DISTRIBUTED COMPUTING.

UNIT - 2.

1] MESSAGE ORDERING PARADIGMS:

* The message ordering means the order of delivering the messages to the intended recipients.

* The common message order themes are FIFO, NON FIFO, causual order.

Notations:-

msg - m_i

send event - S_i

Receive event - R_i

$a \sim b \Rightarrow a \& b$ @ same process

T => all send-rec pairs.

$s^1 \propto s^2 \Rightarrow s^1_i$ causally dependent on s^2

1. NON-FIFO EXECUTIONS:-

* It is also called as Asynchronous execution.

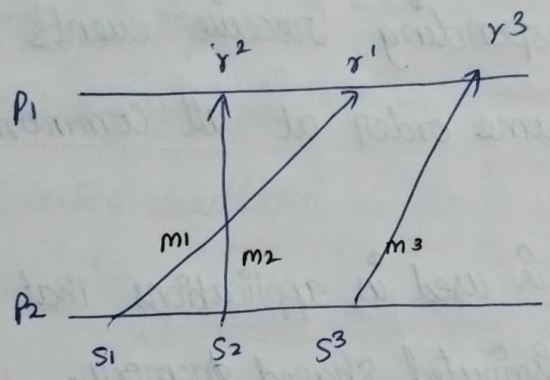
* Asynchronous execution is an execution for which the causality relation is a partial order.

* There cannot be any causal relationship between events in asynchronous execution.

* The messages can be delivered in any order.

* Multiple paths may exist between the two end points of the logical link.

* All physical links obey FIFO rule.

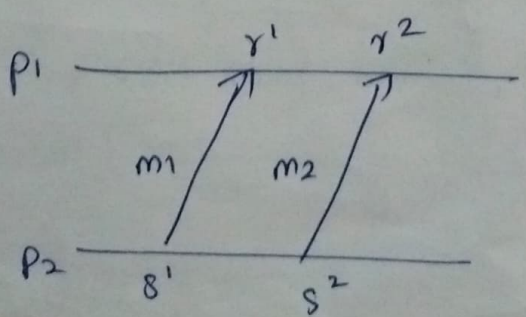


2. FIFO EXECUTIONS:-

* A FIFO execution is an A-execution in which, for all (s,r) and $(s',r') \in T$. $(s \sim s'$ and $r \sim r'$ and $s \lessdot s') \Rightarrow r \lessdot r'$

* To implement FIFO over non-FIFO link, use sequence number and connection id for each message.

* Receiver uses buffer to order messages as per the Senders sequence numbers.

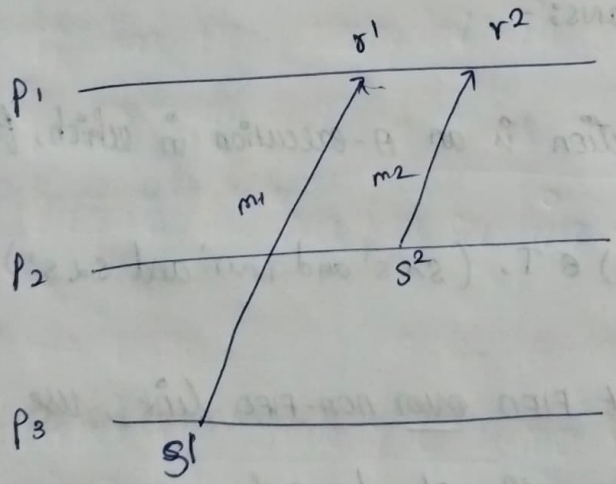


3. CASUAL ORDER :-

* CO execution is an A-execution in which, for all (s, r) and $(s', r') \in T$, $(r \sim r'$ and $s \prec s') \Rightarrow r \prec r'$

* Two send events s and s' are related by causality ordering, then a causality ordered execution requires that their corresponding receive events r and r' occur in the same order at all common destinations.

* Casual order is used in applications that update shared data, distributed shared memory.



8] SNAPSHOT ALGORITHMS FOR FIFO CHANNELS :

* A Snapshot captures the local states of each process along with the state of each Communication channel.

Snapshots are required to :-

* Checkpointing
* Collecting garbage
* Detecting deadlocks.
* Debugging.

Chandy - Lamport algorithm :

* The algorithm will record a global snapshot for each process channel.
* The Chandy - Lamport algorithm uses a control message, called a marker.
* After a site has recorded its snapshot, it sends a marker along all of its outgoing channels before sending out any more messages.
* Any process may initiate the snapshot (send 'Marker').

Algorithm :-

Marker Sending rule for process P_i.

1. Process P_i records its state.
2. For each outgoing channel 'c' on which a marker has not been sent, P_i sends a marker along c before P_i sends further messages along c.

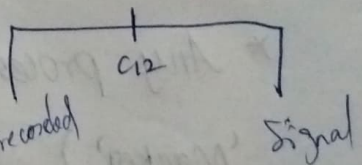
Marker receiving rule for process P_j.

→ On receiving a marker along channel c.

If P_j has not recorded its state then
Record the state of c as empty set
Execute the "marker sending rule".

else

Record the state of c as the set of messages received along c after P_j's state was recorded and before P_j received the marker along c.



Correctness :-

m_1 m_2 ⓧ m_3
msg ⓧ

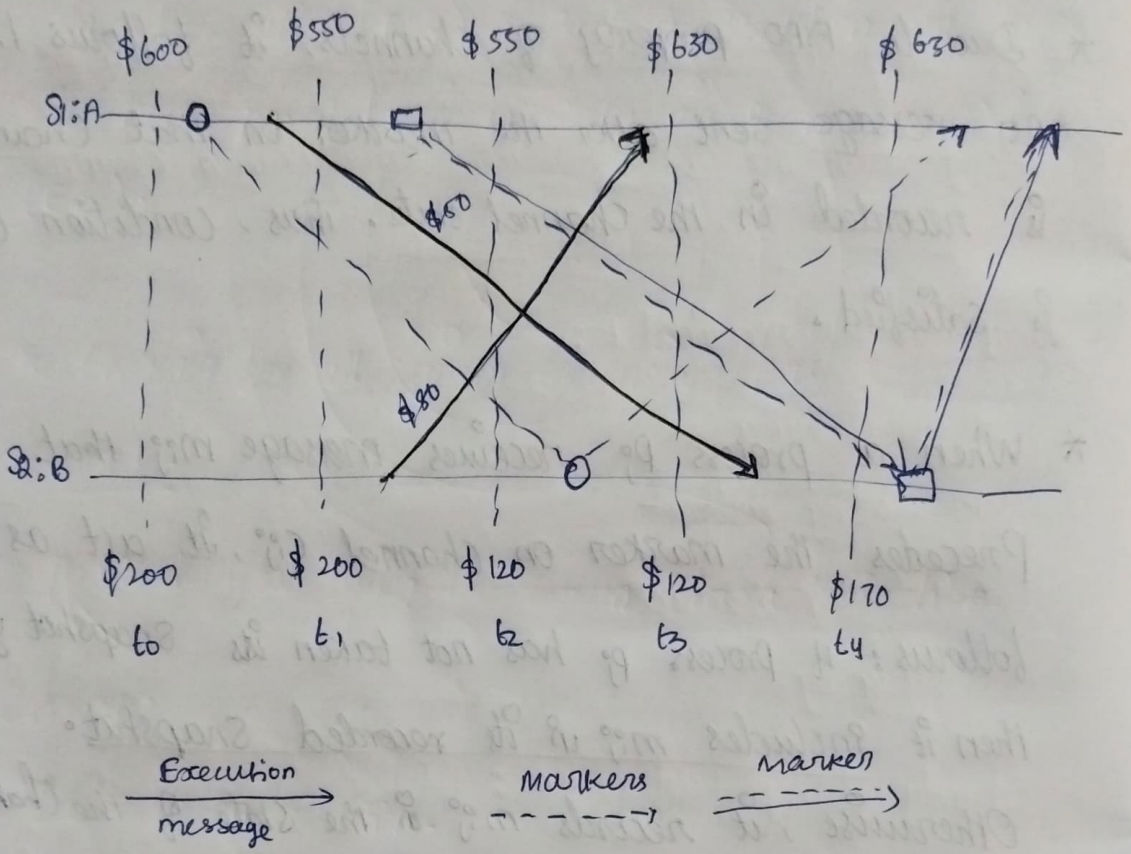
* Due to FIFO property of channels, it follows that no message sent after the marker on that channel is recorded in the channel state. Thus, condition C_2 is satisfied.

* When a process p_j receives message m_{ij} that precedes the marker on channel C_{ij}, it act as follows: If process p_j has not taken its snapshot yet, then it includes m_{ij} in its recorded snapshot. Otherwise, it records m_{ij} in the state of the channel C_{ij}. Thus, condition C_1 is satisfied.

Complexity :-

* The recording part of a single instance of the algorithm requires $O(e)$ messages and $O(d)$ time, where e is the number of edges and d is the diameter of the network.

Properties of the recorded global State :-



→ Timing diagram of two possible executions of the banking examples.

□, 1. A = $500
B = $170
C12 = $0
C21 = $80
$800

○, 2. A = $600
B = $120
C12 = $0
C21 = $80
$800

3] ASYNCHRONOUS EXECUTION WITH SYNCHRONOUS COMMUNICATION:

* A distributed algorithm designed to run correctly on asynchronous systems may not run correctly on synchronous systems.

* An algorithm that runs on an asynchronous system may deadlock on a synchronous system.

Process A	Process B
Send (B)	Send (A)
Receive (B)	Receive (A)

Realizable with Synchronous Communication (RSC).

* A-execution can be realized under Synchronous communication is called Realizable with Synchronous Communication (RSC).

RSC EXECUTIONS :-

Non-Separated linear extension of $(E, \prec)$

∴ A linear Extension of $(E, \prec)$ such that for each pair (s, r) the interval $\{x \in E / s \prec x \prec r\}$ is empty.

Rsc execution

An A-Execution (E, α) is an RSA execution iff there exist non-separated linear extension of the partial order (E, α).

* Checking for all linear extensions has exponential cost!.

* Practical test using the crown characterization.

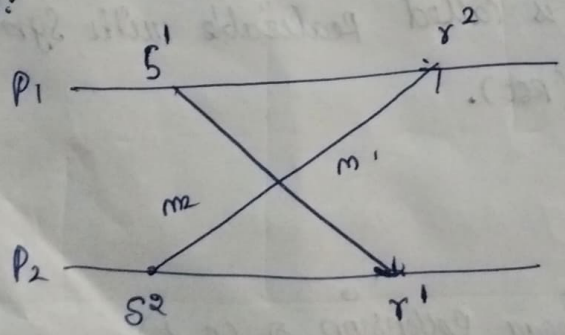
CROWN:

Let E be an execution.

A Crown of size K in E $((s^i, r^i), i \in \{0, \ldots k-1\})$ of pairs of corresponding send and receive events such that:

$s^0 < r^1, s^1 < r^2, \ldots s^{k-2} < r^{k-1}, s^{k-1} < r^0$

Example:



P_3 ———————

(a)

* Fig (a) : The crown is $\{(s^1, r^1), (s^2, r^2)\}$ as we have $s^1 \prec r^2$ and $s^2 \prec r^1$. This execution represents the program execution

* Cyclic dependencies may exist in a crown.

* The crown criterion states that an A-computation is RSC, i.e., it can be realized on a system with synchronous communication if and only if it contains no crown.

Hierarchy of (msg) ordering paradigms:

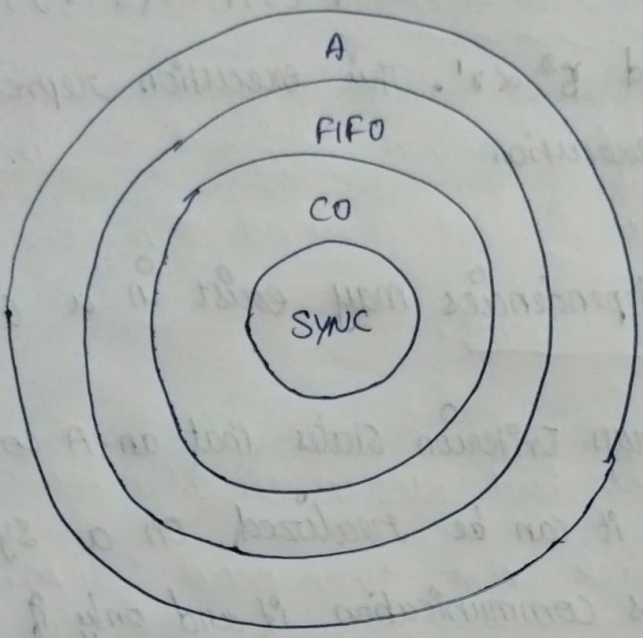
* For an A-execution A is RSC if and only if A is an S-execution.

* RSC $\subset$ CO $\subset$ FIFO $\subset$ A.

* The degree of concurrency is most in A and least in SYNC.

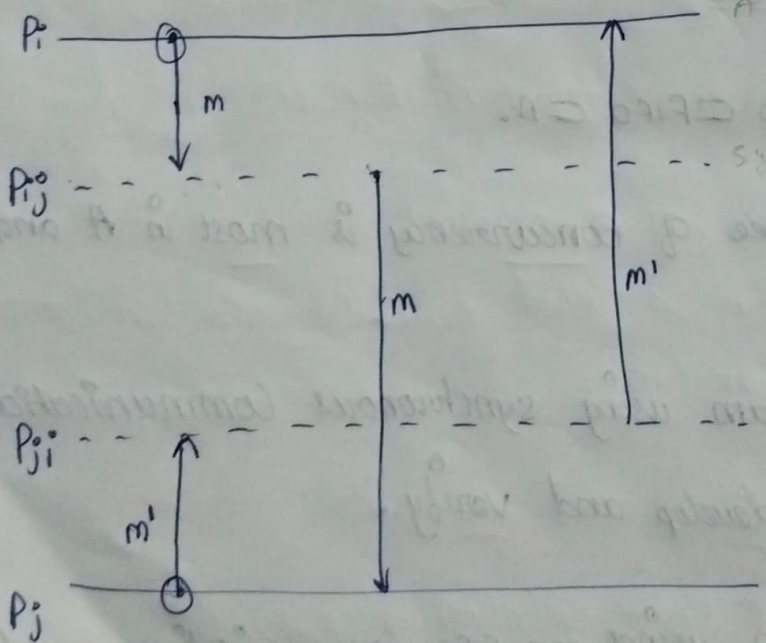
* A program using synchronous communication is easiest to develop and verify.

* A program using non-FIFO communication, resulting is an A execution, is hardest to design and verify.



Hierarchy of the communication model.

Simulation - Asynchronous program on Synchronous systems. (How to make non RSC to RSC).



* Each channel C_{ij} is modeled by a Control process P_{ij} that simulates the channel buffer.

* An asynchronous communication from i to j becomes a synchronous communication from i to P_{ij} followed by synchronous communication from P_{ij} to j.