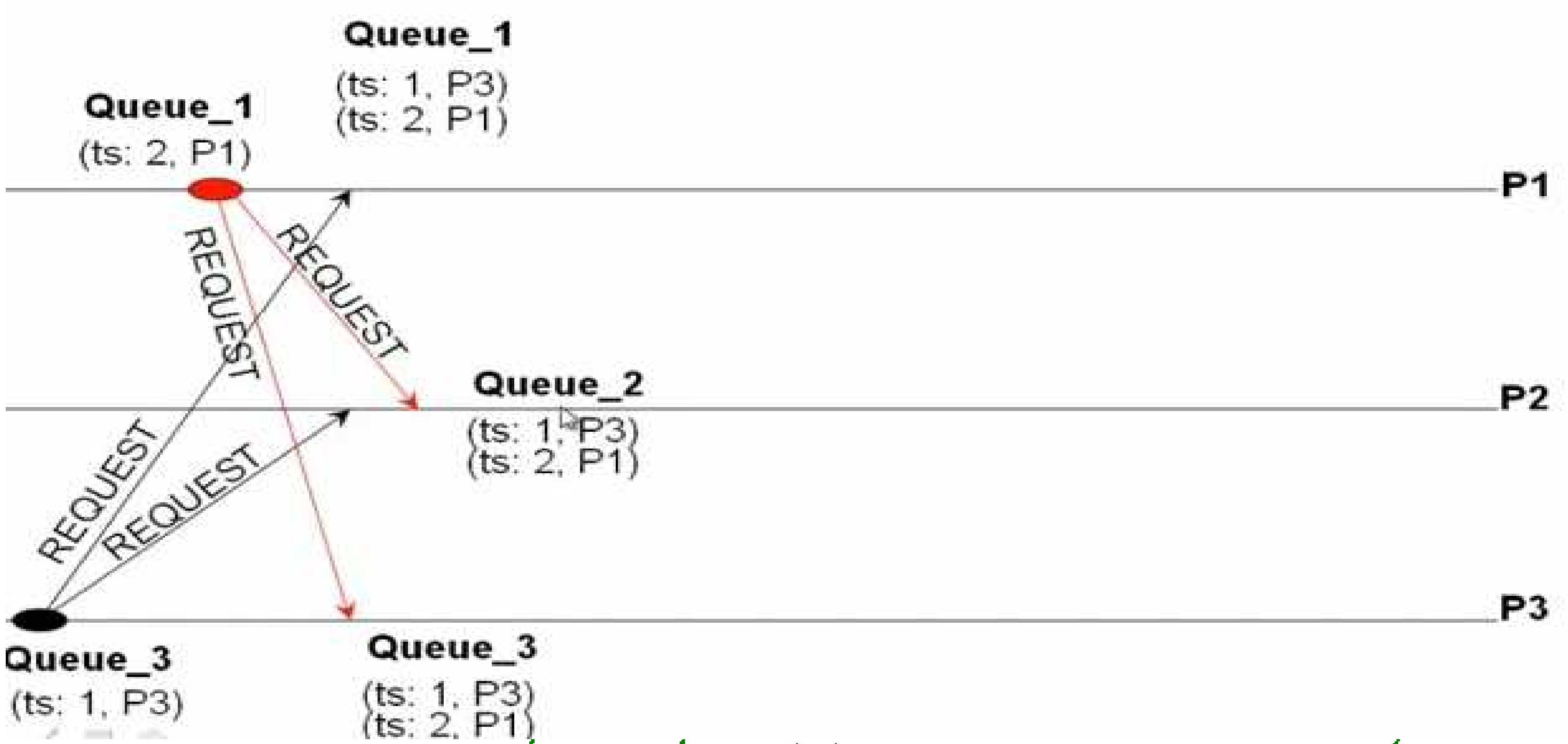
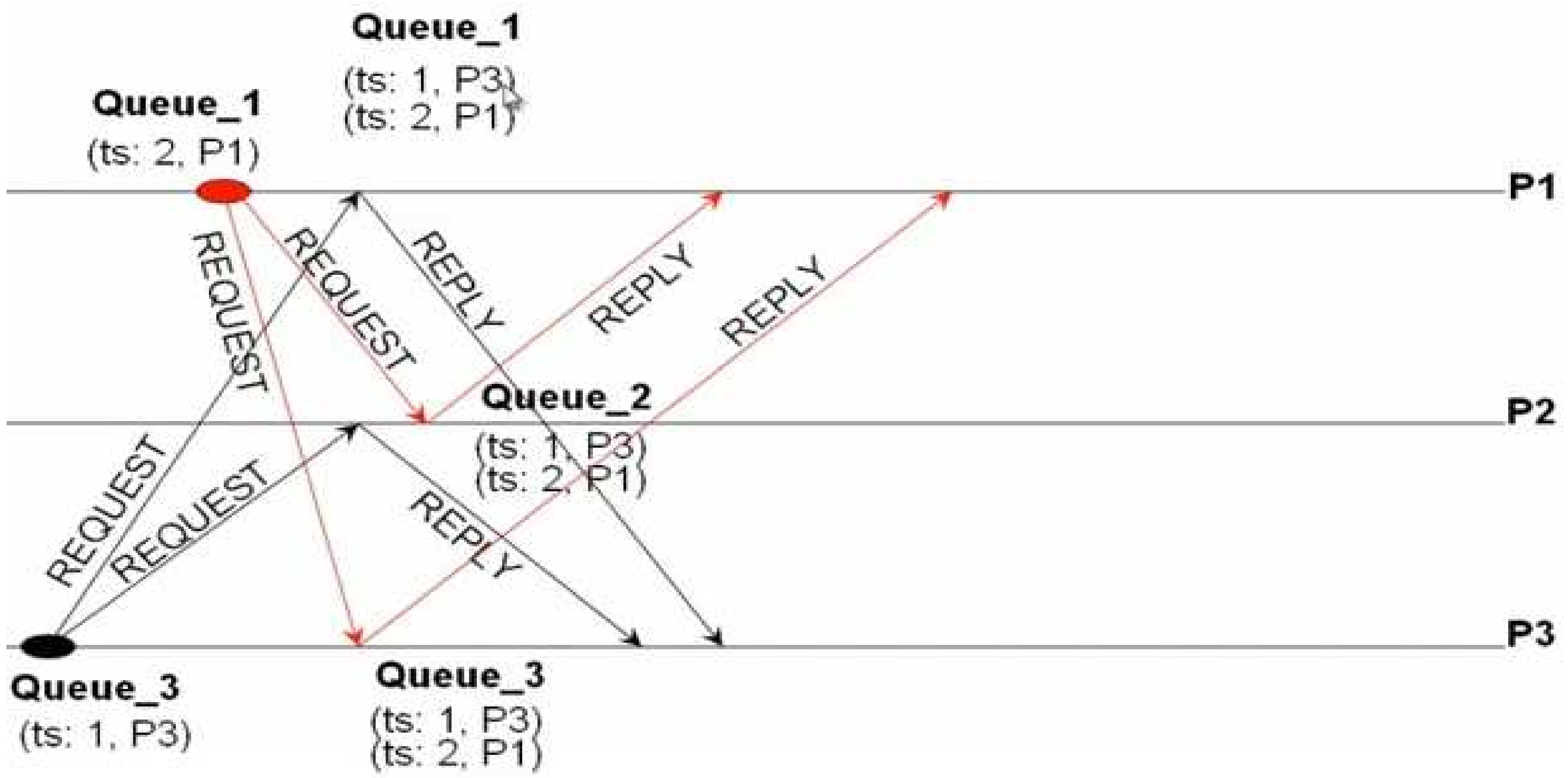
Lamport’s Distributed Mutual Exclusion Algorithm

Reference : Mukesh Singhal & N.G. Shivaratri, Advanced Concepts in Operating Systems,

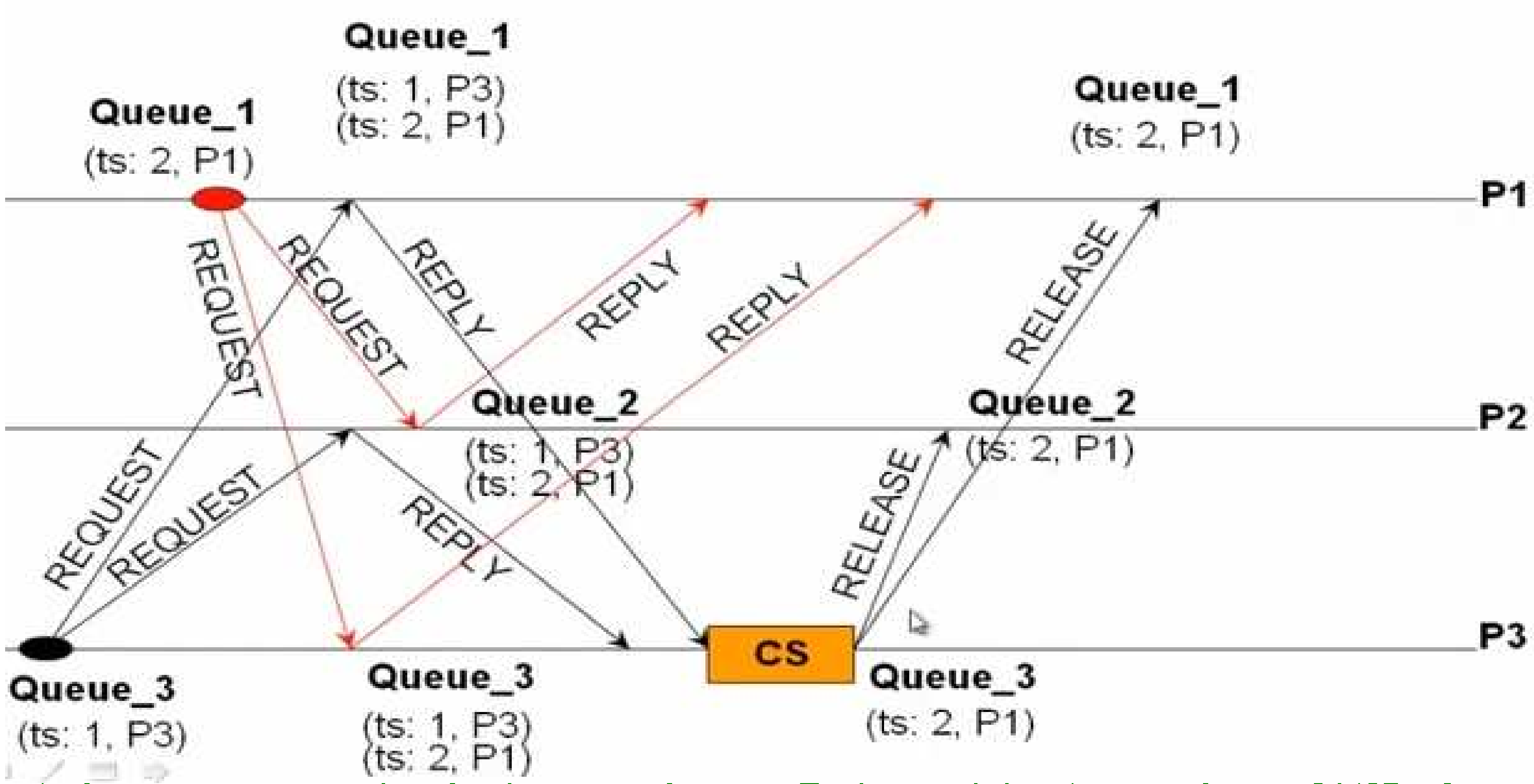
Lamport's Distributed Mutual Exclusion Algorithm



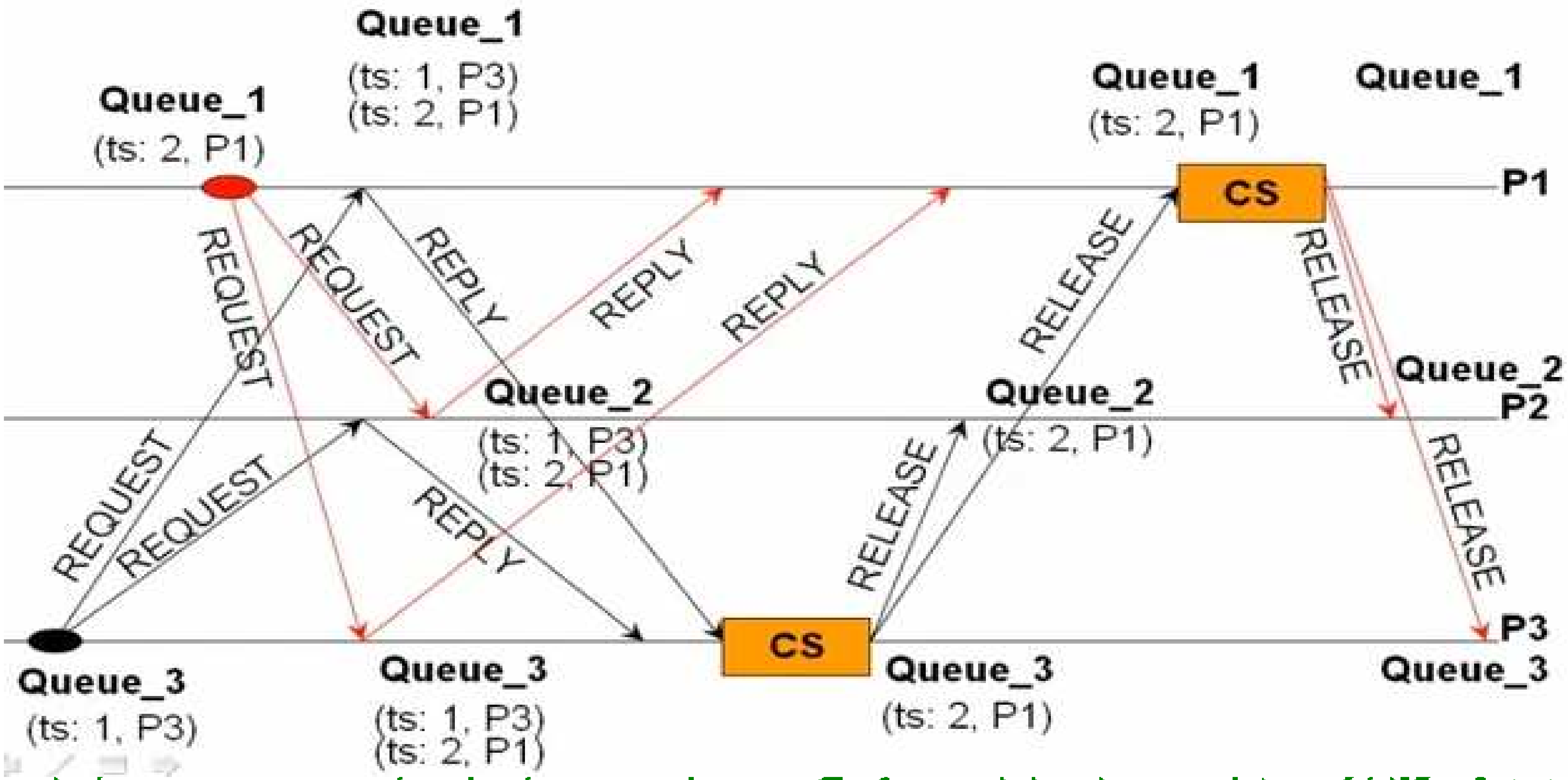
Lamport's Distributed Mutual Exclusion Algorithm



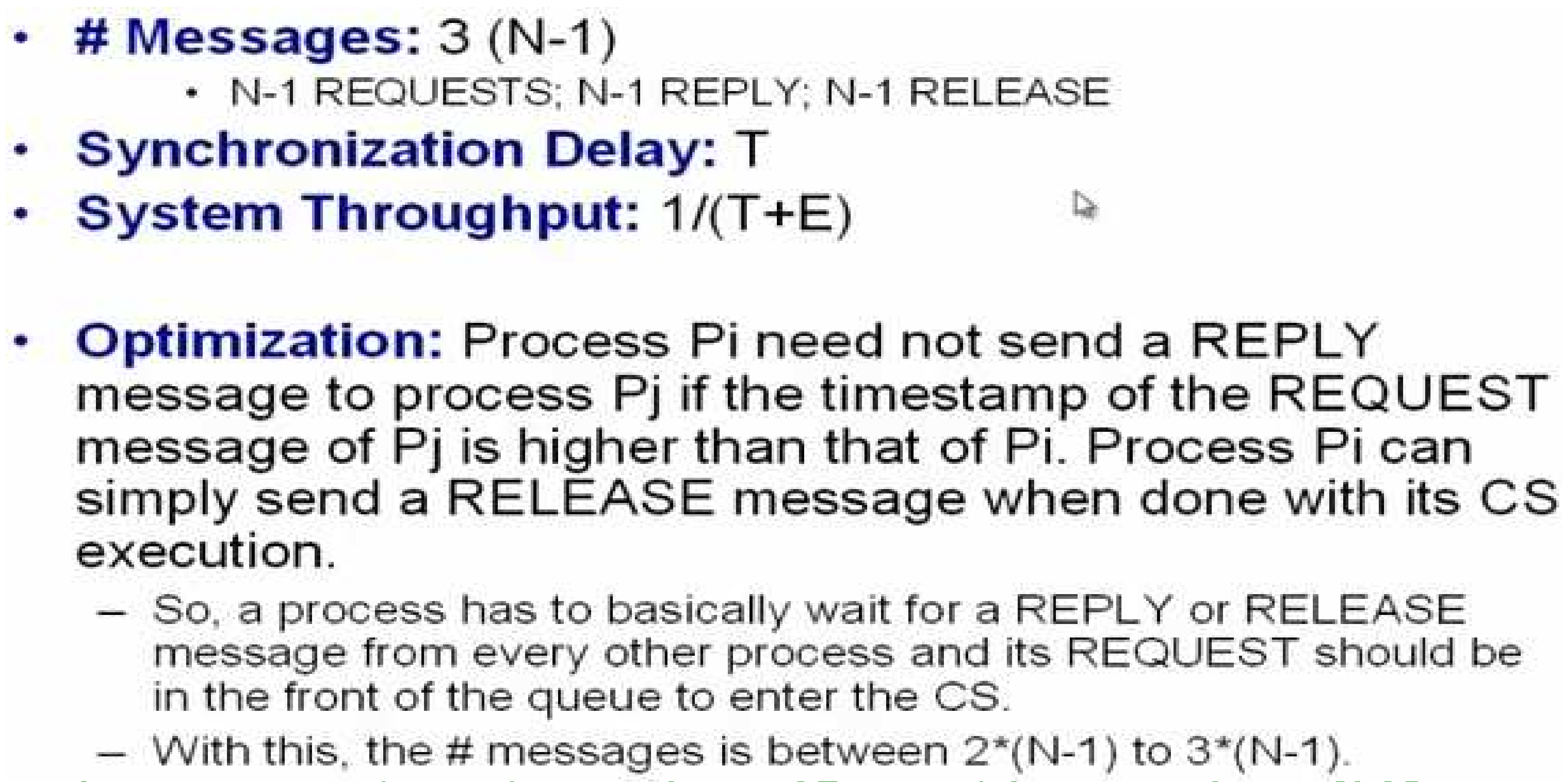
Lamport's Distributed Mutual Exclusion Algorithm



Lamport's Distributed Mutual Exclusion Algorithm



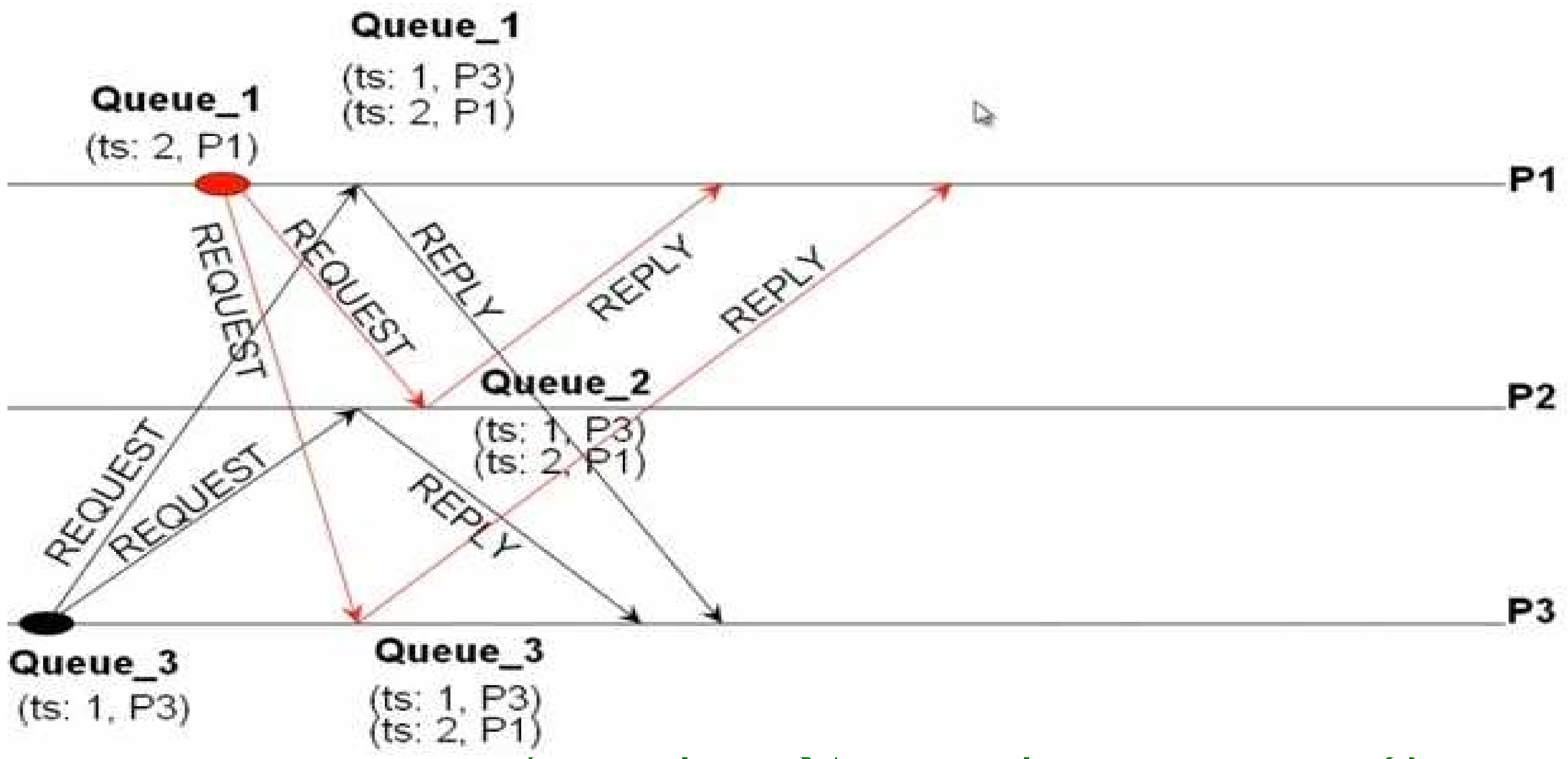
Lamport's Distributed Mutual Exclusion Algorithm



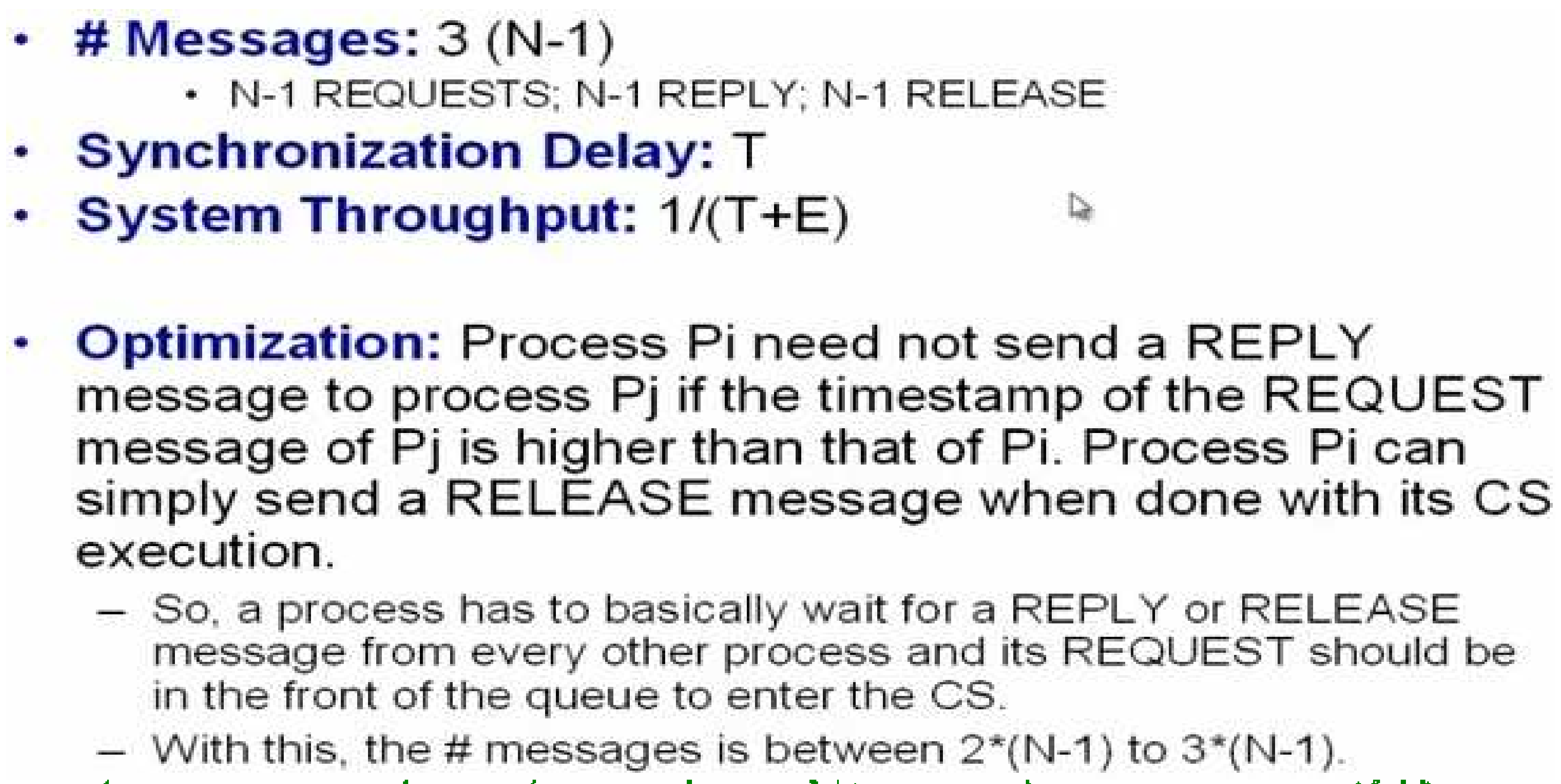
- **# Messages:** 3 (N-1)
  - N-1 REQUESTS; N-1 REPLY; N-1 RELEASE
- **Synchronization Delay:** T
- **System Throughput:** 1/(T+E)

- **Optimization:** Process Pi need not send a REPLY message to process Pj if the timestamp of the REQUEST message of Pj is higher than that of Pi. Process Pi can simply send a RELEASE message when done with its CS execution.
  – So, a process has to basically wait for a REPLY or RELEASE message from every other process and its REQUEST should be in the front of the queue to enter the CS.
  – With this, the # messages is between 2*(N-1) to 3*(N-1).

Lamport's Distributed Mutual Exclusion Algorithm



Lamport's Distributed Mutual Exclusion Algorithm



- **# Messages:** 3 (N-1)
  - N-1 REQUESTS; N-1 REPLY; N-1 RELEASE
- **Synchronization Delay:** T
- **System Throughput:** 1/(T+E)

- **Optimization:** Process Pi need not send a REPLY message to process Pj if the timestamp of the REQUEST message of Pj is higher than that of Pi. Process Pi can simply send a RELEASE message when done with its CS execution.
  – So, a process has to basically wait for a REPLY or RELEASE message from every other process and its REQUEST should be in the front of the queue to enter the CS.
  – With this, the # messages is between 2*(N-1) to 3*(N-1).

Source

https://www.youtube.com/watch?v=r7SJOhGF4Nc

Thank You