Distributed mutual Exclusion Algorithm

1. Token-based approach
2. Non-token based approach
3. Quorum based approach

Requirements of mutual exclusion algorithms

1. Safety property :- at a time only one process can execute CS.
2. Liveness property :- No starvation / deadlock
3. Fairness :- (logical clock)

FIFO → message ordering.

Performance Metrics

1. Message Complexity :- # of msgs.
2. Synchronization delay (SD) :-
3. Response time
4. System Throughput :-

$$\frac{1}{(SD + E)}$$

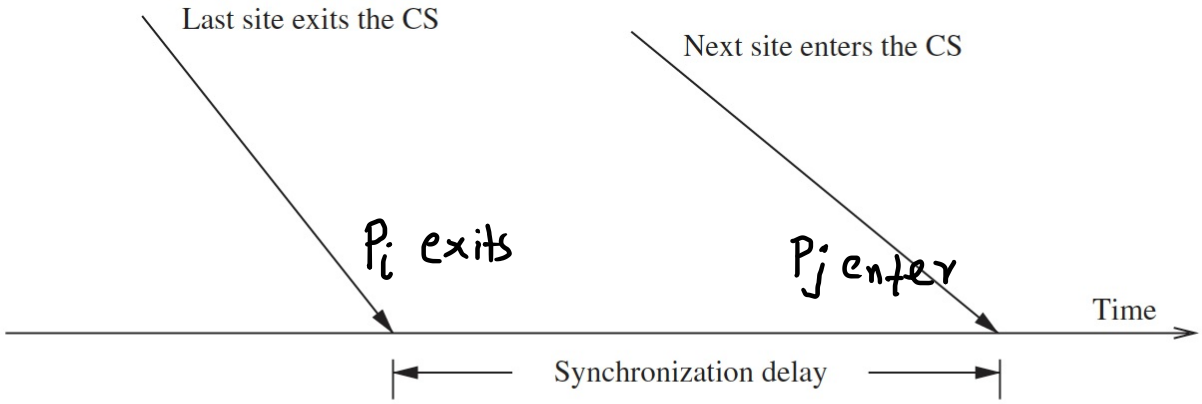


Figure 9.1 Synchronization delay.

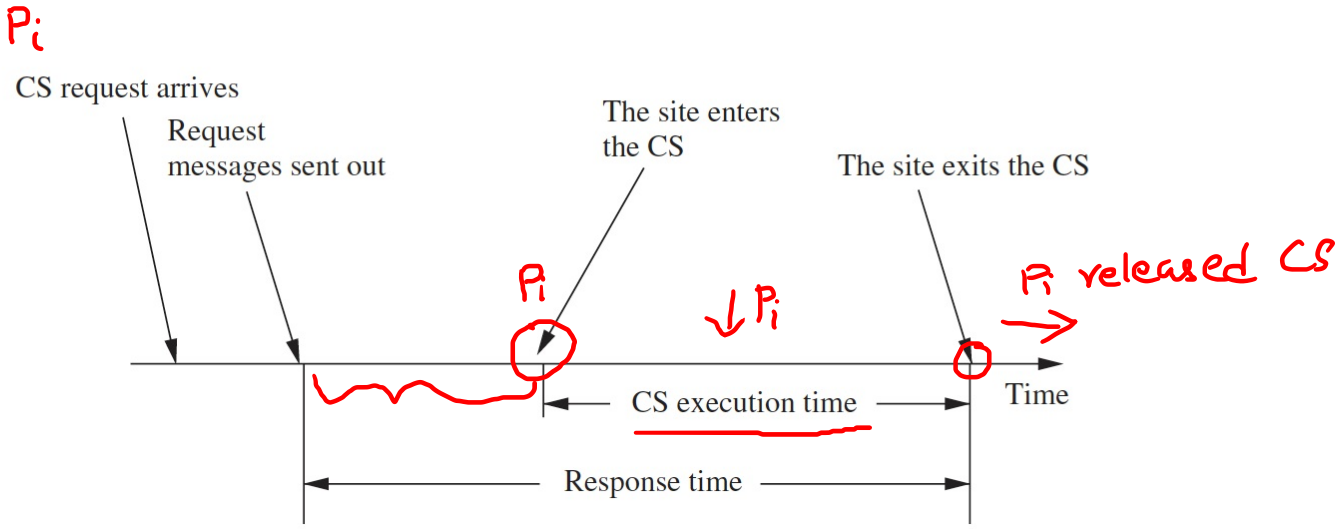


Figure 9.2 Response time.

Lamports Algorithm

"n" → (n-1)

Requesting the critical section

- When a site S_i wants to enter the CS, it broadcasts a REQUEST(ts_i, i) message to all other sites and places the request on $request\_queue_i$. ((ts_i, i) denotes the timestamp of the request.)
- When a site S_j receives the REQUEST(ts_i, i) message from site S_i, it places site S_i's request on $request\_queue_j$ and returns a timestamped REPLY message to S_i.

Executing the critical section

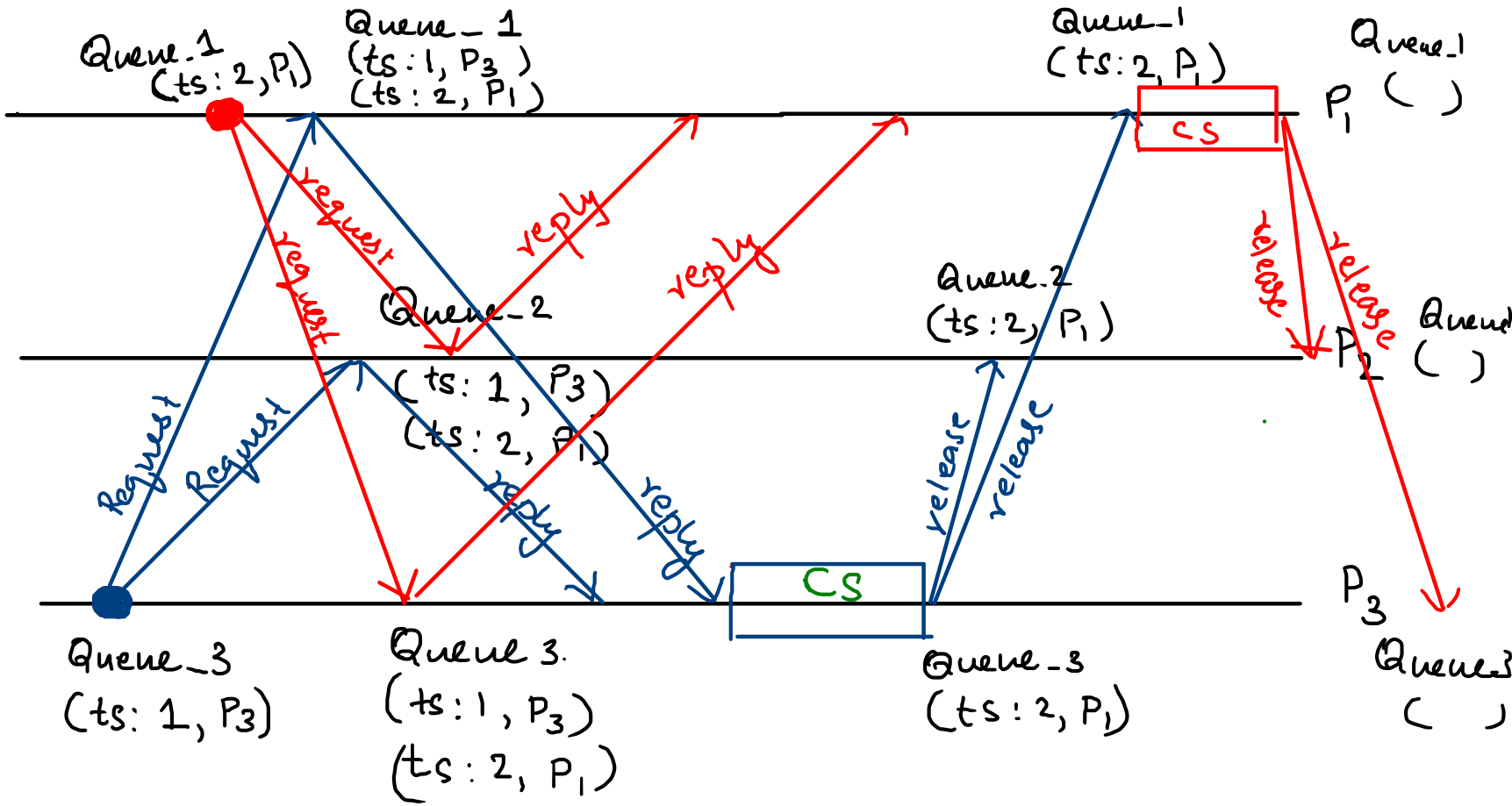
Site S_i enters the CS when the following two conditions hold:

- **L1:** S_i has received a message with timestamp larger than (ts_i, i) from all other sites.
- **L2:** S_i's request is at the top of $request\_queue_i$.

Releasing the critical section

- Site S_i, upon exiting the CS, removes its request from the top of its request queue and broadcasts a timestamped RELEASE message to all other sites.
- When a site S_j receives a RELEASE message from site S_i, it removes S_i's request from its request queue.

Algorithm 9.1 Lamport's algorithm.



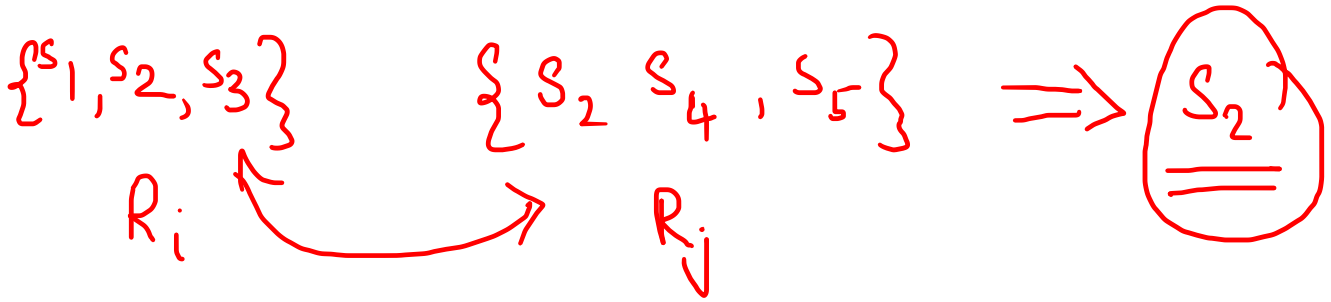
message complexity

$3(N-1)$

Quorum based Mutual Exclusion Algm.

I.N: R_i & R_j

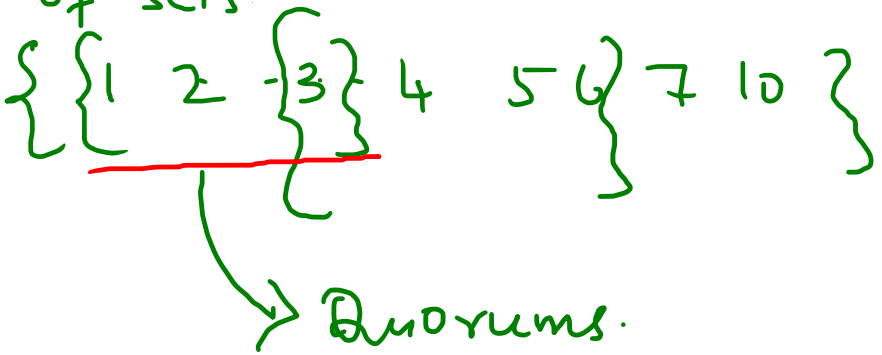
$R_i \cap R_j \neq \phi$ → conflict resolver



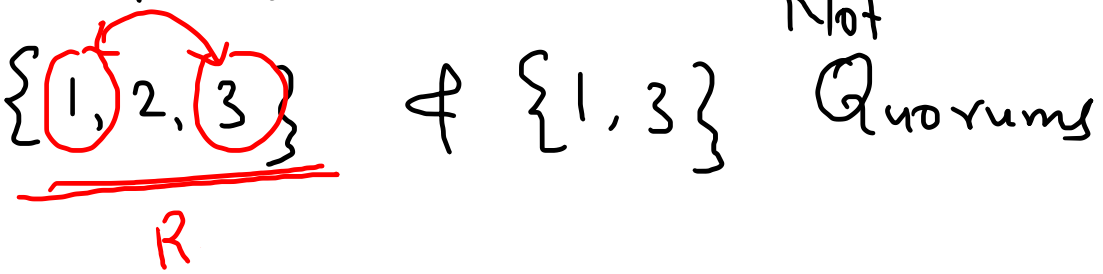
REPLY MESSAGES

Reducing the msg.

- Coteries (C) → Set of sets
- Quorums.



- Intersection Property
- Minimality Property



Site "a" ∈ Quorum "A" → {a, b, c}

↓ request for ME

{b, c}

" Common site send permission to only one site ".

Maekaw's Algm

M_1 : Intersection Property

M_2 : $\forall i \; 1 \leq i \leq N : S_i \in R_i$

M_3 : $|R_i| = k \implies$ '$\sqrt{N}$'

M_4 : Any site S_j is contained in K number of R_i's

Requesting the critical section:

(a) A site S_i requests access to the CS by sending REQUEST(i) messages to all sites in its request set R_i.

(b) When a site S_j receives the REQUEST(i) message, it sends a REPLY(j) message to S_i provided it hasn't sent a REPLY message to a site since its receipt of the last RELEASE message. Otherwise, it queues up the REQUEST(i) for later consideration.

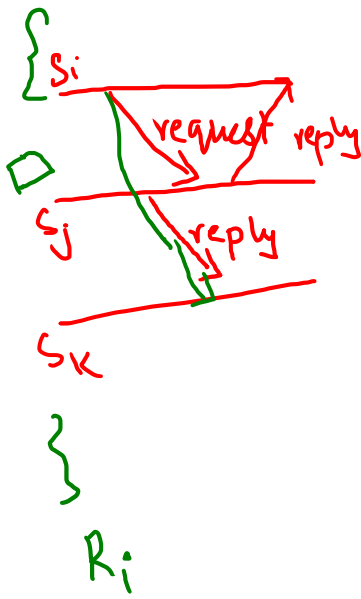
Executing the critical section:

(c) Site S_i executes the CS only after it has received a REPLY message from every site in R_i.

Releasing the critical section:

(d) After the execution of the CS is over, site S_i sends a RELEASE(i) message to every site in R_i.

(e) When a site S_j receives a RELEASE(i) message from site S_i, it sends a REPLY message to the next site waiting in the queue and deletes that entry from the queue. If the queue is empty, then the site updates its state to reflect that it has not sent out any REPLY message since the receipt of the last RELEASE message.



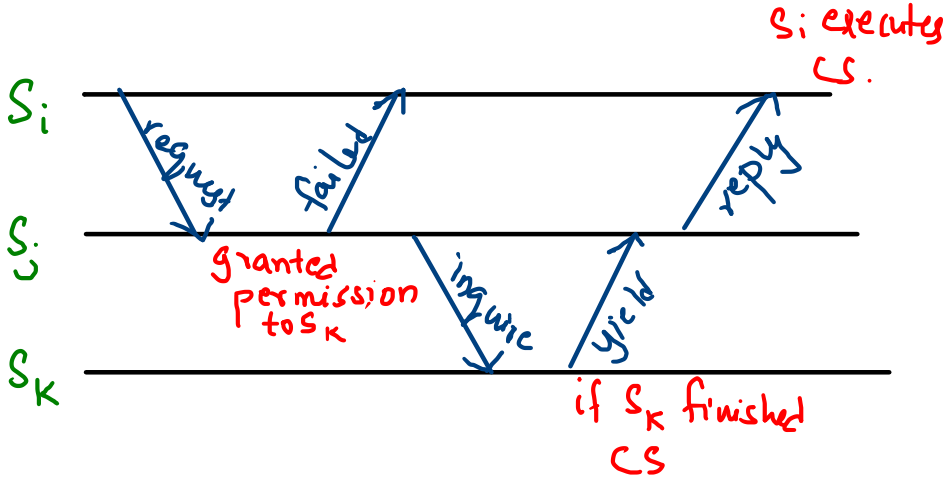
Algorithm 9.5 Maekawa's algorithm.

Deadlock

When higher priority request arrives and wait.

3 types of messages

1. FAILED → S_i → S_j
2. INQUIRE → lock
3. YIELD →



Token based Algorithm

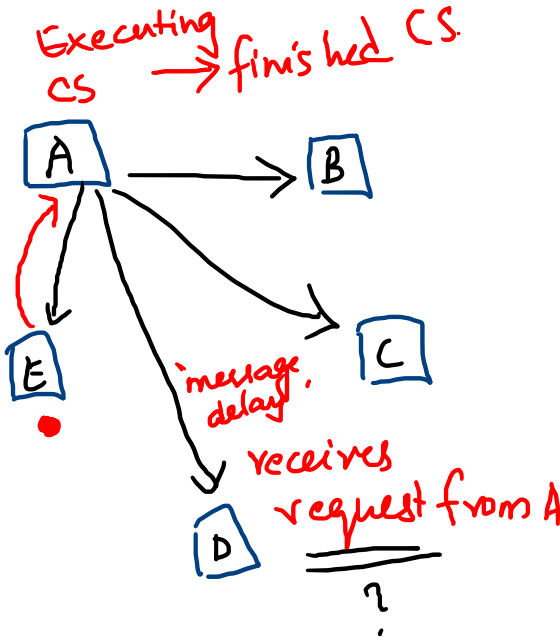
Suzuki_kasami's "broadcast" algorithm

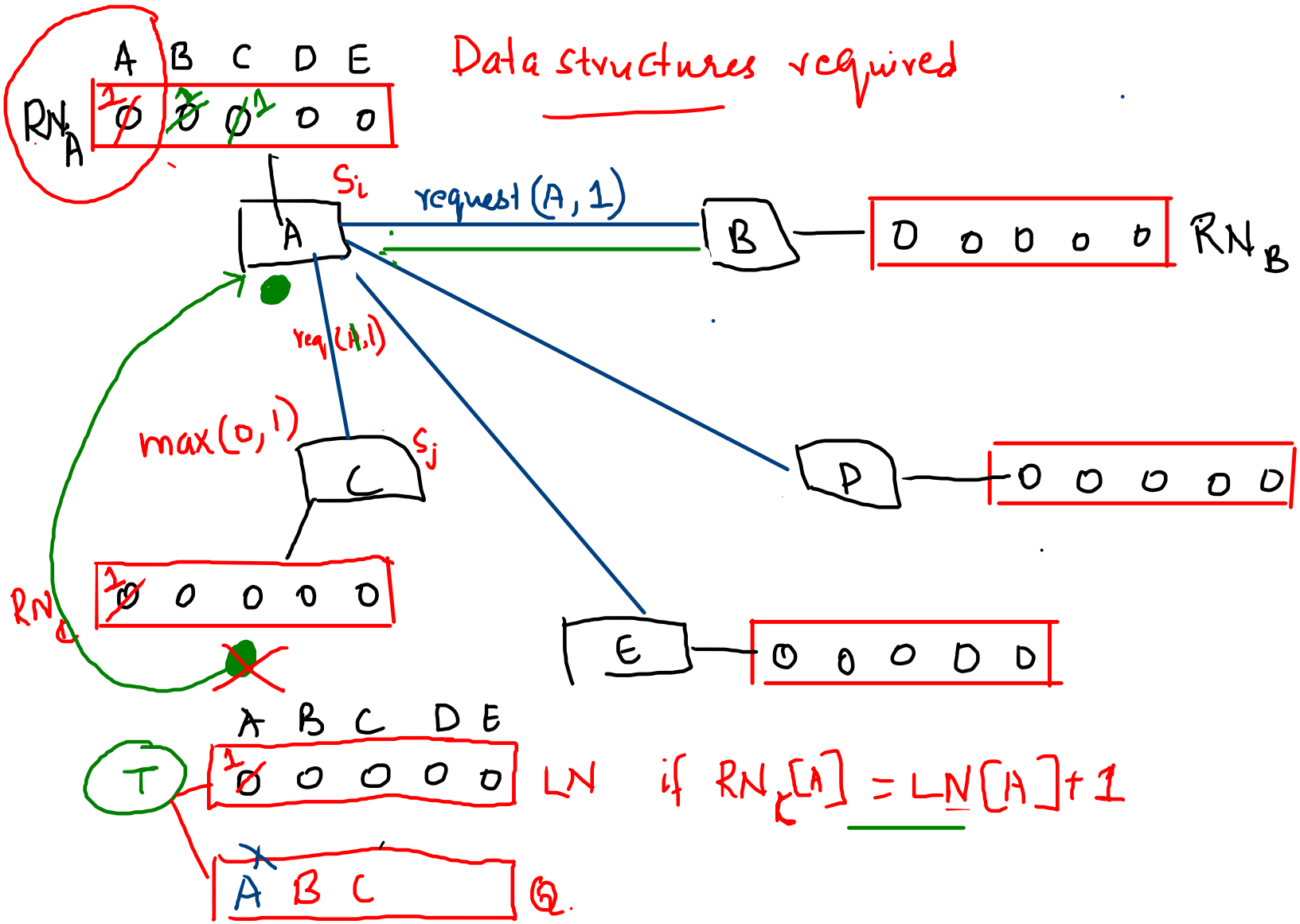
1. Request for the token

   1. Outdated Vs Current request (messages)
   2. Outstanding request for CS. → (Site)

      ↓

      Token → Queue




Data structures required
A B C D E
RN A
1 0 2 0 1 0 0
Si
request (A, 1)
A
B
0 0 0 0 0
RN B
req (A,1)
max(0,1)
C
Sj
P
0 0 0 0 0
RN C
1 0 0 0 0 0
E
0 0 0 0 0
A B C D E
T
1 0 0 0 0 0
LN
if RN i[A] = LN[A] + 1
X
A B C
Q