Ricart-Agrawala Algorithm

- The Ricart-Agrawala algorithm assumes the communication channels are FIFO. The algorithm uses two types of messages: REQUEST and REPLY.
- A process sends a REQUEST message to all other processes to request their permission to enter the critical section. A process sends a REPLY message to a process to give its permission to that process.
- Processes use Lamport-style logical clocks to assign a timestamp to critical section requests and timestamps are used to decide the priority of requests.
- Each process p_i maintains the Request-Deferred array, RD_i, the size of which is the same as the number of processes in the system.
- Initially, $\forall i \; \forall j$: RD_i[j]=0. Whenever p_i defer the request sent by p_j, it sets RD_i[j]=1 and after it has sent a REPLY message to p_j, it sets RD_i[j]=0.

Description of the Algorithm

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.

Algorithm

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 send a REPLY message to S_j and set RD_i[j]=0.

Notes:

- When a site receives a message, it updates its clock using the timestamp in the message.
- When a site takes up a request for the CS for processing, it updates its local clock and assigns a timestamp to the request.

Correctness

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, S_i 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 S_i returns a REPLY to S_j (in response to S_j's request) before S_i exits the CS.
- However, this is impossible because S_j's request has lower priority.Therefore, Ricart-Agrawala algorithm achieves mutual exclusion.

Performance

- For each CS execution, Ricart-Agrawala algorithm requires $(N-1)$ REQUEST messages and $(N-1)$ REPLY messages.
- Thus, it requires $2(N-1)$ messages per CS execution.
- Synchronization delay in the algorithm is T.