End Sem ~~Exam~~ Exams 2020-21

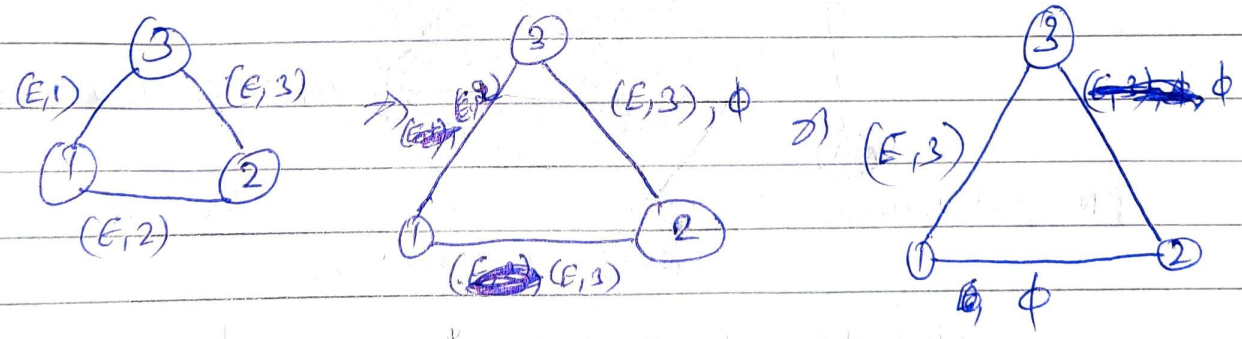
IIIT Nagpur

Name ⇒ Himanshu Gupta
Roll No ⇒ BT17CSE032
Course ⇒ DCS , code ⇒
Sem. & Year ⇒ 7th sem, final year
Branch ⇒ CSE
Date ⇒ 25/11/2020

Himgupta

Ans ⇒ 4) Lenann & Chang-Roberts Algorithm :-

When a node N_i receiving an election message has a greater id and is already a participant, then it sends an empty message
~~Or a node~~ Or a node N_j receiving an empty message or a message with greater id that is already a participant, then forward an empty message
Eos



Disadvantages

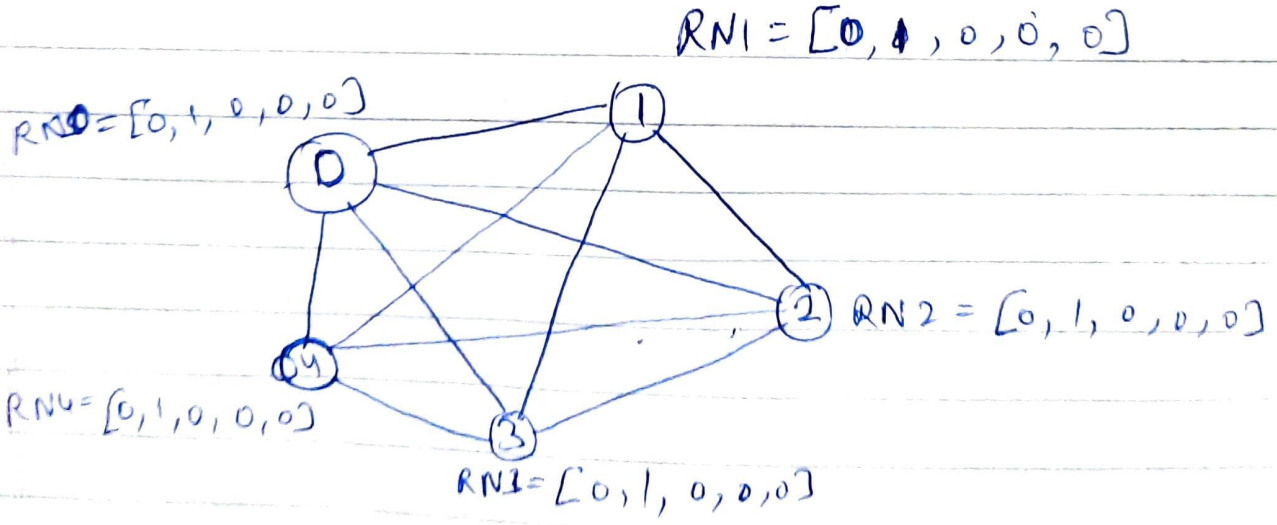
(1) It uses unidirectional communication
(2) High worst case complexity = $O(n^2)$, if n = high
(3) Uses asynchronous system
(4) No ~~broad~~ Broadcasting, each message goes in network from one process to another.

(5) It requires ring network in which message can only travel clockwise.

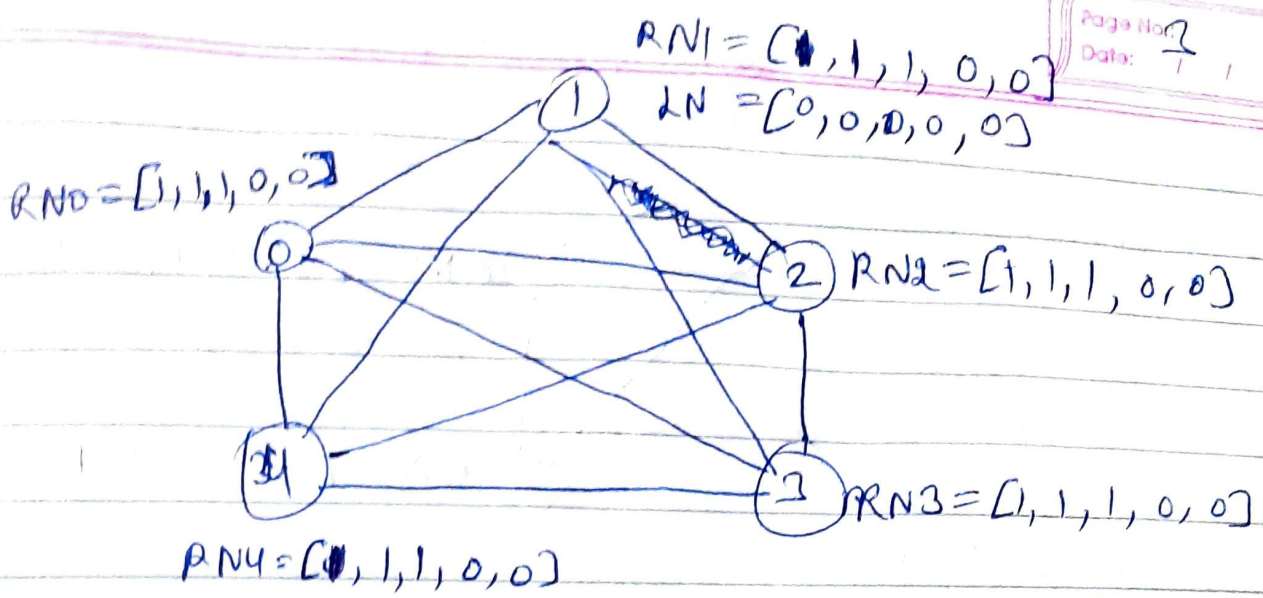
(6) If somehow, the node with greatest id crashed then we need to run this algorithm again that result in time overhead.

The complexity of Lelann's algorithm can be reduced if we can setup bidirectional communication between nodes. ~~Or if the Algorithm is initiated, only by greater id node and all~~

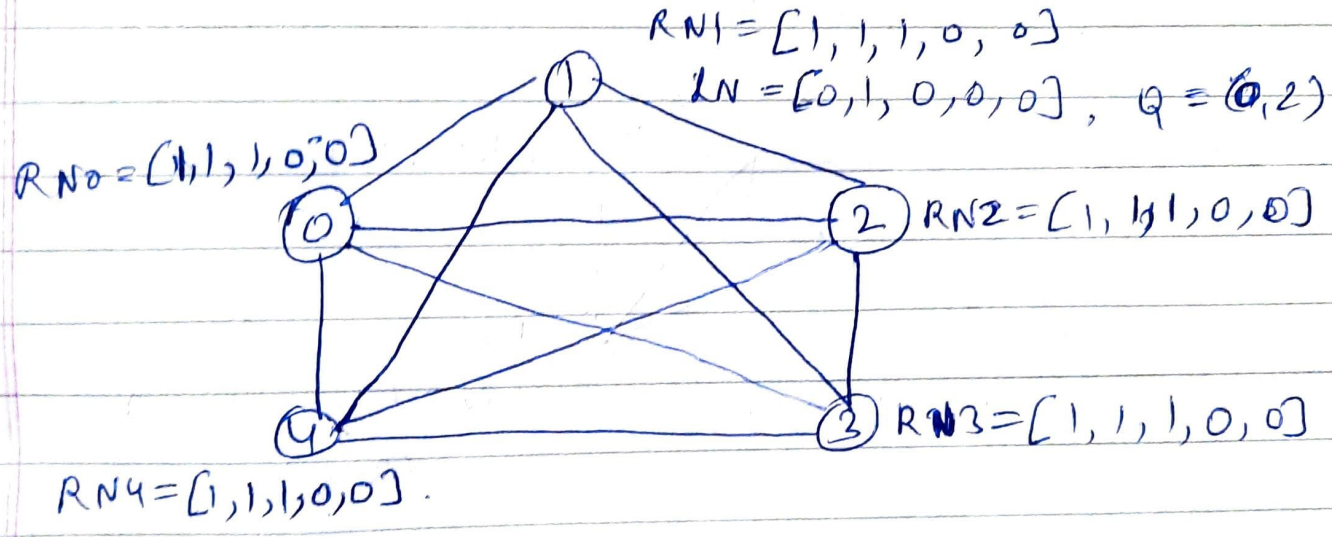
Ques 3)



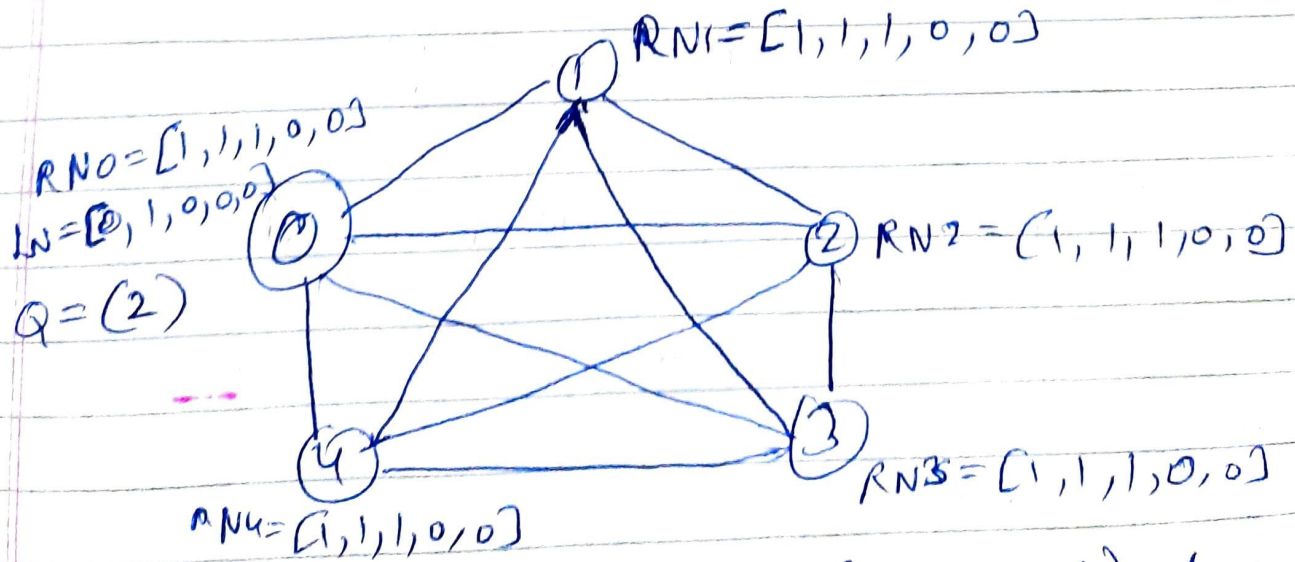
initial state :-> node has sent a request to all and grabbed the token.



node 0 & 2 request to enter CS



Node 1 prepares to exit CS

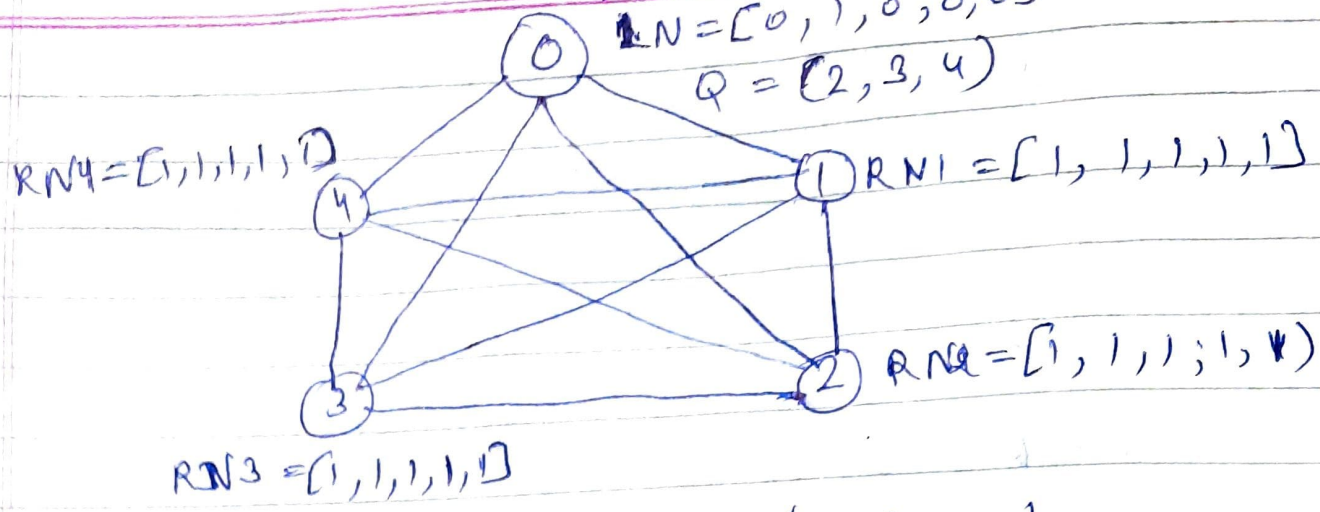


node 1 passed toked (Q and last) to node 0

~~RNO = [1, 1, 1, 1, 1]~~

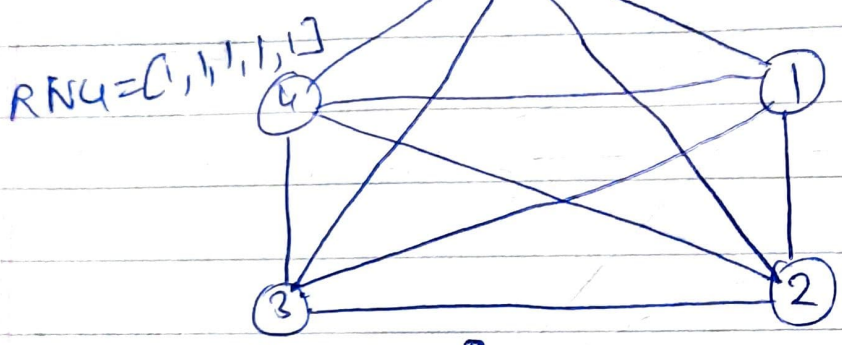
LN = [0, 1, 0, 0, 0]

Q = (2, 3, 4)



node 3 and 4 sent request

RNO = [1, 1, 1, 1, 1]

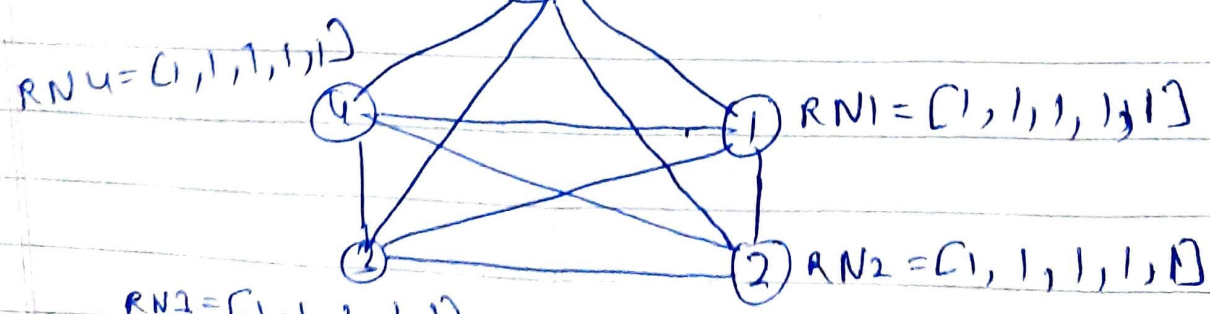


LN = [1, 1, 0, 0, 0]

Q = (3, 4)

Node 0 exit CS & sent token to node 2

RNO = [1, 1, 1, 1, 1]

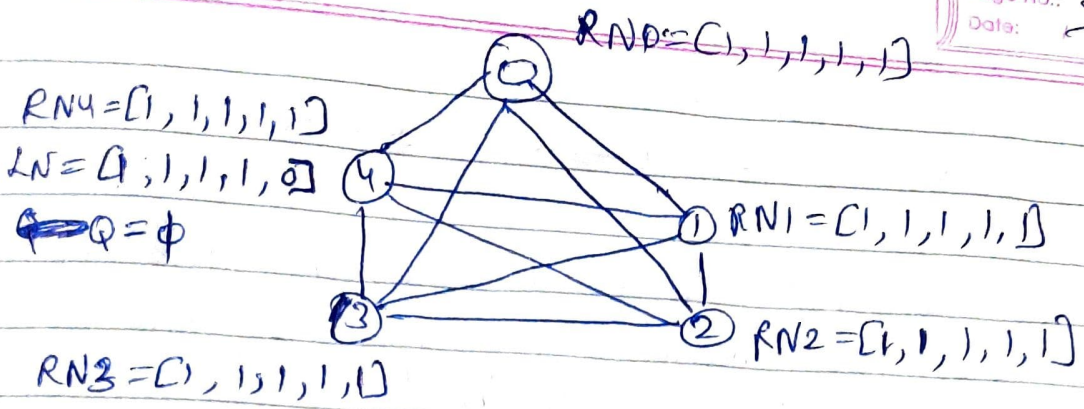


RN2 = [1, 1, 1, 1, 1]

LN = [1, 1, 1, 0, 0]

Q = (4)

Node 2 sends token to Node 3

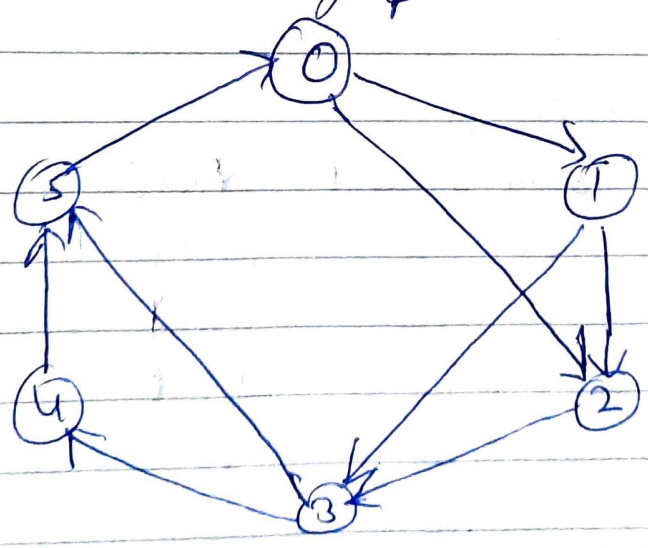


Node 4 in CS

Best case complexity = 0 (for node 1, as it does not wait for token)

Worst case complexity = N=5 (for node 4)

Ans 6) For given wait-for-graph,



Using the Chandy-Misra-Haas Algorithm for detecting a communication deadlock (OR model), it is clear that this WFG has communication deadlock. Because the necessary condition for deadlock is the presence of cycle.
And for OR model, sufficient condition is presence of 'knot'.

Since this WFG has cycle
$0 \to 1 \to 2 \to 3 \to 4 \to 5 \to 0$

Also, it has not sub-group which does not end up in a cycle, it means it has a knot.
So, there will be deadlock.

Deadlock detection Process for OR Model

Given Node 0 → initiator
Probe format → Query(i, j, k)
where i = initiator (constant)
j = sender (edge node)
k = receiver (edge node of the other site)

and Reply (i, j, k)

Diffusion Computation

→ Query(0, 0, 1), from 0 to 1, Num0 = 1, parent(1) = 0
→ Query(0, 0, 2), from 0 to 2, Num0 = 2, parent(2) = 0
→ Query(0, 1, 2), from 1 to 2, Num1 = 1, parent(2) = 1
→ Query(0, 1, 3), from 1 to 3, Num1 = 2, parent(3) = 1
→ Query(0, 2, 3), 2 to 3, Num2 = 1, Parent(3) = 2
→ Query(0, 2, 4), 2 to 4, Num2 = 2, parent(4) = 2
→ Query(0, 3, 4), 3 to 4, Num3 = 1, parent(4) = 3
→ Query(0, 3, 5), 3 to 5, Num3 = 2, parent(5) = 3
→ Query(0, 4, 5), 4 to 5, Num4 = 1, parent(5) = 4
→ Query(0, 5, 0), Num5 = 1, ~~parent(0~~
i j k

↳ Here i = k
Now, Reply (i, i, j) set to j

when k receives Reply(i, j, k) from j, then.
Numk = 1
As soon as Numk = 0 ⇒ Deadlock

→ Reply(0, 0, 5), from 0 to 5, Num 5 = 1 − 1 = 0
→ Reply(0, 5, 4), —, Num 4 = 0
→ — (0, 5, 3), Num 3 = 1
→ (0, 4, 3), Num 3 = 0
→ (0, 4, 2), Num 2 = 1
→ (0, 3, 2), Num 2 = 0
(0, 3, 1), Num 1 = 1
(0, 2, 1), Num 1 = 0
(0, 2, 0), Num 0 = 1
→ Reply(0, 1, 0), from 1 to 0, Num 0 = 0 → Deadlock

Ans. 2) Correctness of Ricart-Agerwala Algorithm
Proof by contradication
→ suppose two nodes N_i & N_j are executing the CS concurrently and Node N_i's request has ~~high~~ smaller timestamp than N_j's request
→ Clearly, N_i receive N_j's request after inserting its own request
→ Similary for N_j's request set.
→ Thus, N_j can concurrently execute the CS with N_i only if N_i returns a reply to N_j (for N_j's request) before N_j exits the CS.
→ However, this is impossible because N_j's request ~~also~~ has lower priority.

Therefore, Ricart - Agarwala Algorithm achives mutual exclusion.

In the Ricart - Agarwala Algorithm, for every requesting pairs of nodes, the node with ~~have~~ smaller timestamp request will always defer than the request of higher timestamp. So, smaller timestamp's request succeded first.

Performance :-

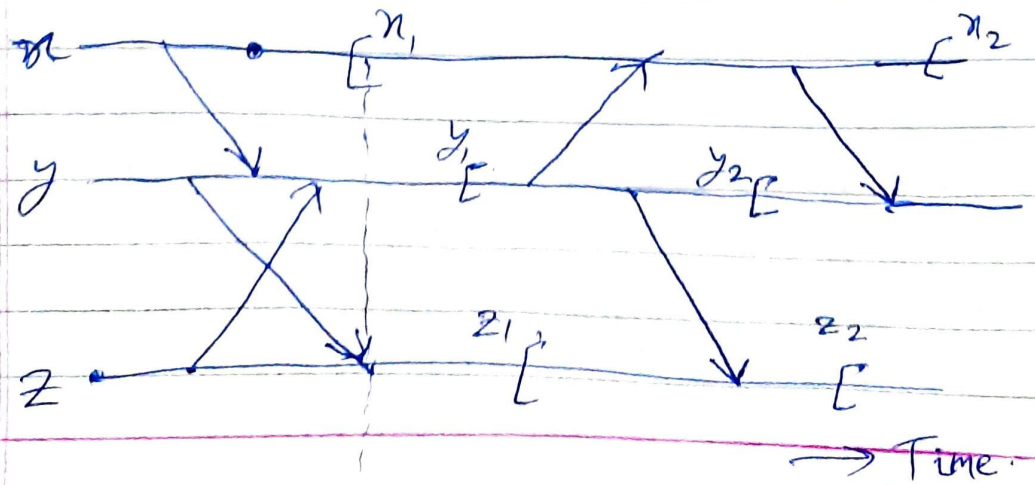
For each CS execution, Ricart - Agarwala Algo requires (N-1) Request messages and (N-1) Reply messages.

Its complexity can be further ~~increase~~ reduce by eliminating the part of Reply, message.

Ans 5) Strongly Consitent Checkpoints :-

It is the establishment of set of local check-points such that no information flow takes place during the interval spanned by the checkpoints.

Strongly consistent set of checkpoints with diagram



In the above diagram x, y, z are strongly consistent set of checkpoint which are dotted line work that is in the commit form. If we have to rollback set of process before the recovery point x_1, y_1, z_1, then no message is lost and no message is tracked as Orphan message. But (x_2, y_2, z_2) are not strongly consistent set of checkpoint because if the transaction fail at m then it roll-back to its initially point at y_2 then message m will be treated as lost message consistent set of checkpoints.

Each message that is received in a checkpoint should also be ~~recovered~~ recorded as sent in another checkpoint. It is similar to consistent global state.

Suppose y fails after receiving message m. If y restart from checkpoint y_2, the m is lost. The set (x_2, y_2, z_2) are reffered as consistent set of checkpoints. It is similar to a consistent global state, such that each message is recorded. Therefore system set of checkpoint have to deal with lost message during roll back recovery. While system which having strongly consistent set of checkpoints do not have to deal with lost message. They experience delays during checkpointing process as process can't exchange message while checkpoint in progress.

1) Given Assumptions are
① N is large
② Nodes are heterogeneous
③ multiple nodes can work simultaneously.
④ Different Nodes have different OS.

Design Aspects
① Heterogenity :- For Heterogenity there should be some kind of interpolability between components of network.
There should be common way of communication.

② Scalability
for n to be large, the design should be able to add new nodes for computation.

③ Concurrency :- To handle deadlock, we should consider implementing mutual exclusion Algo, and deadlock detection Algo.

④ for different OS's.
↳ have access transparancy.
↳ same system call implementation