ALM-3

2300030525

1. AND vs OR Deadlock Models

AND Model: A process is blocked until all requested resources are granted.

Example: P1 waits for R1 and R2 → both must be free.

OR Model: A process proceeds if any one of the requested resources is granted.

Example: P1 waits for R1 or R2 → if either becomes available, P1 proceeds.

2. Lamport's Algorithm for Mutual Exclusion

Uses logical clocks and message timestamps.

Steps:

Send REQUEST with timestamp to all.

Wait for REPLY from all processes and ensure your request is earliest in your queue.

Enter Critical Section.

Send RELEASE after exiting.

Ensures: FIFO order of access, no two processes enter CS simultaneously.

3. Purpose of Distributed Snapshots in Termination Detection

Used to capture a consistent global state of a distributed system.

Helps detect if all processes have finished and no messages are in transit.

Foundation for Chandy-Lamport algorithm used in termination and checkpointing.

4. Chandy-Misra-Haas (AND model) Deadlock Detection
Given:

P1 → P2 → P3 → P1 (wait-for cycle)

Probe Messages:

P1 sends <P1, P1, P2>

P2 forwards <P1, P2, P3>

P3 forwards <P1, P3, P1>

Since init = P1 and dest = P1 in final probe → Deadlock Detected.

5. Lost ACK in Spanning-Tree Termination Detection
If ACK is lost, the parent waits indefinitely → termination is not detected.

Mechanisms:

Timeout and retransmission.

Reliable transport (TCP-like).

Rechecking with child nodes.

6. Ricart-Agrawala Algorithm (3 Processes: P1, P2, P3)

All send REQUESTs with timestamps.

A process replies only if it is not in CS or its own request has lower priority.

Example:

P1, P2, P3 send REQUESTs.

P1 gets replies from P2 & P3 (earliest timestamp) → enters CS.

Others wait → conflict resolved by timestamp ordering.

7. AND vs OR Model (with examples)
AND: All resources needed simultaneously.

P1 waits for R1 and R2 → Deadlock if both are held by others.

OR: Any one of multiple resources suffices.

P1 waits for R1 or R2 → can proceed as soon as one is available.

AND → more prone to deadlocks.

8. Ricart-Agrawala with Network Delays
Scenario: P2 gets P3's request before P1's (P1 has earlier timestamp).

P2 still waits for P1's request before replying to P3.

Ordering is based on timestamps, not arrival time → ensures consistency.

9. Lamport vs Ricart-Agrawala (Complexity)

Algorithm Messages/CS Synchronization
Lamport 3(N-1) (Request + Ack + Release) Logical clocks
Ricart-Agrawala 2(N-1) (Request + Reply) Logical clocks

Ricart-Agrawala is more efficient in message count.

10. Suzuki-Kasami's Token Algorithm (All Request Simultaneously)
All nodes request token.

Node A (initial holder) checks token queue.

It serves the next process with highest sequence number.

Token Queue ensures fairness and avoids starvation by FIFO order.

11. Maekawa's Algorithm Analysis
Each process needs permission from a quorum (subset of processes).

Quorum sets intersect → ensures mutual exclusion.

Drawbacks:

Deadlocks possible without extra mechanisms.

Message complexity is lower than Ricart-Agrawala but needs quorum design.

12. Spanning-Tree vs Weight-Throwing Detection
Feature Spanning-Tree Weight-Throwing
Failure Impact ACK loss → indefinite wait More fault-tolerant
Strength Simple tree-based structure Can tolerate message loss
Limitation Needs reliable channels More complex to implement

13. Chandy-Misra-Haas Deadlock Detection (AND Model)

Processes send probes when blocked.

Probe: <initiator, sender, receiver>

Deadlock detected if a probe returns to initiator → cycle detected.

14. Ricart-Agrawala - Reply Delays and Fairness

Fairness: Based on timestamps.

Even if a reply is delayed, a process can't enter CS until it gets all replies.

Delayed reply delays CS entry, but does not break fairness or mutual exclusion.

15. Suzuki-Kasami Token Algorithm with Multiple Requests

A has token; B, C, D request.

Requests are added to token queue in order of request number.

A passes token to the next in queue (e.g., B).

After B, token goes to C, then D.

Token Queue preserves fairness and avoids starvation.