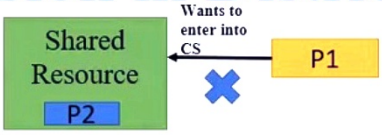
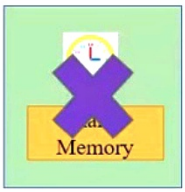
❖ Mutual Exclusion in Distributed Systems

- Mutual Exclusion

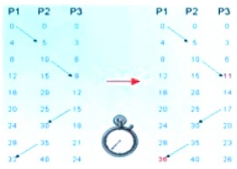


- Mutual Exclusion in Distributed Systems
  - ➢ so cannot solve Mutual Exclusion using semaphores (as it requires Shared memory)
  - ➢ Hence an approach based on message passing is used to solve Mutual Exclusion problem



- Solution to Mutual Exclusion in DS based on message passing

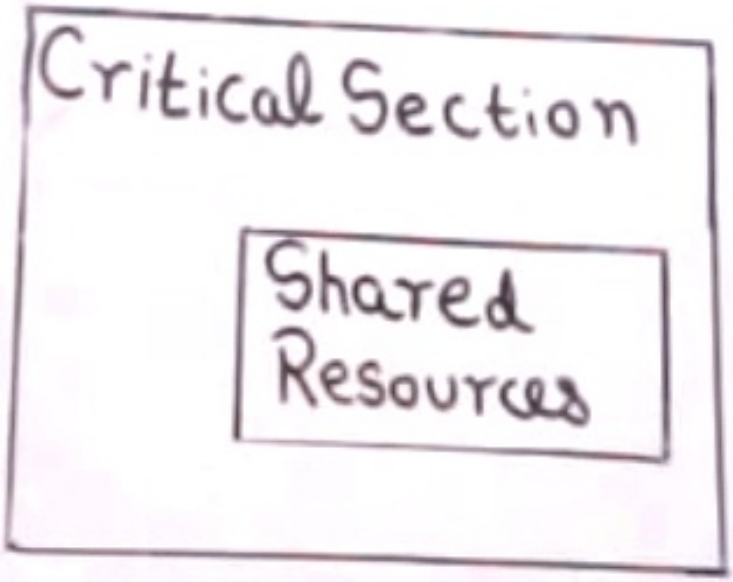
➢ Non Token based algorithms
  - ➢ Lamport's Mutual Exclusion algorithm
  - ➢ Ricart-Agrawala algorithm



Lamport's Algorithm

Critical Section: It is a Section which is accessible only to one Process at a time.

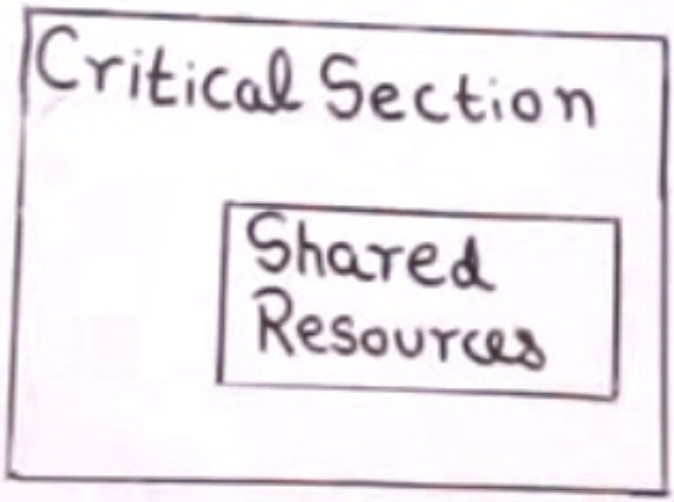
Mutual Exclusion: It makes Sure that Processes access Shared resources or data in Serealized way.

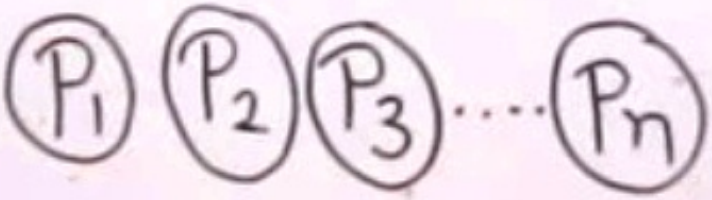


Lamport's Algorithm

Critical Section: It is a Section which is accessible only to one Process at a time.

Mutual Exclusion: It makes Sure that Processes access Shared resources or data in Serealized way.





General System Model

There are 3 types of messages

- Request : sent to all other site to get their permission to enter critical section.
- Reply : message is sent to requesting site to give its permission to enter the critical section
- Release : upon exiting the critical section

Queue is maintained for each process to store critical section requests ordered by their timestamps.

Assumption : Messages are delivered in FIFO order.

-  → critical section request → 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.

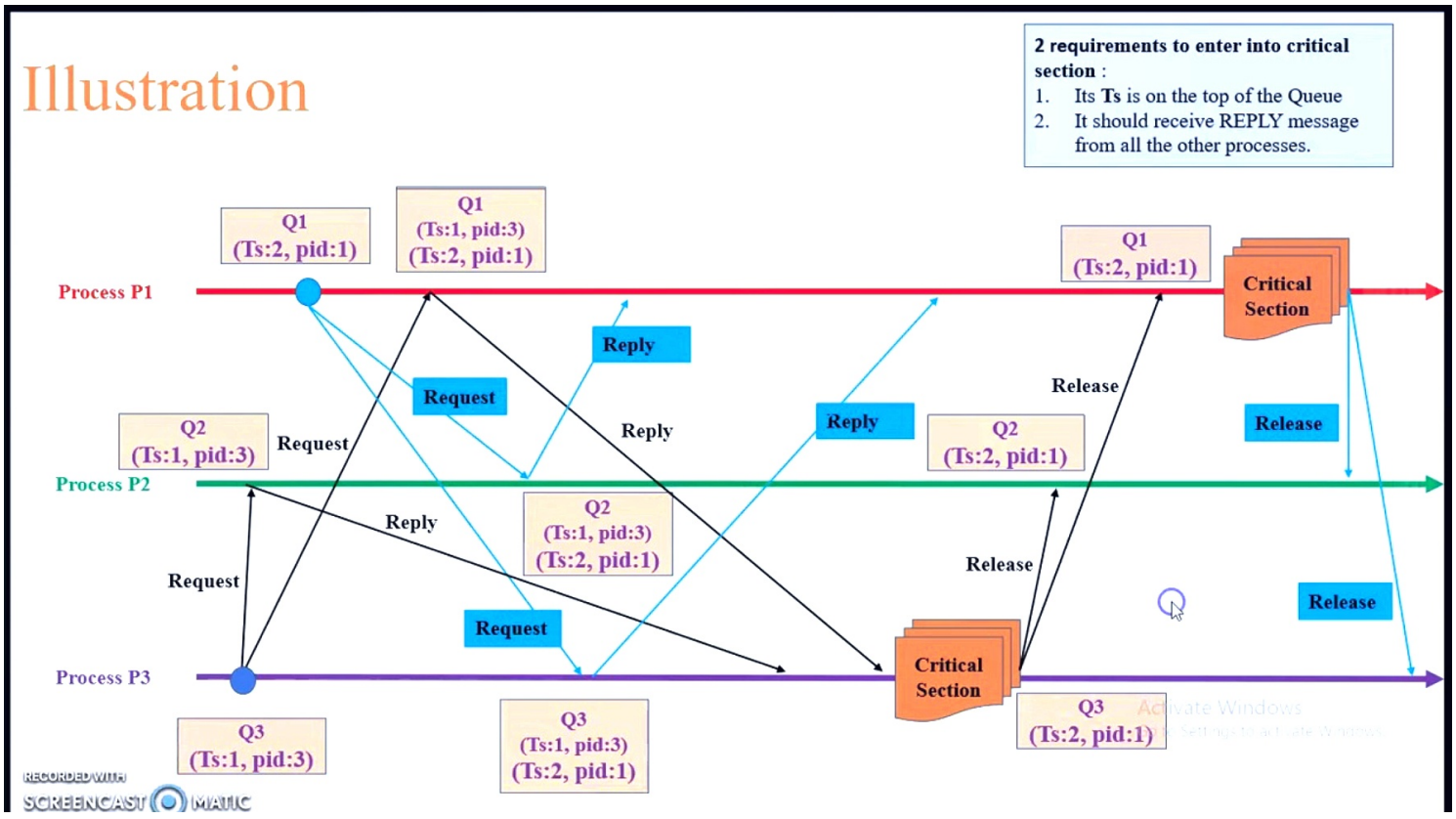
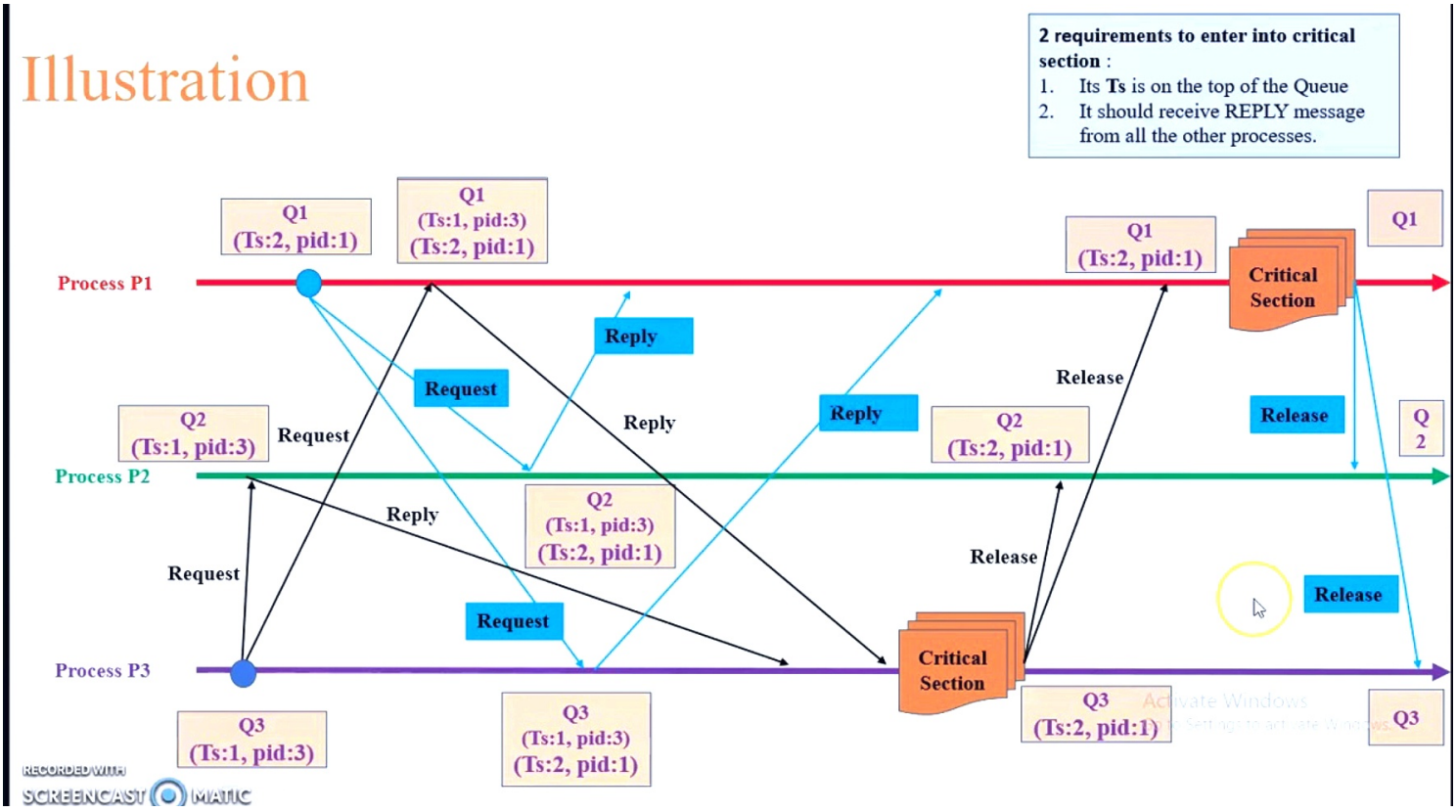


Illustration
2 requirements to enter into critical section :
1. Its Ts is on the top of the Queue
2. It should receive REPLY message from all the other processes.
Process P1
Process P2
Process P3
Q1 (Ts:2, pid:1)
Q1 (Ts:1, pid:3) (Ts:2, pid:1)
Q1 (Ts:2, pid:1)
Critical Section
Request
Reply
Release
Q2 (Ts:1, pid:3)
Q2 (Ts:2, pid:1)
Q2 (Ts:1, pid:3) (Ts:2, pid:1)
Q3 (Ts:1, pid:3)
Q3 (Ts:1, pid:3) (Ts:2, pid:1)
Q3 (Ts:2, pid:1)
RECORDED WITH SCREENCAST MATIC

Illustration



Algorithm

- **To enter Critical section:**

➢ Request message **Request(ts_i, i)** is sent to all other sites and places the request on **$queue_i$**.

➢ After receiving the request message , a timestamped REPLY message is sent and and places the request on **$queue_j$**

- **To execute the critical section:**
  - if it has received the message from all other sites.
  - its own request is at the top of **$queue_i$**

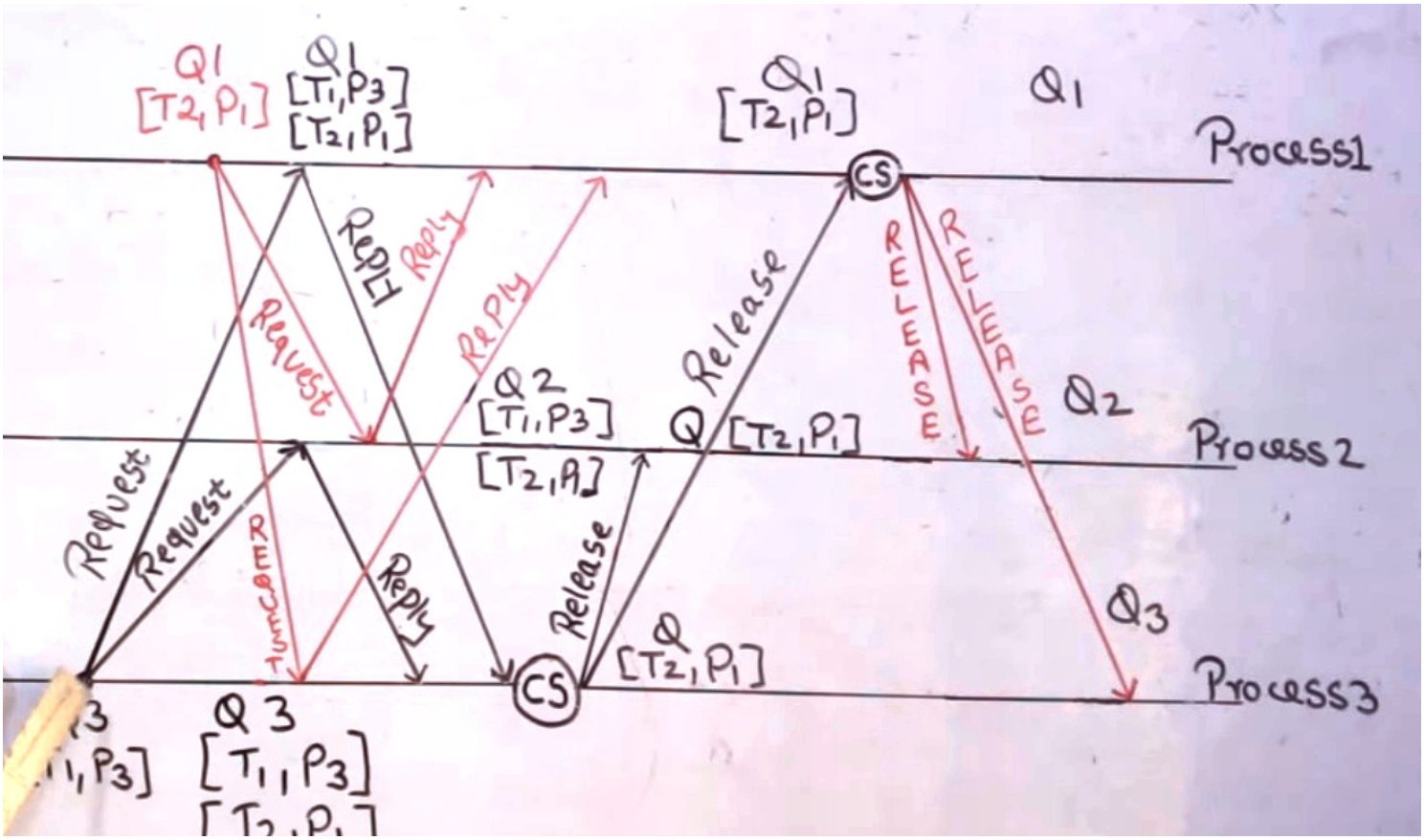
- **To release the critical section:**
  - When a P_i exits the critical section, it removes its own request from the top of its queue and sends a timestamped **RELEASE** message to all other sites

Performance.

- 3(N-1) messages per CS invocation.
  - (N - 1) REQUEST, (N - 1) REPLY, (N - 1) RELEASE messages.

Algorithm

1. P_i sends request to all P_js
2. P_i makes entry in its own queue
3. P_j receives Request message
4. P_j makes entry in its own queue & sends reply to P_i
5. Conditions for Critical Section
   - P_i has received Reply from all those P_js whome he has sent request
   - In Queue his request is the topmost
6. P_i exits the CS and removes itself from Que
7. P_i Broadcast Release message
8. On receiving message P_j removes P_i's Entry


Q1
[T2,P1]
Q1
[T1,P3]
[T2,P1]
Q1
[T2,P1]
Q1
Process1
Request
Request
Reply
Reply
Reply
CS
RELEASE
RELEASE
Q2
[T1,P3]
[T2,P1]
Q
[T2,P1]
Q2
Process2
Request
Request
Request
Reply
Release
Release
Q
[T2,P1]
Q3
Process3
CS
Q3
[T1,P3]
[T2,P1]