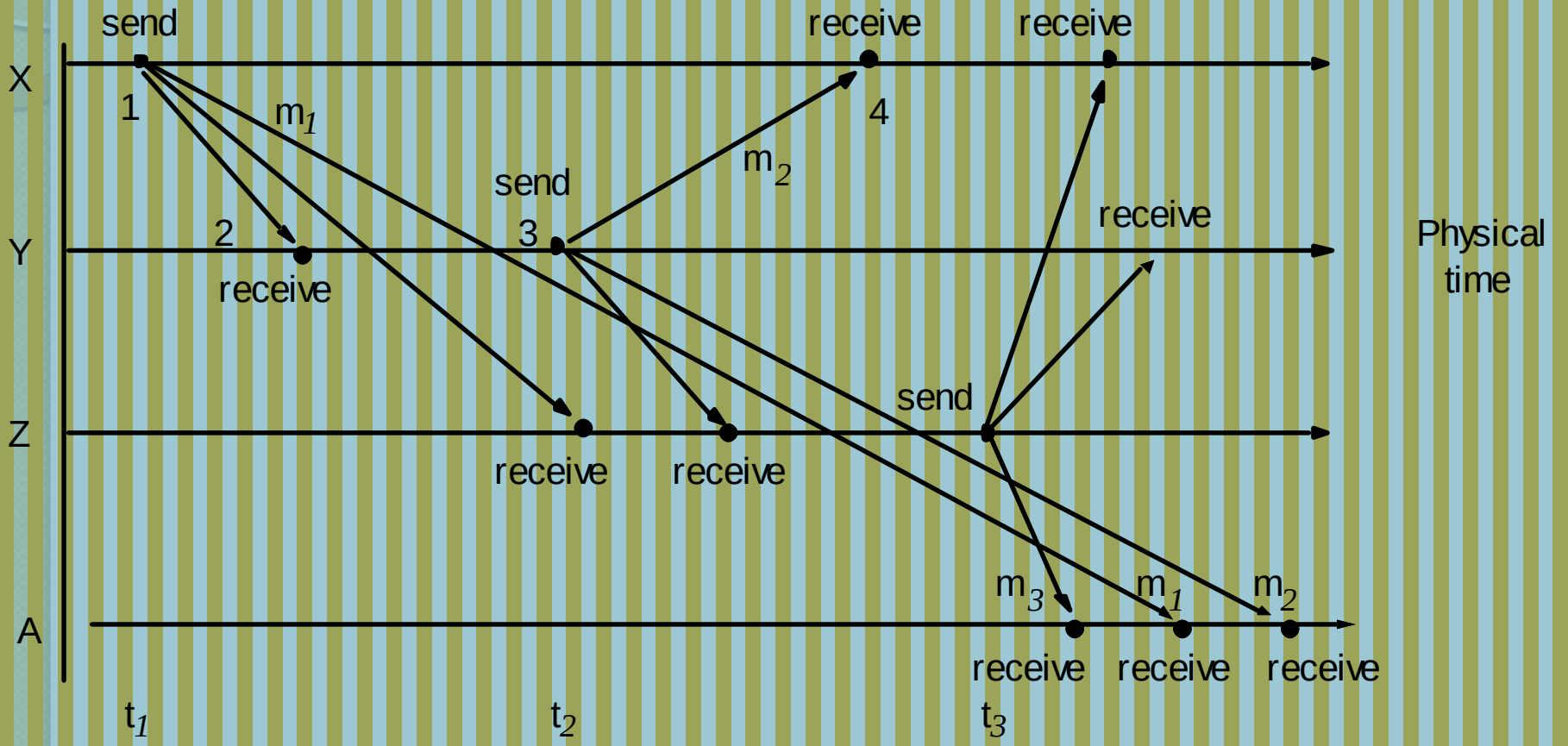
Fundamental Models
(Interaction Model)

- *Asynchronous* distributed systems:
  - A system in which there are no bounds on:
    - process execution times.
    - message delivery times.
    - clock drift rate.
  - Allows no assumptions about the time intervals involved in any execution.
  - Exactly models the Internet.
    - Browsers are designed to allow users to do other things while they are waiting.
  - More abstract and general:
    - A distributed algorithm executing on one system is likely to also work on another one.

Fundamental Models
(Interaction Model)

- **Event ordering**: when need to know if an event at one process (sending or receiving a message) occurred before, after, or concurrently with another event at another process.
- It is impossible for any process in a distributed system to have a view on the current global state of the system.
- The execution of a system can be described in terms of events and their ordering despite the lack of accurate clocks.
- Logical clocks define some event order based on causality.
- Logical time can be used to provide ordering among events in different computers in a distributed system (since real clocks cannot be synchronized).

Fundamental Models
(Interaction Model)

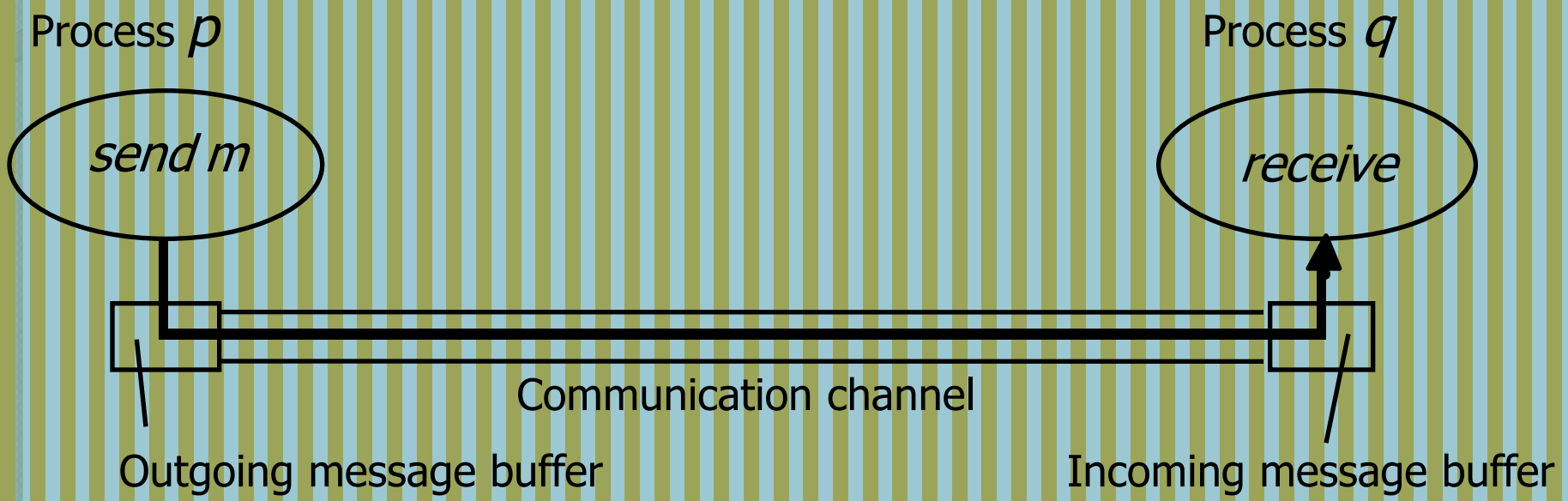


Real-time ordering of events

Fundamental Models
(Failure Model)

- Defines the ways in which failure may occur in order to provide an understanding of its effects.
- A taxonomy of failures which distinguish between the failures of processes and communication channels is provided:
  - *Omission* failures
    - Process or channel failed to do something.
  - *Arbitrary* failures
    - Any type of error can occur in processes or channels (worst).
  - *Timing* failures
    - Applicable only to synchronous distributed systems where time limits may not be met.

Fundamental Models
(Failure Model)



Processes and channels

Fundamental Models
(Failure Model)

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.

Fundamental Models
(Failure Model)

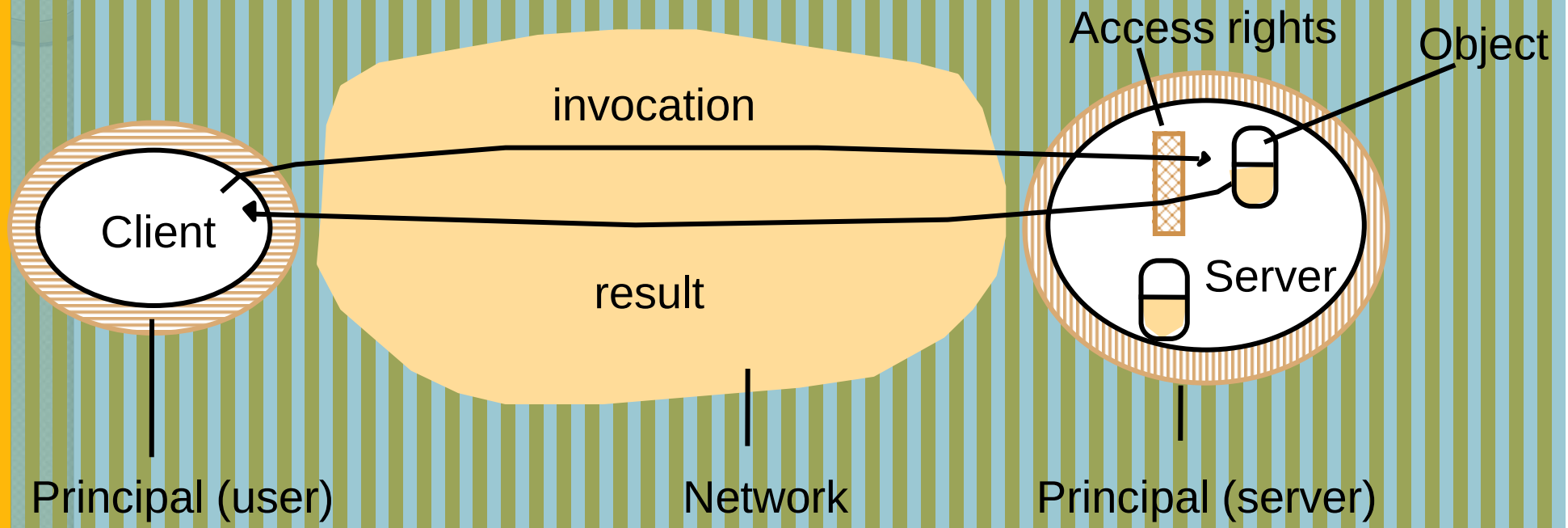
Timing failures

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.

Fundamental Models
(Security Model)

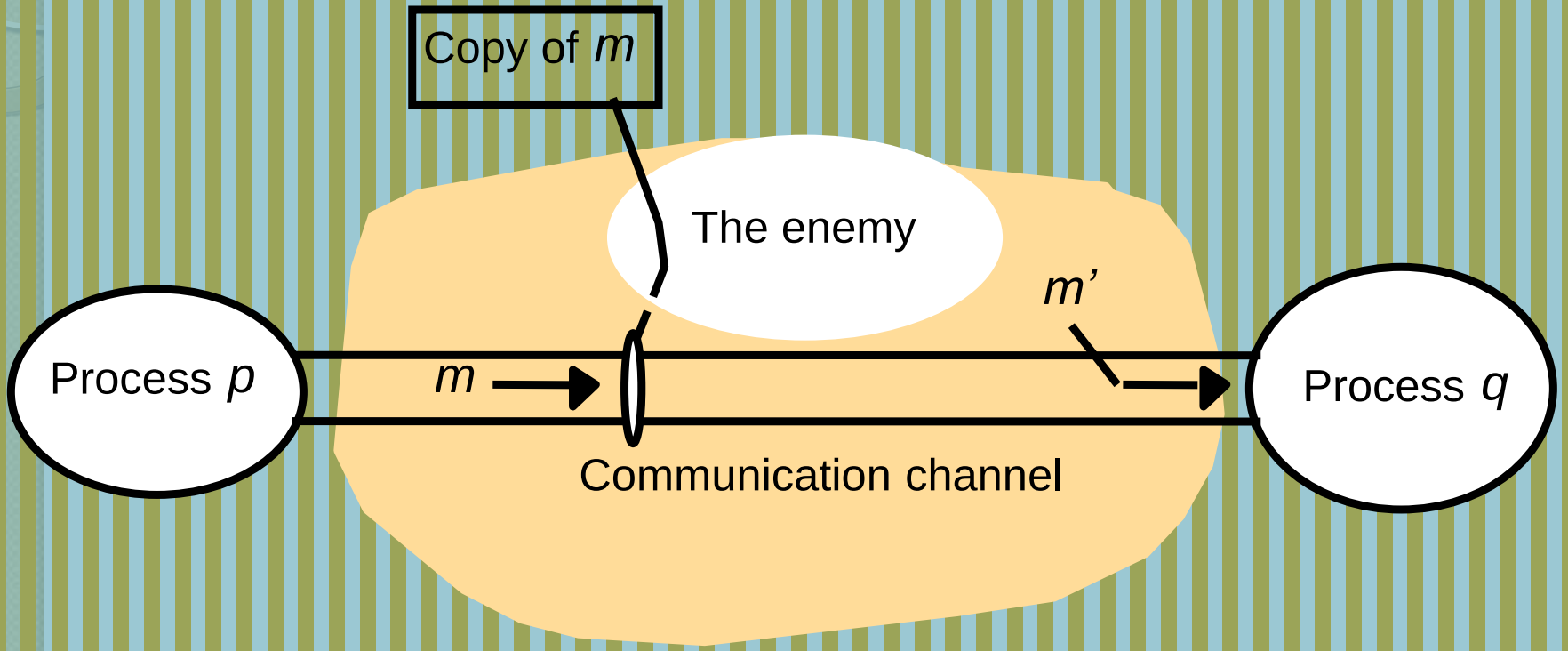
- Secure processes and channels and protect objects encapsulated against unauthorized access.
- **Protecting access to objects**
  - Access rights
  - In client server systems: involves authentication of clients.
- **Protecting processes and interactions**
  - Threats to processes: problem of unauthenticated requests / replies.
    - e.g., "man in the middle"
  - Threats to communication channels: enemy may copy, alter or inject messages as they travel across network.
    - Use of “secure” channels, based on cryptographic methods.

Fundamental Models
(Security Model)



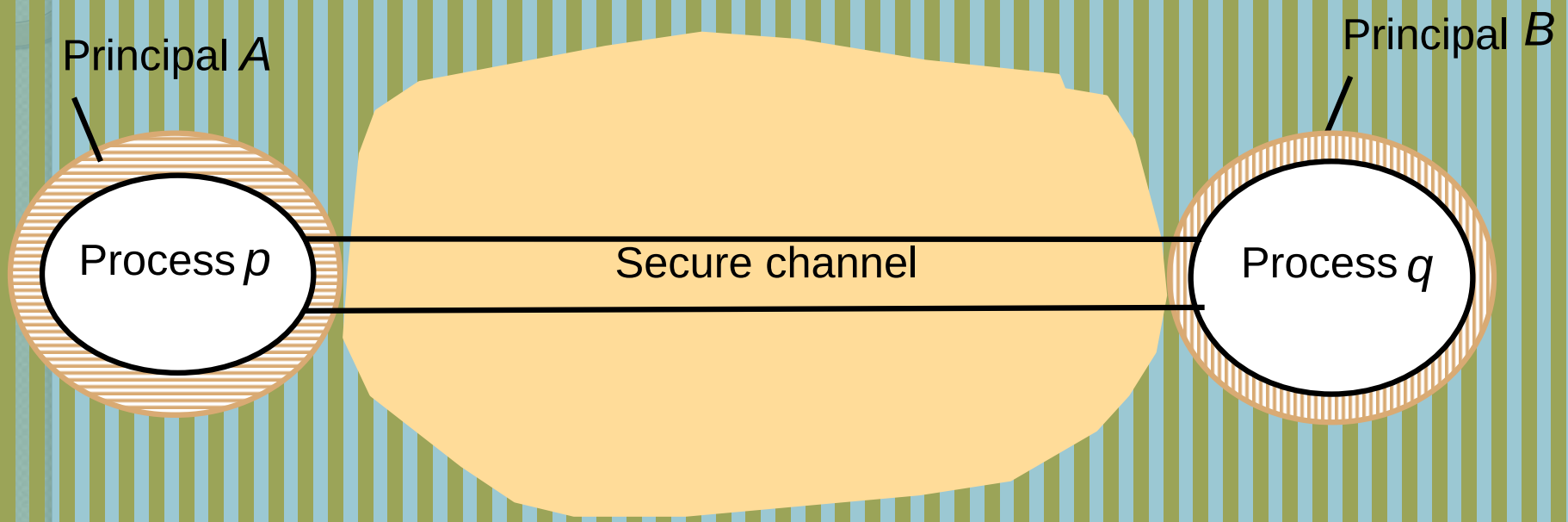
Objects and principals

Fundamental Models
(Security Model)



The enemy

Fundamental Models
(Security Model)



Secure channels

Fundamental Models
(Security Model)

- **Denial of service**
  - e.g., “pings” to selected web sites
  - Generating debilitating network or server load so that network services become de facto unavailable
- **Mobile code:**
  - Requires executability privileges on target machine
  - Code may be malicious (e.g., mail worms)

ASSIGNMENT

- Q: Explain various fundamental models in distributed operating system.