- # Non Token based Algorithms:Lamport Algorithm

A site communicates with other sites in order to determine which sites should execute the critical section next.

This requires the exchange of two or more successive rounds of messages among sites.

This approach uses timestamps instead of sequence numbers to order requests for the critical section.

Whenever a site makes a request for a critical section, it gets a timestamp. Timestamp is also used to resolve any conflict between critical section requests.

All algorithms which follow a non-token based approach maintain a logical clock. Logical clocks get updated according to Lamport's scheme

Example: Lamport's algorithm, Ricart–Agrawala algorithm

Lamport's Distributed Mutual Exclusion Algorithm

It is a permission based algorithm proposed by Lamport as an illustration of his synchronization scheme for distributed systems.

In this algorithm, permission based timestamp is used to order critical section requests and to resolve any conflict between requests.

In Lamport's Algorithm, critical section requests are executed in the increasing order of timestamps i.e a request with smaller timestamp will be given permission to execute critical section first than a request with larger timestamp.

In this algorithm:

Three types of messages (REQUEST, REPLY and RELEASE) are used and communication channels are assumed to follow FIFO order.

A site sends a REQUEST message to all other sites to get their permission to enter a critical section.

A site sends a REPLY message to a requesting site to give its permission to enter the critical section.

A site sends a RELEASE message to all other sites upon exiting the critical section.

Every site, Si, keeps a queue to store critical section requests ordered by their timestamps. request_queuei denotes the queue of site Si

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 requests is always in the order of their timestamp.

Algorithm:

- To enter Critical section:

When a site Si wants to enter the critical section, it sends a request message Request(Tsi, i) to all other sites and places the request on request_queuei. Here, Tsi denotes the timestamp of Site Si

When a site Sj receives the request message REQUEST(Tsi, i) from site Si, it returns a timestamped REPLY message to site Si and places the request of site Si on request_queuej

- To execute the critical section:

A site Si can enter the critical section if it has received the message with timestamp larger than (Tsi, i) from all other sites and its own request is at the top of request_queuei

- To release the critical section:

When a site Si exits the critical section, it removes its own request from the top of its request queue and sends a timestamped RELEASE message to all other sites

When a site Sj receives the timestamped RELEASE message from site Si, it removes the request of Si from its request queue

Ricart–Agrawala's Algorithm

Ricart–Agrawala algorithm is an algorithm 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.

Like Lamport's Algorithm, it also follows a permission-based approach to ensure mutual exclusion. In this algorithm:

- Two types of messages (REQUEST and REPLY) are used and communication channels are assumed to follow FIFO order.
- A site sends a REQUEST message to all other sites to get their permission to enter the critical section.
- A site sends a REPLY message to another 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 requests is always in the order of their timestamp.

Algorithm:

- To enter Critical section:
  - When a site Si wants to enter the critical section, it sends a timestamped REQUEST message to all other sites.
  - When a site Sj receives a REQUEST message from site Si, It sends a REPLY message to site Si if and only if
    - Site Sj is neither requesting nor currently executing the critical section.
    - In case Site Sj is requesting, the timestamp of Site Si's request is smaller than its own request.
- To execute the critical section:

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

- To release the critical section:
  - Upon exiting the site Si sends a REPLY message to all the deferred requests.

- **Maekawa's Algorithm**

Instead of requesting permission to execute the critical section from all other sites, Each site requests only a subset of sites which is called a quorum. Any two subsets of sites or Quorum contains a common site. This common site is responsible to ensure mutual exclusion

Example: Maekawa's Algorithm

Maekawa's Algorithm is a quorum based approach to ensure mutual exclusion in distributed systems.

As we know, In permission based algorithms like Lamport's Algorithm, Ricart-Agrawala Algorithm etc. a site request permission from every other site but in quorum based approach, A site does not request permission from every other site but from a subset of sites which is called quorum.

In this algorithm:

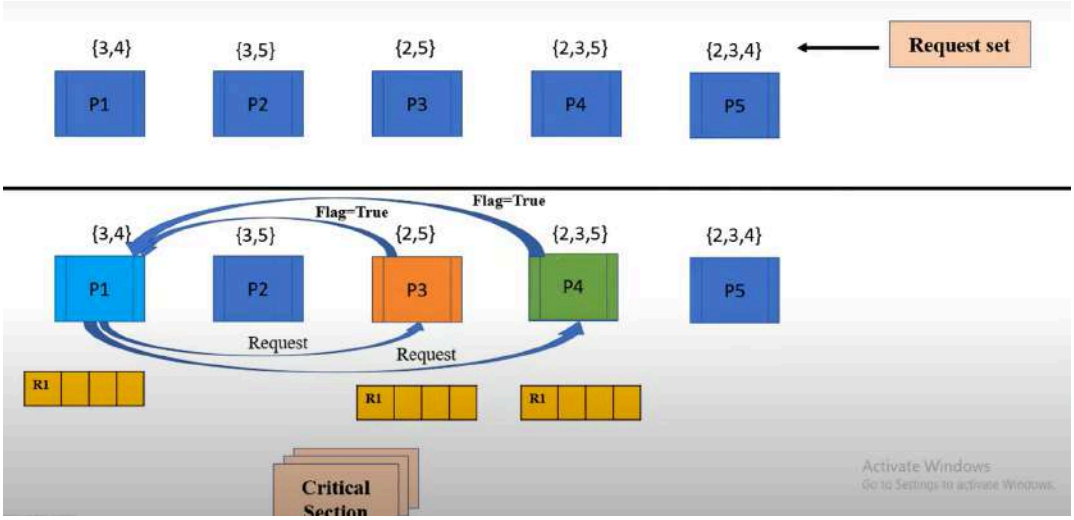
- Three types of messages (REQUEST, REPLY and RELEASE) are used.
- A site sends a REQUEST message to all other sites in its request set or quorum to get their permission to enter the critical section.
- A site sends a REPLY message to a requesting site to give its permission to enter the critical section.
- A site sends a RELEASE message to all other sites in its request set or quorum upon exiting the critical section.

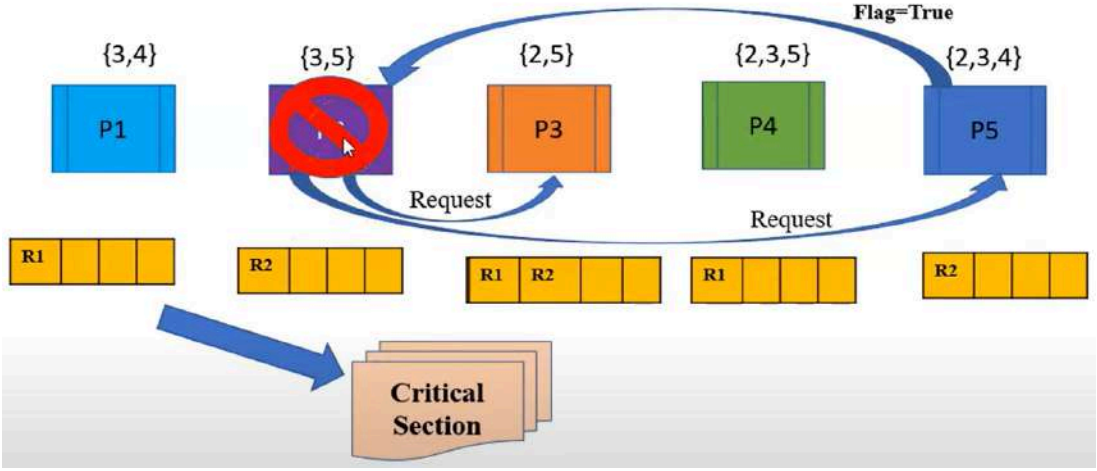
Assumptions and Rules:

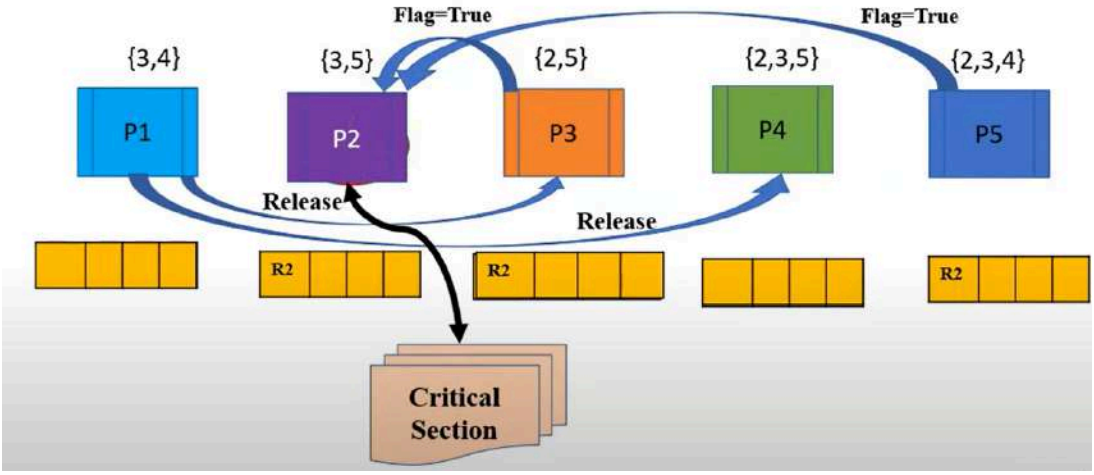
- Every process should belong to some group.

- If any process wants to access Critical section then it will send request to group members and receive reply from group members
- Rules and Process:
- The intersection of two groups != Null
- The Process should be part of any group
- Single process cannot be considered as a single group.

Example:







Algorithm:

- To enter Critical section:

When a site Si wants to enter the critical section, it sends a request message REQUEST(i) to all other sites in the request set Ri.

When a site Sj receives the request message REQUEST(i) from site Si, it returns a REPLY message to site Si if it has not sent a REPLY message to the site from the time it received the last RELEASE message. Otherwise, it queues up the request.

- To execute the critical section:

A site Si can enter the critical section if it has received the REPLY message from all the site in request set Ri

- To release the critical section:

When a site Si exits the critical section, it sends RELEASE(i) message to all other sites in request set Ri

When a site Sj receives the RELEASE(i) message from site Si, it send REPLY message to the next site waiting in the queue and deletes that entry from the queue

In case queue is empty, site Sj update its status to show that it has not sent any REPLY message since the receipt of the last RELEASE message.