Assignment No. 3

Q.1 What are logical clock ? Why do we need them ? Write algorithm for ordering event using Lamports logical clock.

→ Logical clock:

a) In many applications, all machines should agree on same time although this time does not agree with UTC time or real time. In this case, internal consistency of clocks is essential.

b) many algorithms work based on internal consistency of clocks and not with real time.

c) For these algorithms, clocks are said to be logical clocks.

Lamports logical clocks:

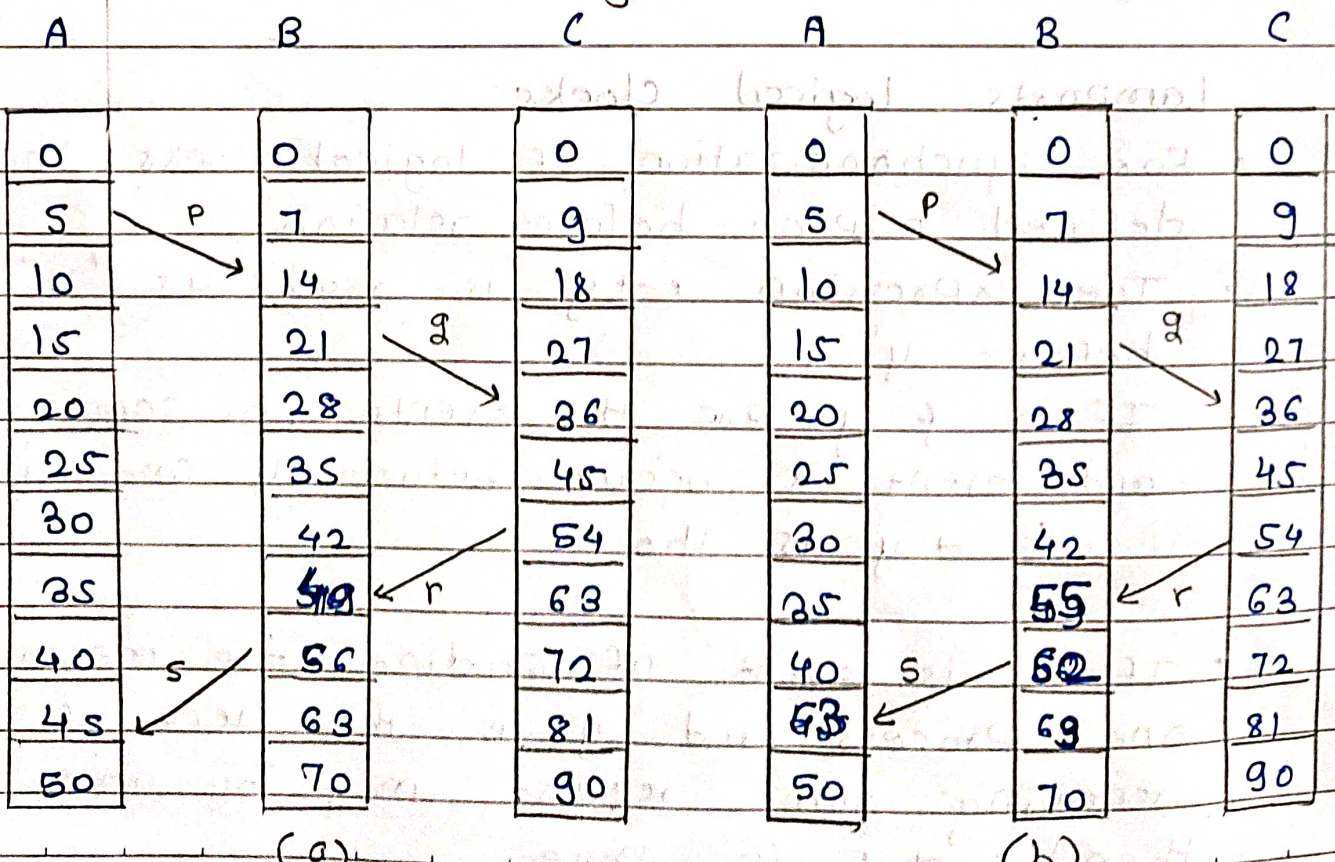
- For synchronization of logical clocks, lamport defined happen-before relation.
- The expression $x \rightarrow y$ is read as "x" happens before "y"
- If x & y are the events in same process and event x occur before y and the $x \rightarrow y$ is the
- If x is event of sending the message by one process and y is the event of receiving same request msg by other process then $x \rightarrow y$ is true.

If x is event of sending the message by one process and y is the event of receiving same message by other process then $x \rightarrow y$ is true.

Pratically, msg takes non-zero time to deliver to other process.

If $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$
It means happen before relation is transitive.

If any two processes does not exchange msg directly or indirectly, then $x \rightarrow y$ & $y \rightarrow x$ where x and y are the events that occurs in these processes. In this case, events are said to be concurrent.
Consider the following example,

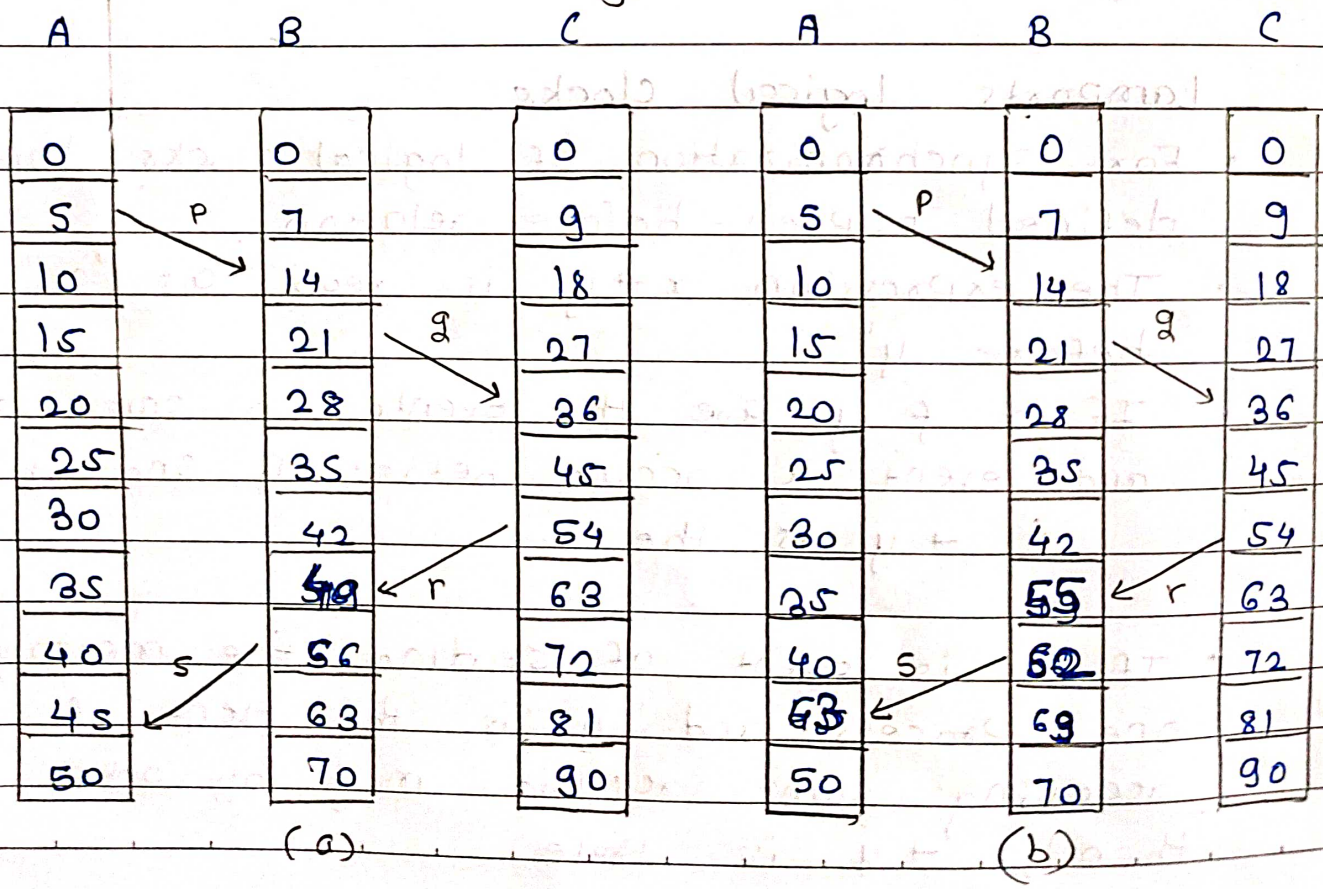


If x is event of sending the message by one process and y is the event of receiving same message by other process then $x \rightarrow y$ is true.

Pratically, msg takes non-zero time to deliver to other process.

If $x \rightarrow y$ and $y \rightarrow z$ then $x \rightarrow z$
It means happen before relation is transitive.

If any two processes does not exchange msg directly or indirectly, then $x \rightarrow y$ & $y \rightarrow x$ where x and y are the events that occurs in these processes. In this case, events are said to be concurrent.
Consider the following example,



- This incoming msg contains its own process at this point, this msg gets connected converted to co-ordinator msg. once again this co-ordinator msg is circulated along the ring to inform to processes about new co-ordinator member of the ring.
  of course the highest process no. is choosen from the list for new co-ordinator by process which has started the election.

- The process 5 notices crash of co-ordinator which has initially process 7. It then sends election msg to processes 6.
  This message contains no. of process 5. As process-7 is crashed, process 6 forward this msg to its new process 0 and append its no. in the source list. In this way msg is received by all the processes in ring.

Eventually, msg arrives at process 5 which had initiated the election. Highest no. in this list is 6. So msg again circulated in previous manner to inform all the processes that process 6 is now new co-ordinator.

Q.3 Gossip Based Co-ordination:

→

1) There are some examples in which gossiping exist.

* Gossip-based overlay construction:

- By sensible choosing and exchanging the entries from partial views, it can be possible to construct and maintain specific topologies of the overlay networks.

It can be acheived with two-layer approach.

- The lowest layer establishes an unstructured peer-to-peer system in which node pratically exchange the entries of their partial views in order to offer a peer sampling service. The partial view should be filled with entries that refers to the randomaly selected live nodes.

- The lowest layer sends its partial view to highest layer where furthe selection of entries occurs. In this way, a second list of neighbours is formed corresponding to the topology.

- At time 5, process A sends msg 'P to Process B and it is received at 14. process B will conclude that msg has taken 9 ticks to travel from process A to process B, if message 9 from Process B to C.

- process B sends msg at time 56 and it is received by process A at time 45.
similarly, Process C sends msg at time 54 and it is recived by process A at time 49. This should ~~be~~ not happenened and it should be prevented. It shows sending time is larger than receiving time.

- Msg x from Process C leaves at time 54. As per happen before relation, it should reach at Process B at time 55 later.

- If two events occur at the same time in different Process then it should be separated by decimal point.

a) If $x \rightarrow y$ in the same process than
$T(x) < T(y)$

b) If x is event of sending the msg y is the even of receiving the msg then $T(x) < T(y)$

c) For all distingwishing events x & y
$T(x) \neq T(y)$.

Q.2 List and explain any one election algorithm.

→

Election algorithms - Bully algorithm
Ring algorithm

Ring algorithm :

- The ~~rign~~ ring algorithm does not use token and processes are physically or logically ordered to ring.
- Each and every process has its sucessor.
- Every process khows that who its is its sucessor.
- When any process notices that co-ordinator is crashed ; it builds the election msg and sends to its successor.
- This msg contains the process no. of sending process.
- If sucessor is down then process sends msg to the next process along the ring. if it finds many crashed process in seq.
- In this manner, the receiver of election msg also forwards the same msg to its sucessor or by appending its own no. in msg.
- This incoming msg contains its own process no. along with process no. of all the processes that had received this msg.

Q.4 centralized mutual exclusion

→ In environment where multiple processes are running, processes shared resources. When Process must read or update shared data structures, it enters, the critical in order to achieve mutual exclusion.

This ensures that, no matter other process enters the critical in order to use shared dat structure when a process is already using it.

Centralized mutual exclusion:

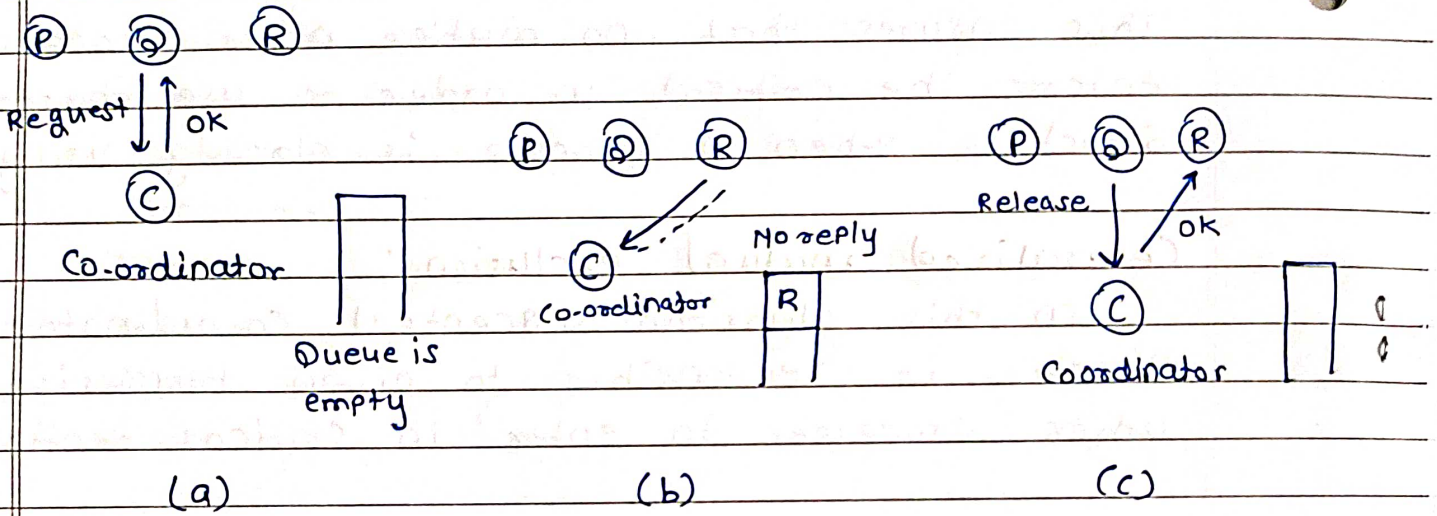
In this algorithm, a central co-ordinator process is responsible to given permission to other processes to enter in critical section (cs)

Co-ordinator process keeps track of which Cs is busy or currently in execution. process which wants to enter in cs sends req msg to Co-ordinator stanoting stating the name of cs.

If cs is free then co-ordinator sends ok msg to requesting process to grant permission to enter in stated cs in msg. If in stated cs, already other process is executing then it puts the req message in its queue.

When currently executing process in Cs exists

it Sends release message to Co-ordinator Process. As a result, Co-ordinator knows that Cs is now free and it takes request from the queue to grant Permission to enter in cs. If requesting process is still blocked, it unblocks and enters in cs.



(a) (b) (c)

- Process C is co-ordinator and currently no process is execting in CS. process Q sends Request msg to co-ordinator and it permits process Q to enter in cs by replying with ok msg.
- process R also send req msg to co-ordinator to get permission to enter in same cs.

Limitation of the algorithm is that Co-ordinator may crash.