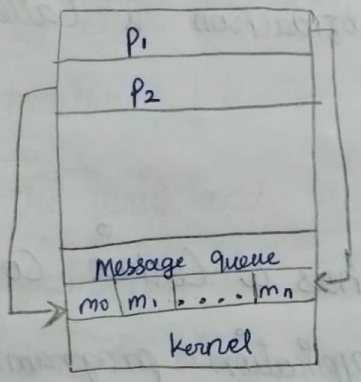
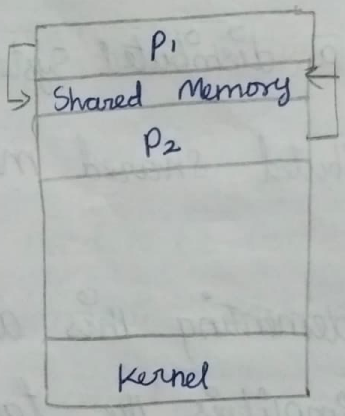
DISTRIBUTED COMPUTING

UINT-1

1. MESSAGE PASSING Vs SHARED MEMORY:



[Message Passing]



[Shared Memory].

* Shared memory Systems are those in which there is a (common) shared address space throughout the System.

* Communication among processors takes place via shared data variables and control variables for Synchronization among the processors.

* Semaphores and monitors that were originally designed for shared memory uniprocessors and multiprocessors are example of how synchronization

can be achieved in shared memory systems.

* All multicomputer systems that do not have a shared address pace provided by the underlying architecture and hardware necessarily communicate by message passing.

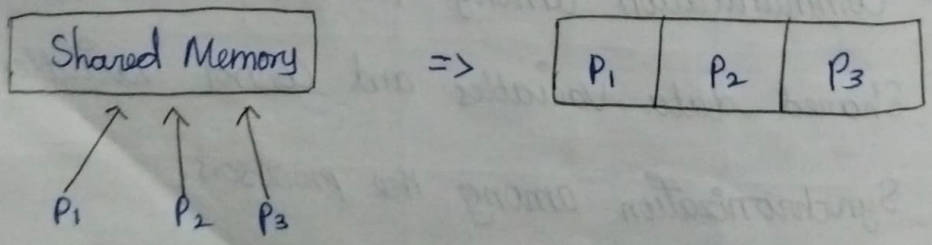
* For a distributed system, this abstraction is called distributed shared memory.

* Implementing this abstraction has a certain cost but it simplifies the task of the application programmer.

Emulating MP in SM :-

MP → Send, Receive

SM → Read, Write.



* The shared address space can be partitioned into disjoint parts, one part being assigned to each processor.

* "Send" and "Receive" operations can be implemented by writing and reading from the destination / sender's processor's address space, respectively.

* A separate location can be reserved as the mailbox for each ordered pair of processes.

* A P_i-P_j message passing can be emulated by a write by P_i to the mailbox and then a read by P_j from the mailbox.

* The write and read operations need to be controlled using synchronization primitives to inform the receiver/sender after the data has been sent/received.

Emulating shr SM in MP :-

SM —> Write, Read.

MP —> send, Receive.

* This involves the use of "send" and "receive" operations for "Write" and "Read" operations.

* Each shared location can be modeled as a separate process :

a. "Write" to a shared location is emulated by sending an update message to the corresponding owner process.

b. a "read" to a shared location is emulated

by sending query message to the owner process.

* The latencies (Time taken) involved in read and write operations may be high even when using shared memory emulation.

* An application can of course use a combination of shared memory and message passing.

* In a MIMD message passing multicomputer system, each "processor" may be a tightly coupled multiprocessor system with shared memory. Within the multiprocessor system, the processors communicate via shared memory. Between two computers, the communication is by message passing.

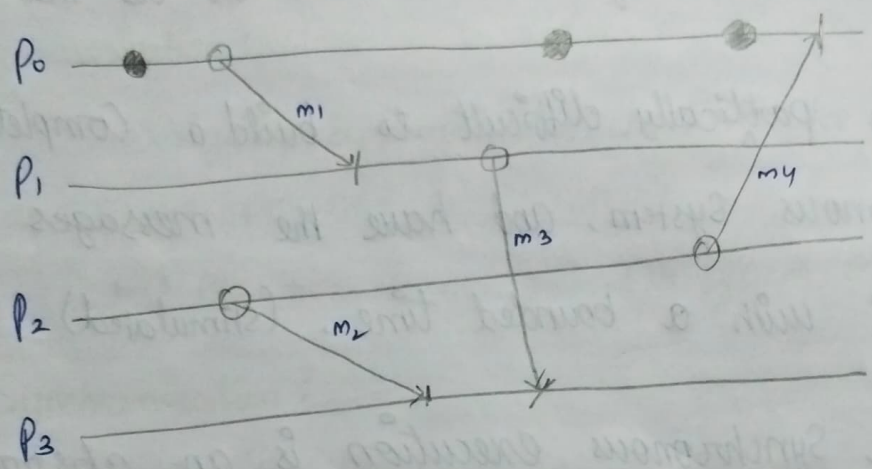
2] SYNCHRONOUS Vs ASYNCHRONOUS EXECUTION:

Asynchronous Execution:-

* There is no processor synchrony and there is no bound on the drift rate of processor clocks.

* Message delays (transmission + propagation times) are finite but unbounded.

* There is no upper bound on the time taken by a process to execute a step.



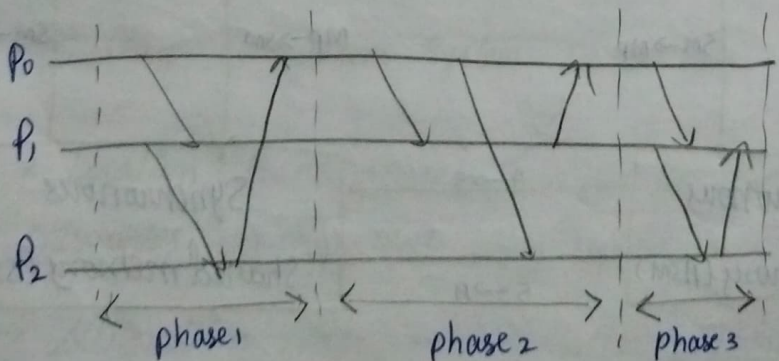
● internal event O send event | receive event

Synchronous Execution :-

* Processors are synchronized and the clock drift rate between any two processors is bounded.

* Message delivery (transmission + delivery) times are such that they occur in one logical step or round.

* There is a known upper bound on the time taken by a process to execute a step.

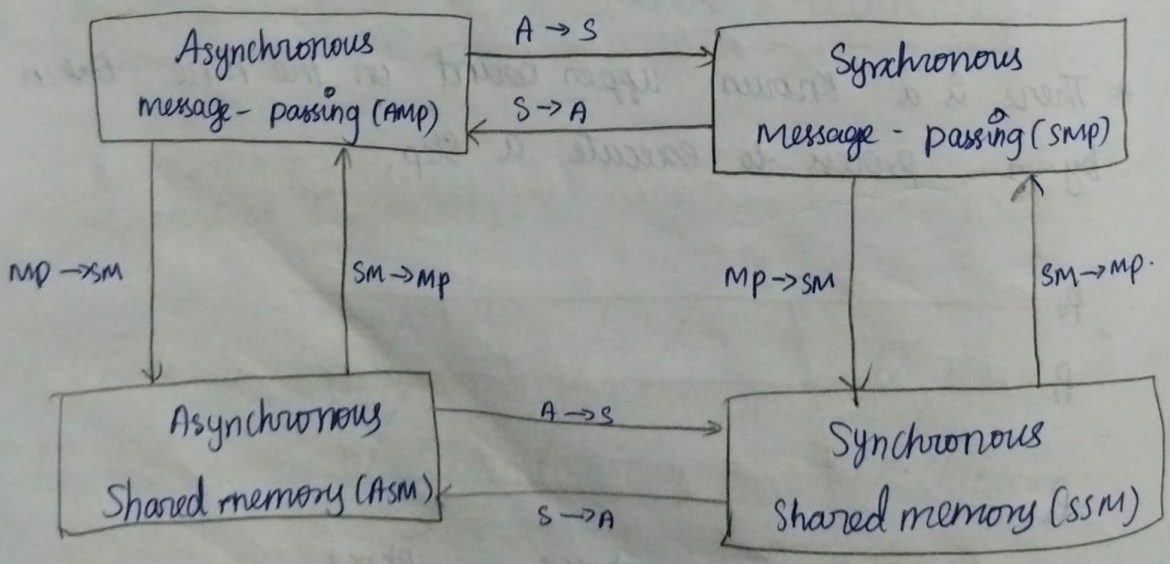


* It is easier to design and verify algorithms assuming synchronous executions because of the coordinated nature of the executions at all the processes.

* It is practically difficult to build a completely synchronous system, and have the messages delivered with a bounded time. (simulated)

* Thus, synchronous execution is an abstraction that needs to be provided to the programs. When implementing this abstraction, observe that the fewer the steps or "synchronizations" of the processors, the lower the delays and costs.

Emulating Systems:-



UNIT-1

1] MODEL OF DISTRIBUTED COMPUTATIONS :

DISTRIBUTED PROGRAM :-

* A Distributed program is composed of a set of n asynchronous processes $P_1, P_2, P_3 \cdots P_n$ that communicate by message passing over the communication network.

C_{ij} - Channel from P_i to P_j.

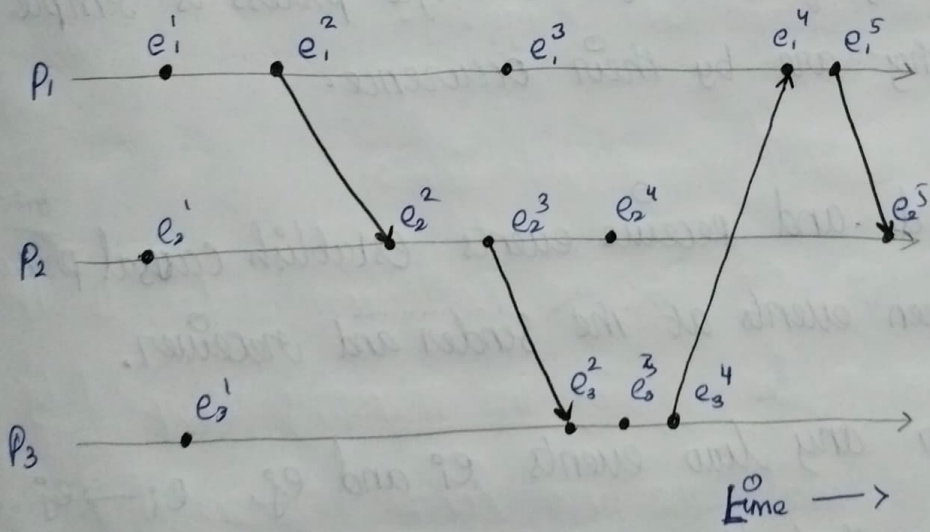
M_{ij} - Message sent by P_i to P_j.

* The processes do not share a global clock.

* The process execution and message transfer are asynchronous.

3] MODEL OF DISTRIBUTED EXECUTION:

* Process execution means sequential execution of the events.
* The events are atomic.
* The events are three types:-
  (i) Internal event.
  (ii) Send event.
  (iii) Receive event.



(i) Internal event:-

* An internal event affects only the process which is executing the event.

(ii) Send event:-

* A Send event changes the State of the process that Sent the message and the state of the channel on which the message is sent.

(iii) Receive Event :-

* A receive event changes the State of the process that receives the message and the State of the channel on which the message is received.

Casual precedence relation :

* Ordering of events for a single process is simple i.e. they are by their occurence.

* Send and receive events establish causal precedence (Ordering) between events at the sender and receiver.

* For any two events e_i and e_j, $e_i \rightarrow e_j$ denotes that the event e_i causally affects the event e_j.

Logical Vs physical concurrency :

* In distributed computation, two events are said to be logically concurrent if and only if they do not causally affect each other.

* physical concurrency denotes that the events occurs at the same instant in physical time.

3] GLOBAL STATE :

* The global state of a distributed system is the set of local states of all individual processes and the state of communication channels.

$$GS = \{ \bigcup_i LS_i , \bigcup_{i,j} SC_{ij} \}$$

Local states → $\bigcup_i LS_i$; States of channels → $\bigcup_{i,j} SC_{ij}$; Set of

* The state of a process is characterized by the state of its local memory and the context.

* The state of a channel is characterized by the set of messages in transit in the channel.

Consistent Global State :

* A global State is said to be Consistent if every message that is recorded as received is also recorded as sent.

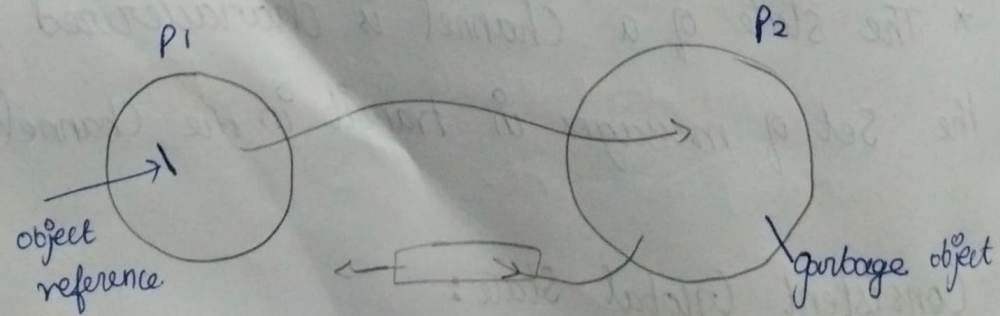
* A message cannot be received if it was Sent.

Requirements of Global States:

(i) Distributed garbage collection.
(ii) Distributed deadlock detection.
(iii) Distributed termination detection.
(iv) Distributed debugging.

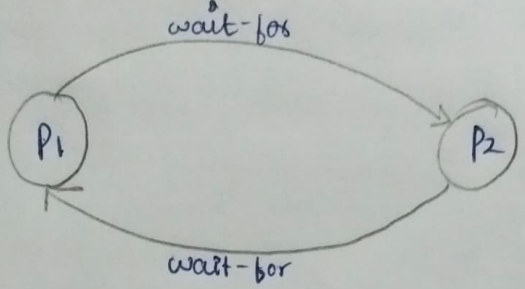
(i) Distributed garbage Collection :-

* Distributed garbage collection refers to the process of recollecting the memory occupied by the objects that are no longer referenced in a distributed system.



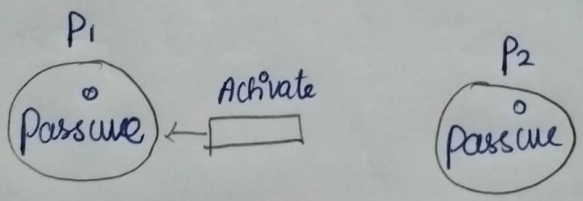
(iii) Distributed deadlock detection :-

* Distributed deadlock detection refers to the process of identifying deadlocks in a distributed system.



(iii) Distributed termination detection :-

* Distributed termination refers to the process of determining when a distributed computation has terminated.



(iv) Distributed debugging :-

* Distributed debugging refers to the process of identifying and rectifying the errors in a distributed system.