RICART–AGRAWALA ALGORITHM

- Ricart–Agrawala algorithm is an algorithm to for mutual exclusion in a distributed system proposed by Glenn Ricart and Ashok Agrawala.
- This algorithm is an extension and optimization of Lamport's Distributed Mutual Exclusion Algorithm.
- It follows permission based approach to ensure mutual exclusion.
- Two type of messages (REQUEST and REPLY) are used and communication channels are assumed to follow FIFO order.
- A site send a REQUEST message to all other site to get their permission to enter critical section.
- A site send a REPLY message to other site to give its permission to enter the critical section.
- A timestamp is given to each critical section request using Lamport's logical clock.
- Timestamp is used to determine priority of critical section requests.
- Smaller timestamp gets high priority over larger timestamp.
- The execution of critical section request is always in the order of their timestamp.

Requesting the critical section

(a) When a site S_i wants to enter the CS, it broadcasts a timestamped REQUEST message to all other sites.

(b) When site S_j receives a REQUEST message from site S_i, it sends a REPLY message to site S_i if site S_j is neither requesting nor executing the CS, or if the site S_j is requesting and S_i's request's timestamp is smaller than site S_j's own request's timestamp. Otherwise, the reply is deferred and S_j sets $RD_j[i] := 1$.

Executing the critical section

(c) Site S_i enters the CS after it has received a REPLY message from every site it sent a REQUEST message to.

Releasing the critical section

(d) When site S_i exits the CS, it sends all the deferred REPLY messages: $\forall j$ if $RD_i[j] = 1$, then sends a REPLY message to S_j and sets $RD_i[j] := 0$.

Fig : Ricart–Agrawala algorithm

To enter Critical section:

- When a site S_i wants to enter the critical section, it send a timestamped REQUEST message to all other sites.

- When a site S_j receives a REQUEST message from site S_i, It sends a REPLY message to site S_i if and only if Site S_j is neither requesting nor currently executing the critical section.
- In case Site S_j is requesting, the timestamp of Site S_i's request is smaller than its own request.
- Otherwise the request is deferred by site S_j.

To execute the critical section:

Site S_i enters the critical section if it has received the REPLY message from all other sites.

To release the critical section:

Upon exiting site S_i sends REPLY message to all the deferred requests.

Theorem: Ricart-Agrawala algorithm achieves mutual exclusion.

***Proof: Proof is by contradiction*.**

- Suppose two sites S_i and S_j ' are executing the CS concurrently and S_i 's request has higher priority than the request of S_j . Clearly, Si received S_j 's request after it has made its own request.
- Thus, S_j can concurrently execute the CS with S_i only if Si returns a REPLY to S_j (in response to S_j 's request) before Si exits the CS.
- However, this is impossible because S_j 's request has lower priority. Therefore, Ricart-Agrawala algorithm achieves mutual exclusion.

Message Complexity:

Ricart–Agrawala algorithm requires invocation of 2(N – 1) messages per critical section execution. These 2(N – 1) messages involve:

- (N – 1) request messages
- (N – 1) reply messages

Drawbacks of Ricart–Agrawala algorithm:

- **Unreliable approach:** failure of any one of node in the system can halt the progress of the system. In this situation, the process will starve forever. The problem of failure of node can be solved by detecting failure after some timeout.

Performance:

Synchronization delay is equal to maximum message transmission time It requires 2(N – 1) messages per Critical section execution.