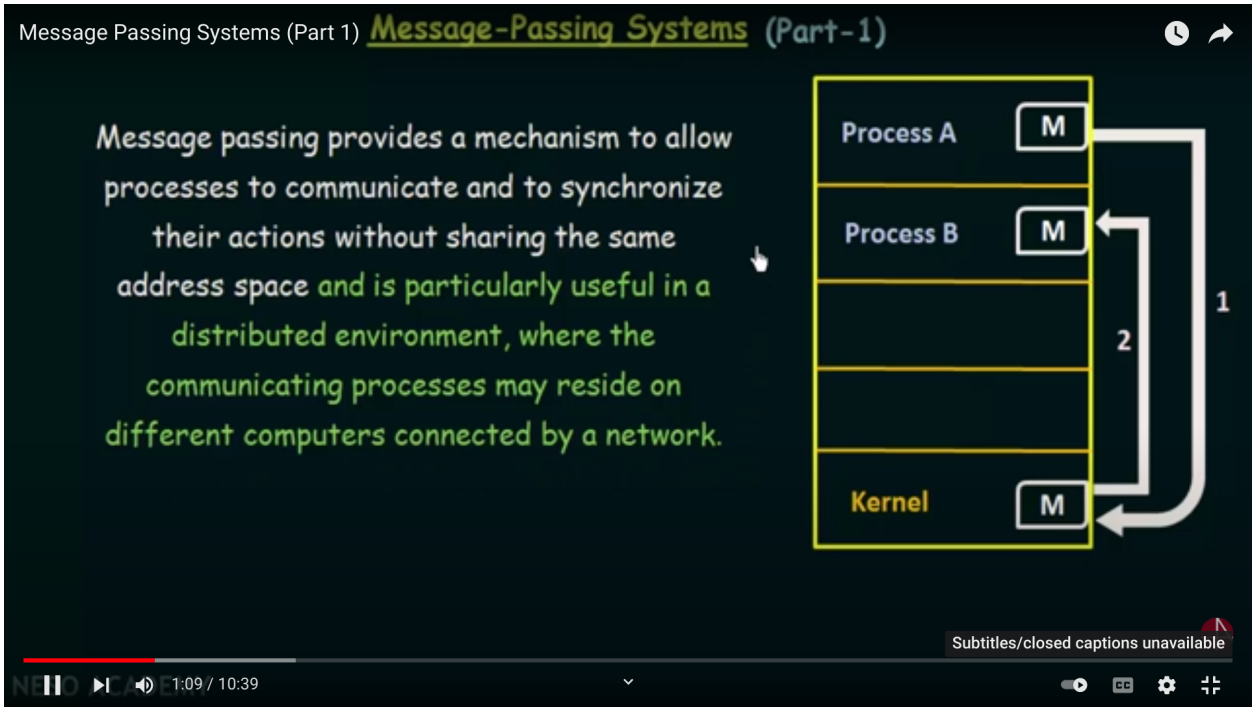
Distributed Algorithm

- runs on a distributed system.
- Distributed system is a collection of independent computers that do not share their memory.

Election Algorithms:

- **Election algorithms** are designed to choose a coordinator.
- choose a process from group of processors to act as a coordinator.
- Election algorithm basically determines where a new copy of coordinator should be restarted.
- Election algorithm assumes that every active process in the system has a unique priority number.
- The process with highest priority will be chosen as a new coordinator.



A message-passing facility provides at least two operations:

- send (message)

and

- receive (message)

Messages sent by a process can be of either fixed or variable size.

FIXED SIZE: The system-level implementation is straightforward.
But makes the task of programming more difficult.

VARIABLE SIZE: Requires a more complex system-level implementation.
But the programming task becomes simpler.

NESO ACADEMY

If processes P and Q want to communicate, they must send messages to and receive messages from each other.

A communication link must exist between them.

This link can be implemented in a variety of ways. There are several methods for logically implementing a link and the send( ) /receive( ) operations, like:

- Direct or indirect communication
- Synchronous or asynchronous communication
- Automatic or explicit buffering

There are several issues related with features like:

- Naming
- Synchronization
- Buffering

NESO ACADEMY

Naming

Processes that want to communicate must have a way to refer to each other. They can use either **direct** or **indirect** communication.

Under direct communication- Each process that wants to communicate must explicitly name the recipient or sender of the communication.

- send (P, message) - Send a message to process P.
- receive (Q, message) - Receive a message from process Q.

A communication link in this scheme has the following properties:

- A link is established automatically between every pair of processes that want to communicate. The processes need to know only each other's identity to communicate.
- A link is associated with exactly two processes.
- Between each pair of processes, there exists exactly one link.

This scheme exhibits **symmetry in addressing**; that is, both the sender process and the receiver process must name the other to communicate

Another variant of Direct Communication- Here, only the sender names the recipient; the recipient is not required to name the sender.

- send (P, message) - Send a message to process P.
- receive (id, message) - Receive a message from any process; *the variable id is set to the name of the process with which communication has taken place.*

This scheme employs **asymmetry in addressing**.

The disadvantage in both of these schemes (symmetric and asymmetric) is the limited modularity of the resulting process definitions. Changing the identifier of a process may necessitate examining all other process definitions.

With indirect communication:

The messages are sent to and received from **mailboxes**, or ports.

- A mailbox can be viewed abstractly as an object into which messages can be placed by processes and from which messages can be removed.
- Each mailbox has a unique identification.
- Two processes can communicate only if the processes have a shared mailbox

  - send (A, message) — Send a message to mailbox A.
  - receive (A, message) — Receive a message from mailbox A.

NESO ACADEMY

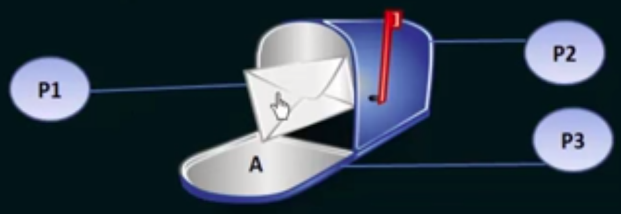
Message Passing Systems (Part 2)

A communication link in this scheme has the following properties:

- A link is established between a pair of processes only if both members of the pair have a shared mailbox.
- A link may be associated with more than two processes.
- Between each pair of communicating processes, there may be a number of different links, with each link corresponding to one mailbox.

15:22 / 22:59

Now suppose that processes P1, P2, and P3 all share mailbox A



Process P1 sends a message to A, while both P2 and P3 execute a receive() from A. **Which process will receive the message sent by P1?**

NESO ACADEMY

Process P1 sends a message to A, while both P2 and P3 execute a receive() from A. **Which process will receive the message sent by P1?**

The answer depends on which of the following methods we choose:

- Allow a link to be associated with two processes at most.
- Allow at most one process at a time to execute a receive () operation.
- Allow the system to select arbitrarily which process will receive the message (that is, either P2 or P3, but not both, will receive the message). The system also may define an algorithm for selecting which process will receive the message (that is, round robin where processes take turns receiving messages). The system may identify the receiver to the sender.

A **mailbox** may be **owned** either by a **process** or by the **operating system.**

NESO ACADEMY

Message passing may be either **blocking** or **nonblocking**— also known as **synchronous** and **asynchronous**.

Blocking send: The sending process is blocked until the message is received by the receiving process or by the mailbox.

Nonblocking send: The sending process sends the message and resumes operation.

Blocking receive: The receiver blocks until a message is available.

Nonblocking receive: The receiver retrieves either a valid message or a null.

NESO ACADEMY

Buffering

Whether communication is direct or indirect, messages exchanged by communicating processes reside in a temporary queue.
Basically, such queues can be implemented in three ways:

NESO ACADEMY

Zero capacity: The queue has a maximum length of zero; thus, the link cannot have any messages waiting in it. In this case, the sender must block until the recipient receives the message.

Bounded capacity: The queue has finite length n; thus, at most n messages can reside in it. If the queue is not full when a new message is sent, the message is placed in the queue and the sender can continue execution without waiting. The links capacity is finite, however. If the link is full, the sender must block until space is available in the queue.

Unbounded capacity: The queues length is potentially infinite; thus, any number of messages can wait in it. The sender never blocks.

NESO ACADEMY

Sockets

Used for communication in Client-Server Systems

- A socket is defined as an endpoint for communication.
- A pair of processes communicating over a network employ a pair of sockets—one for each process.
- A socket is identified by an IP address concatenated with a port number.
- The server waits for incoming client requests by listening to a specified port. Once a request is received, the server accepts a connection from the client socket to complete the connection.

NESO ACADEMY

- A socket is identified by an IP address concatenated with a port number.
- The server waits for incoming client requests by listening to a specified port. Once a request is received, the server accepts a connection from the client socket to complete the connection.
- Servers implementing specific services (such as telnet, ftp, and http) listen to well-known ports
  (a telnet server listens to port 23, an ftp server listens to port 21, and a web, or http, server listens to port 80).
- All ports below 1024 are considered well known; we can use them to implement standard services

NESO ACADEMY