Mansoura University
Faculty of Computers and Information Sciences
Department of Computer Science
First Semester- 2020-2021

[CS412P] Distributed Systems

Grade : Fourth grade

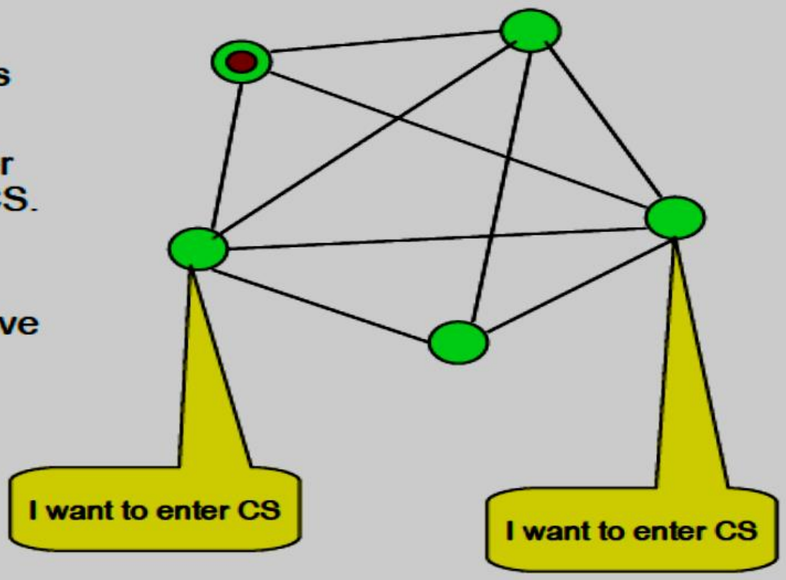
By : Zeinab Awad

TOKEN PASSING ALGORITHMS

Completely connected network of processes

There is **one token** in the network. The holder of the token has the permission to enter CS.

Any other process trying to enter CS must acquire that token. Thus the token will move from one process to another based on demand.



TOKEN PASSING ALGORITHMS

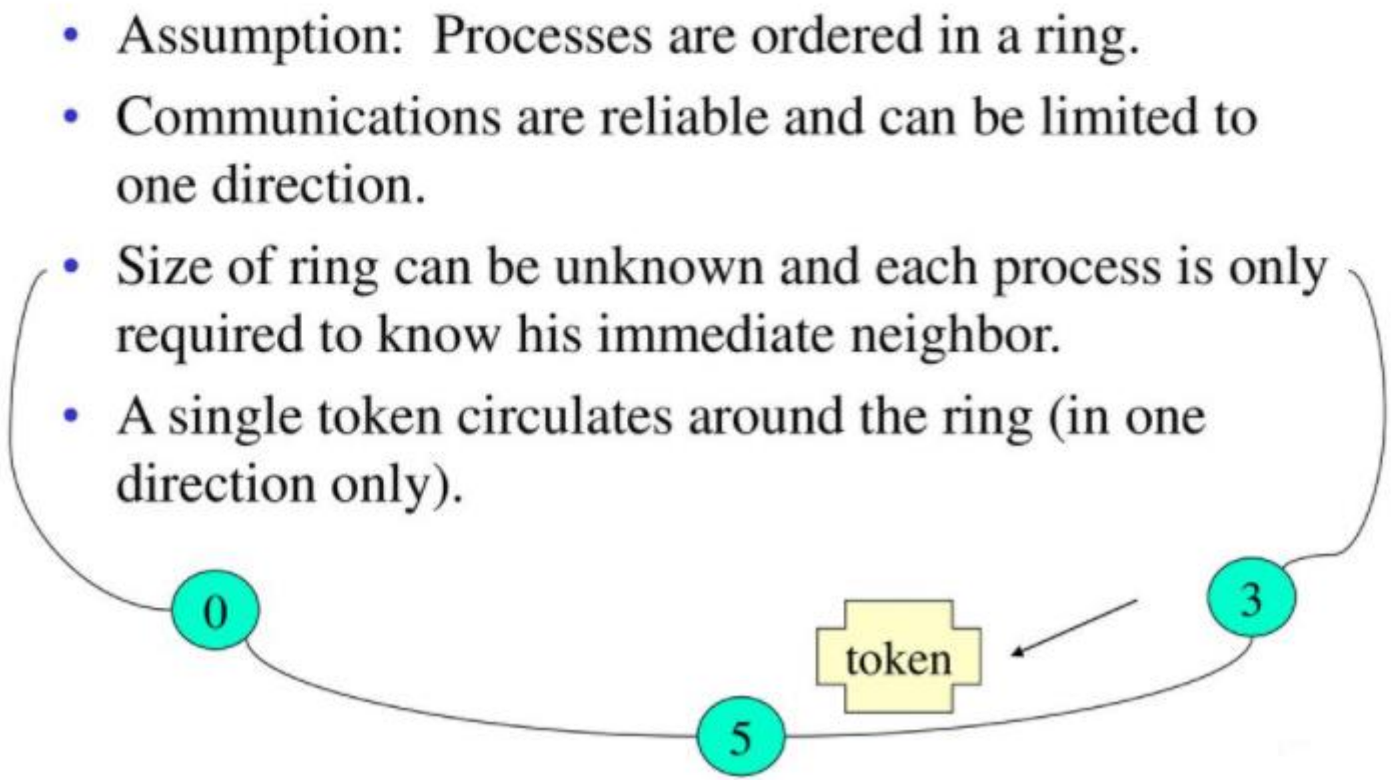
- A token is circulated in a logical ring.
- A process enters its CS if it has the token.
- Issues:
  - If the token is lost, it needs to be regenerated.
  - Detection of the lost token is difficult since there is no bound on how long a process should wait for the token.
  - If a process can fail, it needs to be detected and then by-passed.
  - When nobody wants to enter, processes keep on exchanging messages to circulate the token

TOKEN PASSING ALGORITHMS

-in token-based algorithms, A site is allowed to enter its critical section if it possesses the unique token.

-Non-token based algorithms uses timestamp to order requests for the critical section where as sequence number is used in token based algorithms.
Each requests for critical section contains a sequence number. This sequence number is used to distinguish old and current requests.

TOKEN RING MUTUAL EXECLUSION ALGORITHIM



- Assumption: Processes are ordered in a ring.
- Communications are reliable and can be limited to one direction.
- Size of ring can be unknown and each process is only required to know his immediate neighbor.
- A single token circulates around the ring (in one direction only).

TOKEN RING MUTUAL EXECLUSION ALGORITHIM

Processes are organized in a logical ring: pi has a communication channel to p(i+1) mod (n).

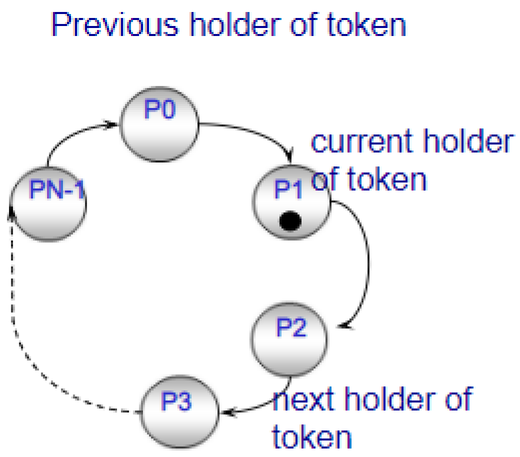
Operations:

Only the process holding the token can enter the CS.

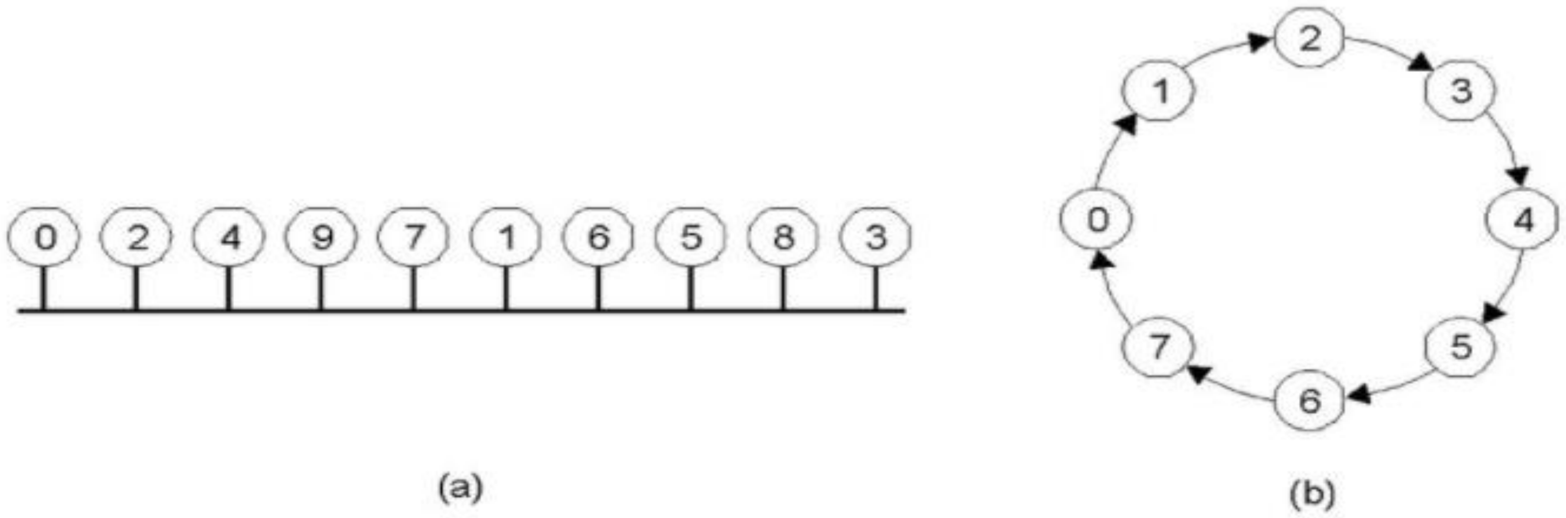
To enter the critical section, wait passively for the token. When in CS, hold on to the token.

To exit the CS, the process sends the token onto its neighbor.

If a process does not want to enter the CS when it receives the token, it forwards the token to the next neighbor.



TOKEN RING MUTUAL EXCLUSION ALGORITHM



a) An unordered group of processes on a network.

b) A logical ring constructed in software.

SUZUKI-KASAMI ALGORITHM-DATA STRUCTURE

•An array of integers **RN[1…N]** A site S_i keeps **RN$_i$[1…N]**, where **RN$_i$[j]** is the largest sequence number received so far through **REQUEST** message from site S_i.
•An array of integer **LN[1…N]** This array is used by the token. **LN[J]** is the sequence number of the request that is recently executed by site S_j.
•A queue **Q** This data structure is used by the token to keep record of ID of sites waiting for the token

SUZUKI-KASAMI ALGORITHM

•**To enter Critical section :** When a site S_i wants to enter the critical section and it does not have the token then it increments its sequence number **$RN_i[i]$** and sends a request message **REQUEST(i, sn)** to all other sites in order to request the token.
Here **sn** is update value of **$RN_i[i]$**
•When a site S_j receives the request message **REQUEST(i, sn)** from site S_i, it sets **$RN_j[i]$** to maximum of **$RN_j[i]$** and **sn** i.e **$RN_j[i]$** = max(**$RN_j[i]$**, **sn**).
•After updating **$RN_j[i]$**, Site S_j sends the token to site S_i if it has token and **$RN_j[i]$** = **LN[i]** + 1

SUZUKI-KASAMI ALGORITHM

•**To execute the critical section**:

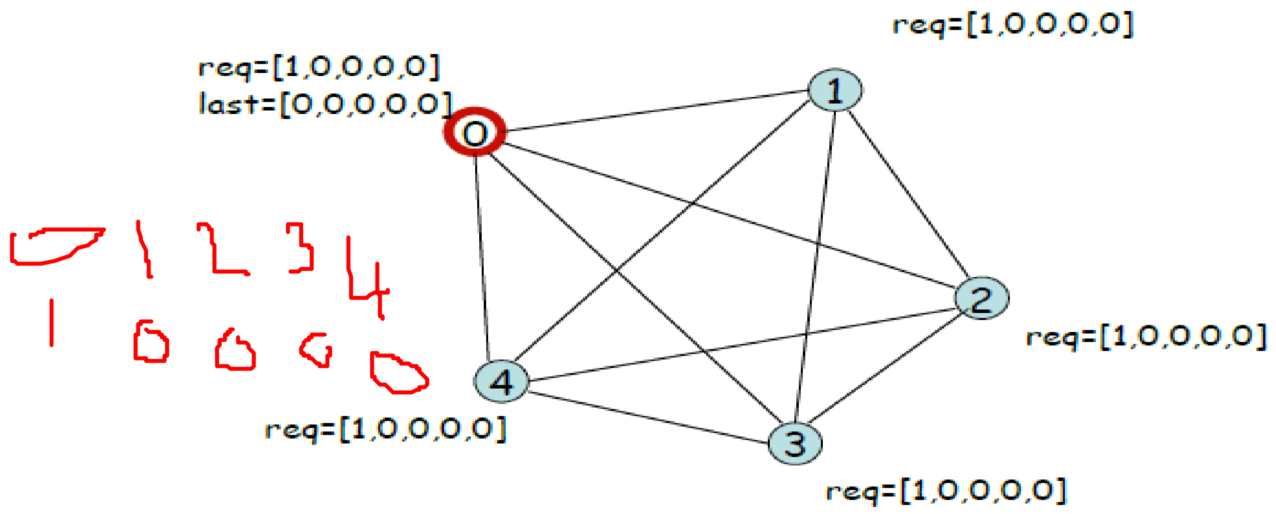
- Site S_i executes the critical section if it has acquired the token.

•**To release the critical section**:

After finishing the execution Site S_i exits the critical section and does following:

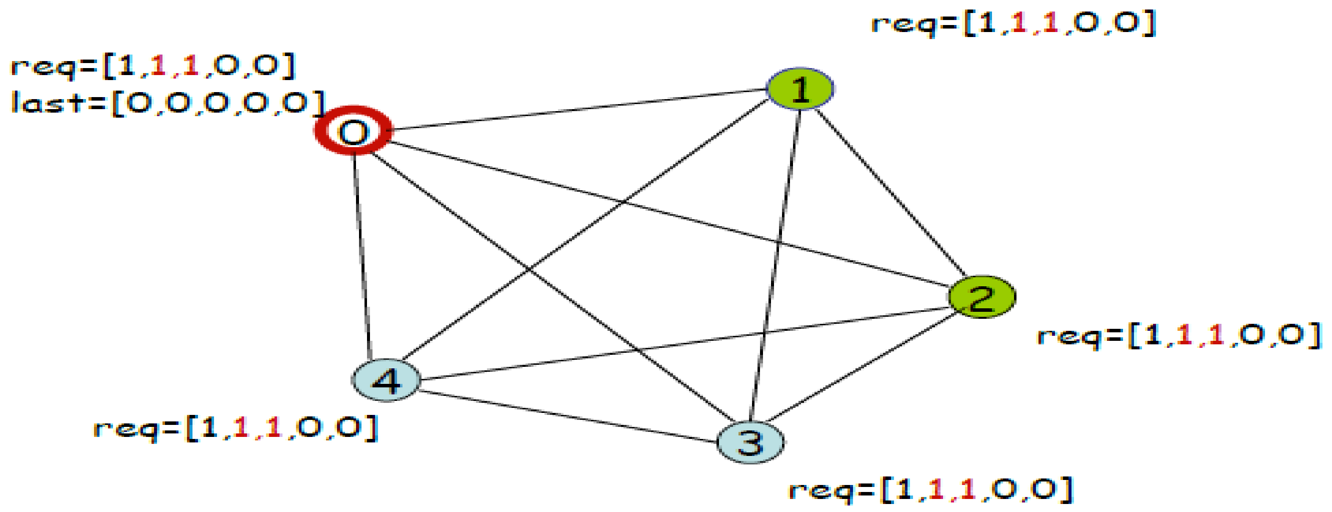
- sets **LN[i]** = **RN_i[i]** to indicate that its critical section request **RN_i[i]** has been executed
- For every site S_j, whose ID is not present in the token queue **Q**, it appends its ID to **Q** if **RN_i[j]** = **LN[j]** + 1 to indicate that site S_j has an outstanding request.
- After above updating, if the Queue **Q** is non-empty, it pops a site ID from the **Q** and sends the token to site indicated by popped ID.
- If the queue **Q** is empty, it keeps the token

SUZUKI-KASAMI ALGORITHM EXAMPLE



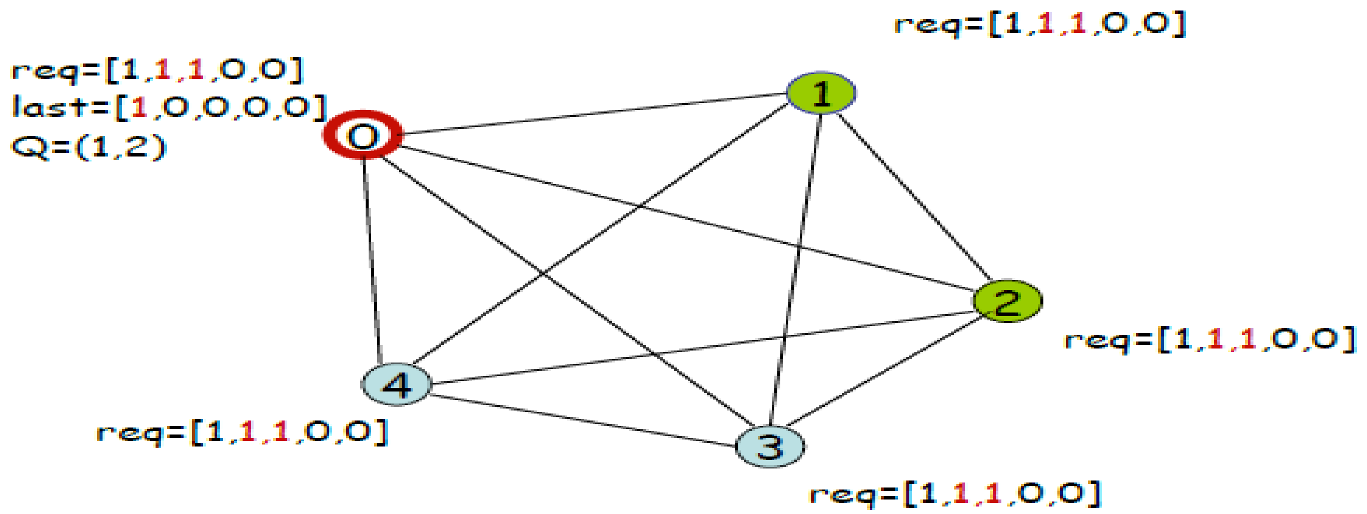
initial state: process 0 has sent a request to all, and grabbed the token

SUZUKI-KASAMI ALGORITHM



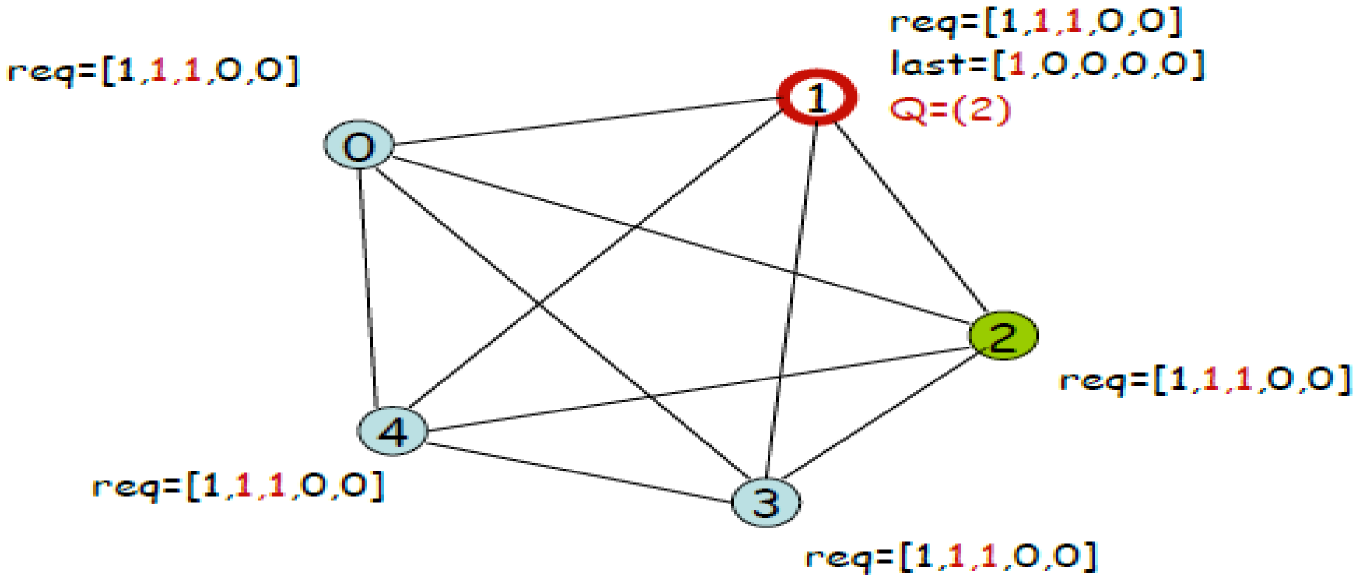
1 & 2 send requests to enter CS

SUZUKI-KASAMI ALGORITHM



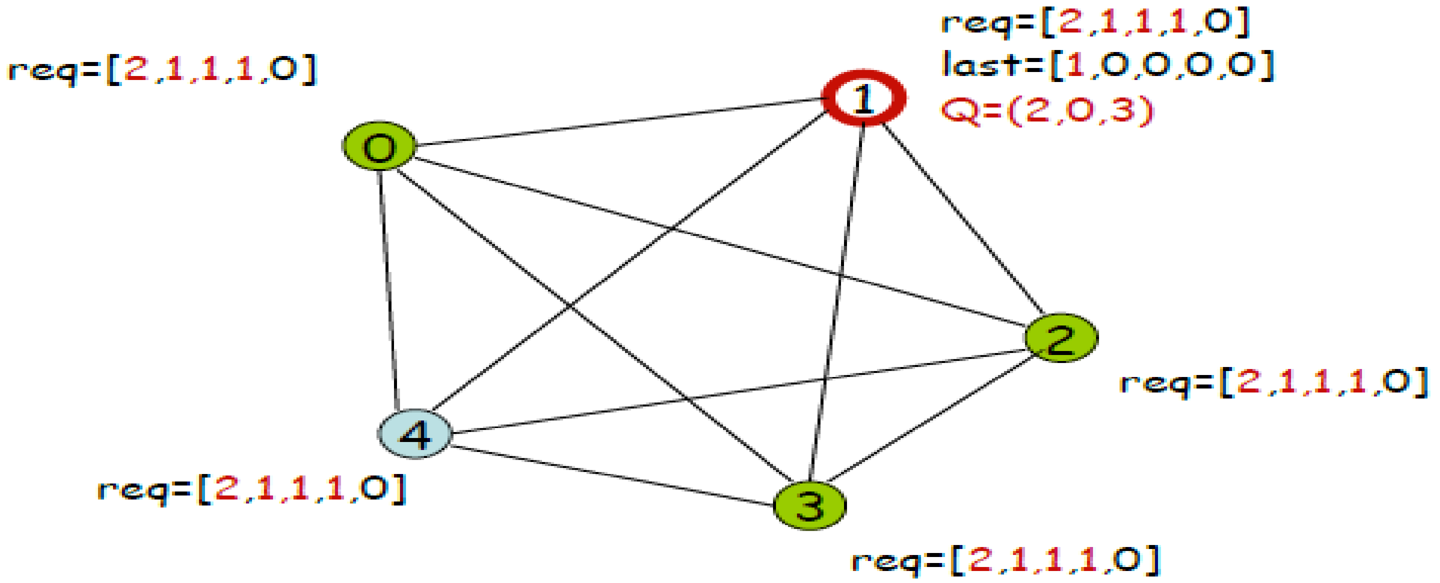
0 prepares to exit CS

SUZUKI-KASAMI ALGORITHM



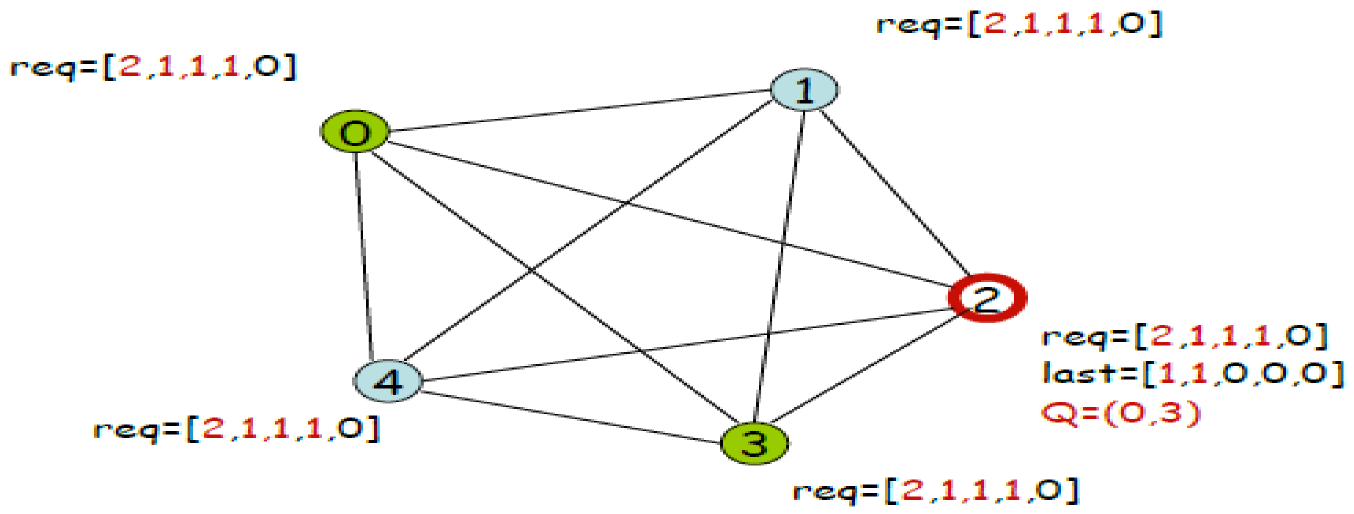
0 passes token (Q and last) to 1

SUZUKI-KASAMI ALGORITHM



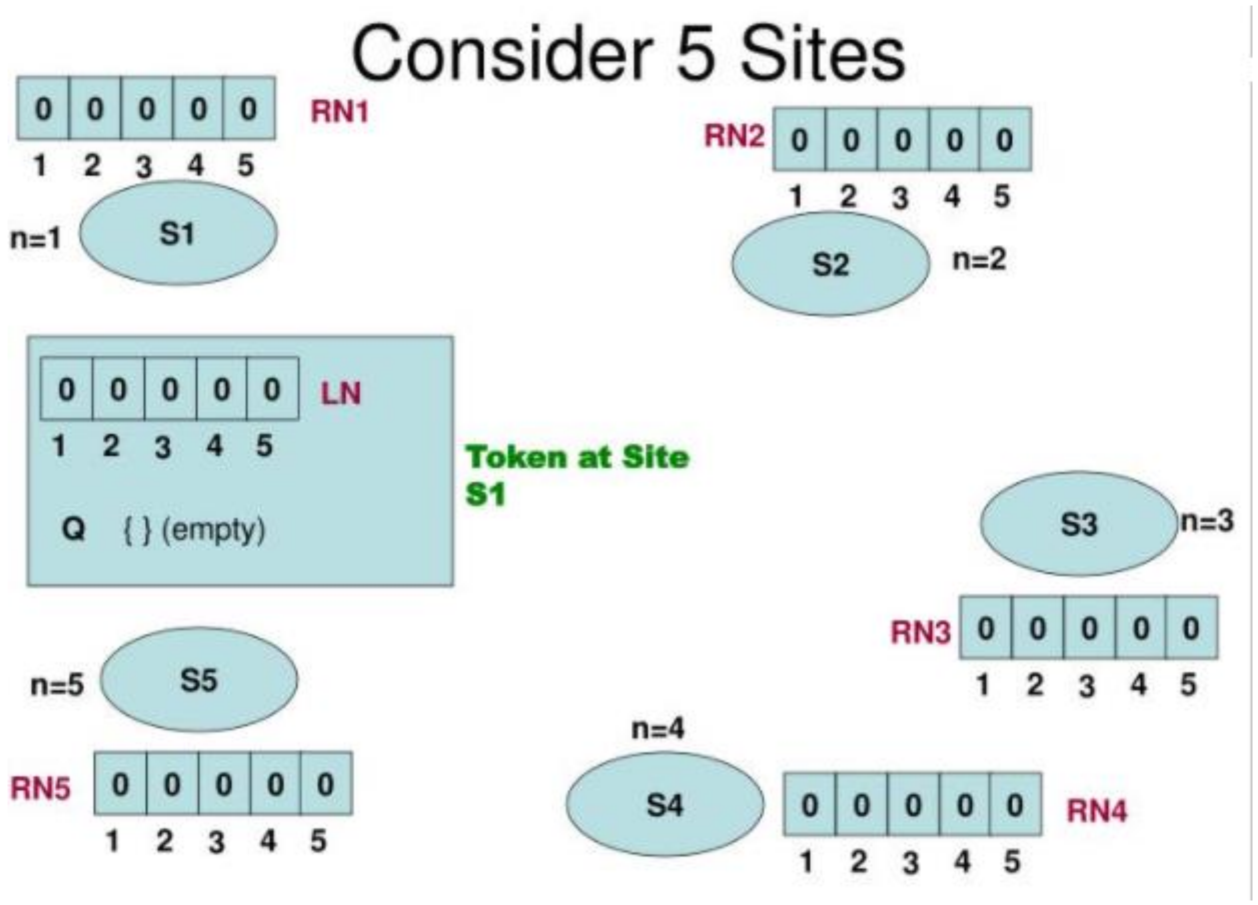
0 and 3 send requests

SUZUKI-KASAMI ALGORITHM

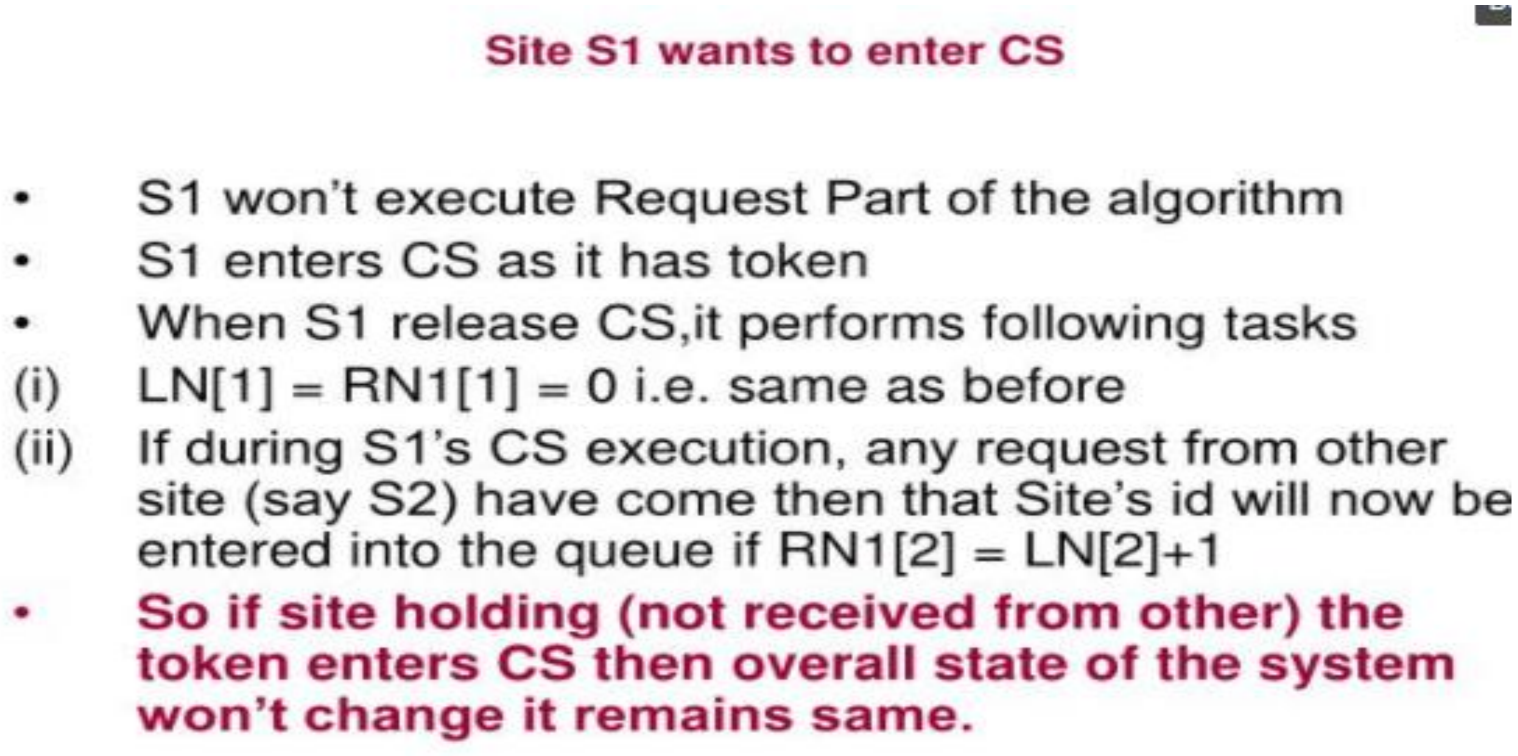


1 sends token to 2

SUZUKI-KASAMI ALGORITHM EXAMPLE-2


Consider 5 Sites
RN1 0 0 0 0 0 (1 2 3 4 5)
n=1 S1
RN2 0 0 0 0 0 (1 2 3 4 5)
S2 n=2
LN 0 0 0 0 0 (1 2 3 4 5)
Q { } (empty)
Token at Site S1
S3 n=3
RN3 0 0 0 0 0 (1 2 3 4 5)
n=5 S5
RN5 0 0 0 0 0 (1 2 3 4 5)
n=4 S4
RN4 0 0 0 0 0 (1 2 3 4 5)

SUZUKI-KASAMI ALGORITHM EXAMPLE-2



Site S1 wants to enter CS

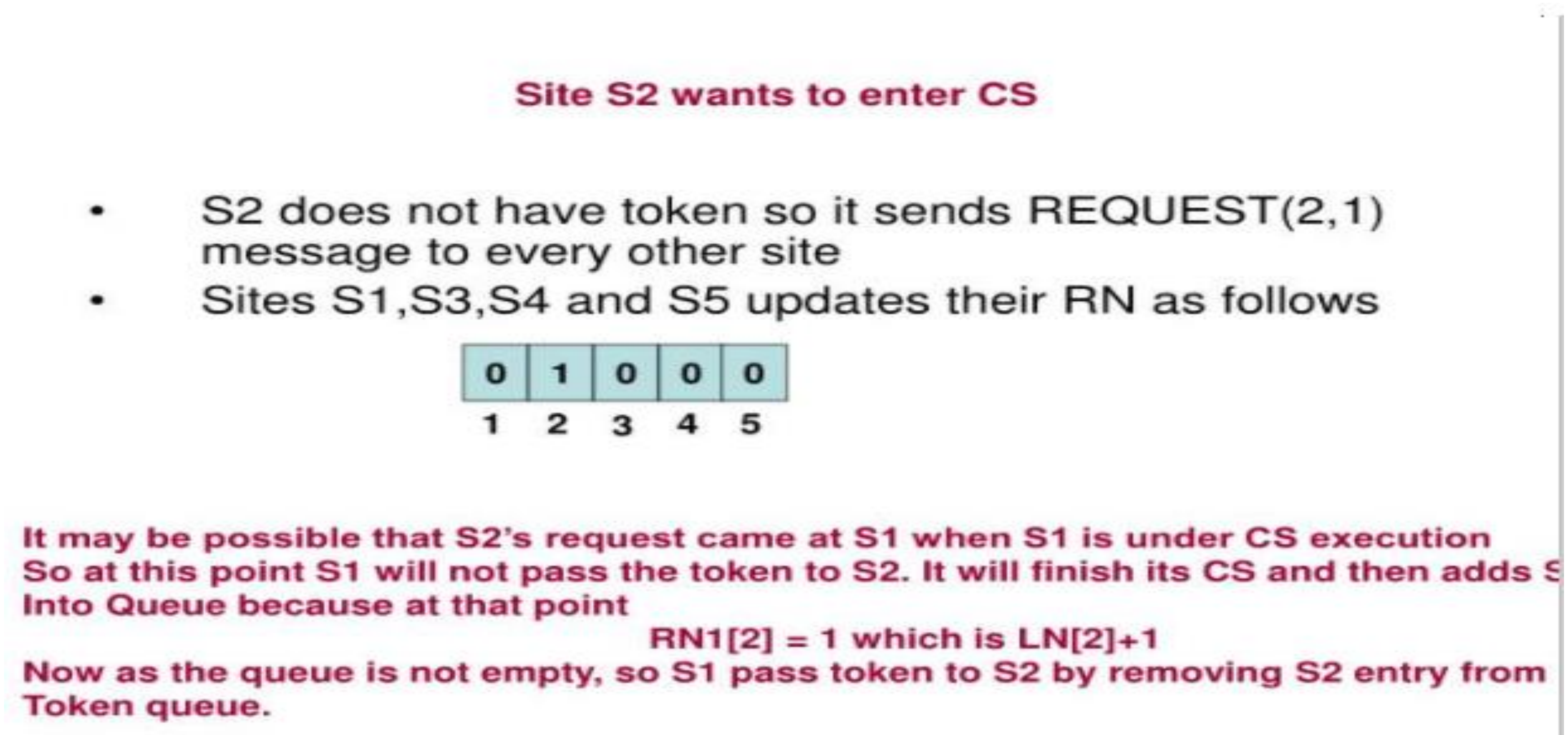
- S1 won't execute Request Part of the algorithm
- S1 enters CS as it has token
- When S1 release CS,it performs following tasks

(i) LN[1] = RN1[1] = 0 i.e. same as before

(ii) If during S1's CS execution, any request from other site (say S2) have come then that Site's id will now be entered into the queue if RN1[2] = LN[2]+1

- **So if site holding (not received from other) the token enters CS then overall state of the system won't change it remains same.**

SUZUKI-KASAMI ALGORITHM EXAMPLE-2



Site S2 wants to enter CS

- S2 does not have token so it sends REQUEST(2,1) message to every other site
- Sites S1,S3,S4 and S5 updates their RN as follows

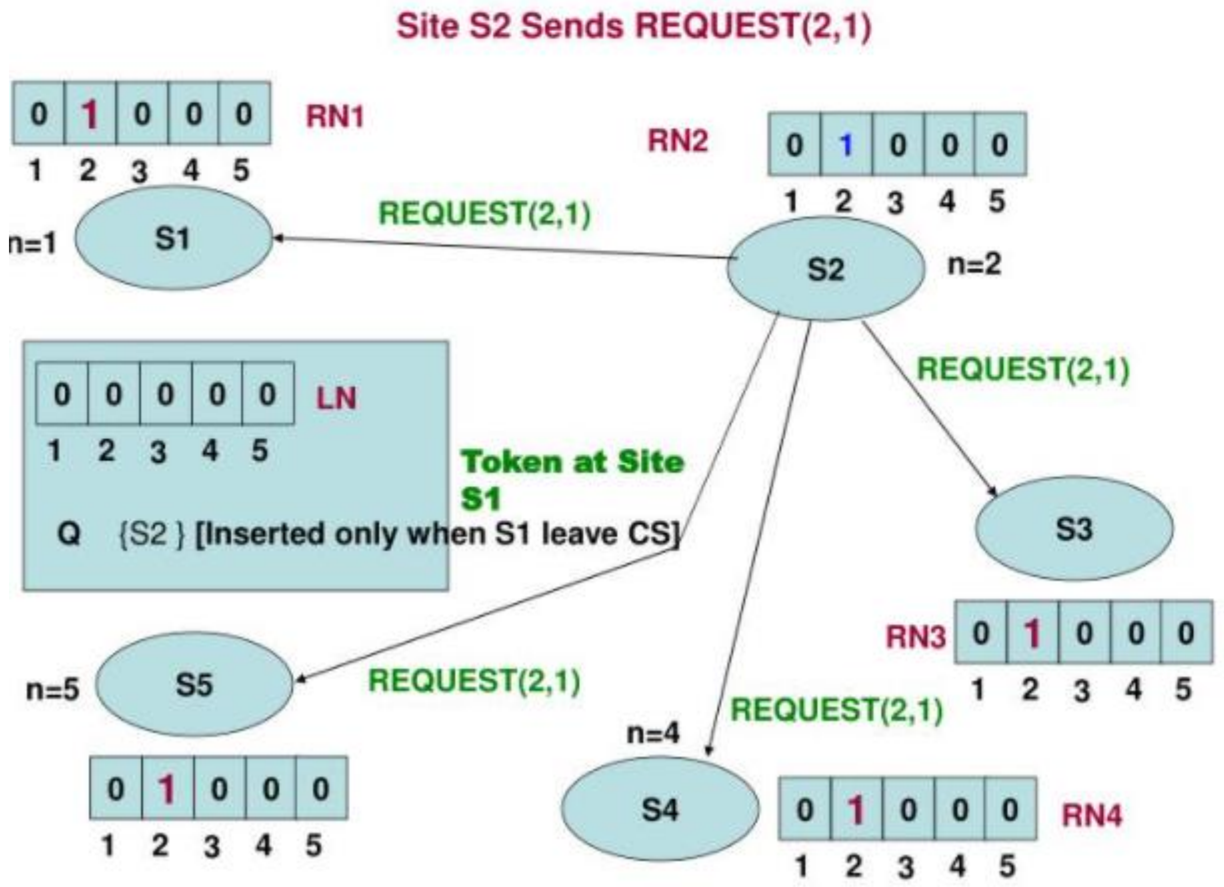
1	2	3	4	5
0	1	0	0	0

It may be possible that S2's request came at S1 when S1 is under CS execution
So at this point S1 will not pass the token to S2. It will finish its CS and then adds S
Into Queue because at that point

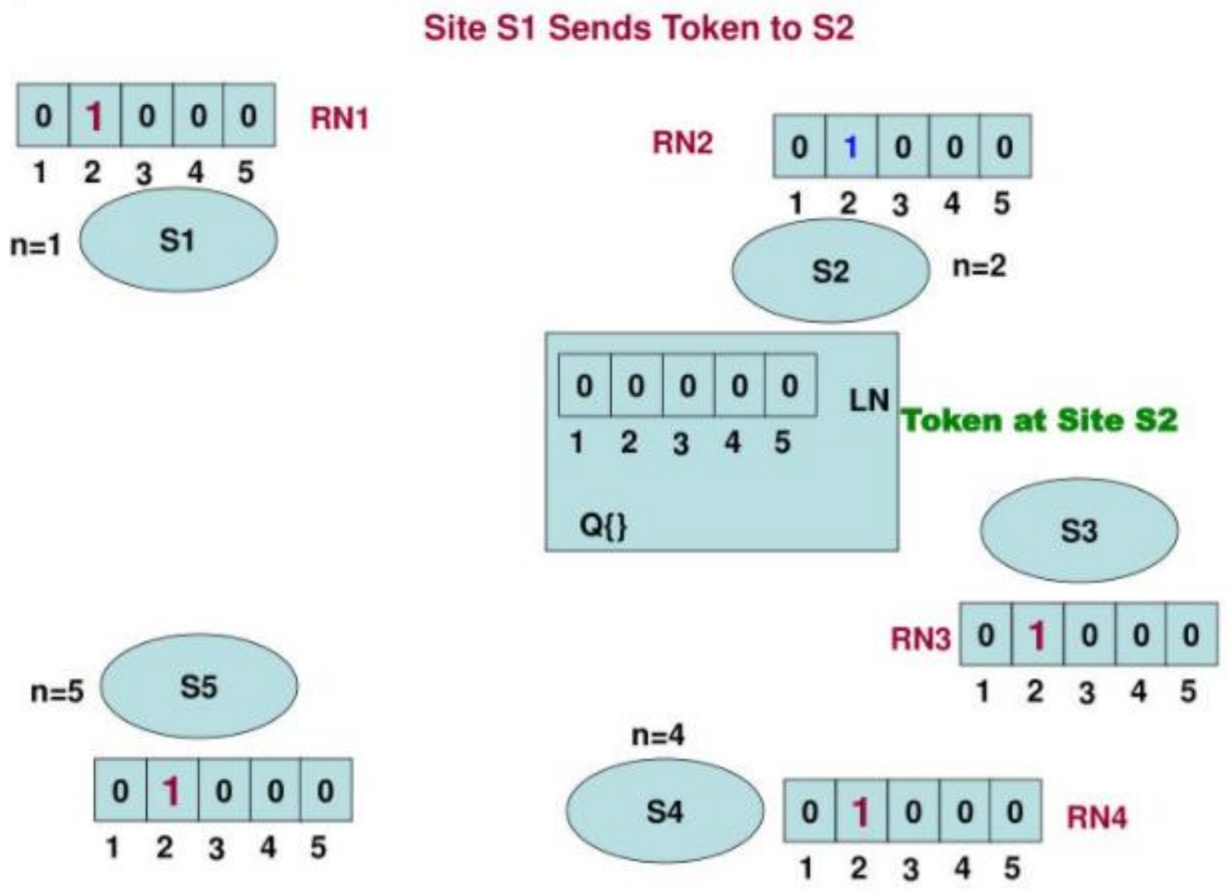
RN1[2] = 1 which is LN[2]+1

Now as the queue is not empty, so S1 pass token to S2 by removing S2 entry from
Token queue.

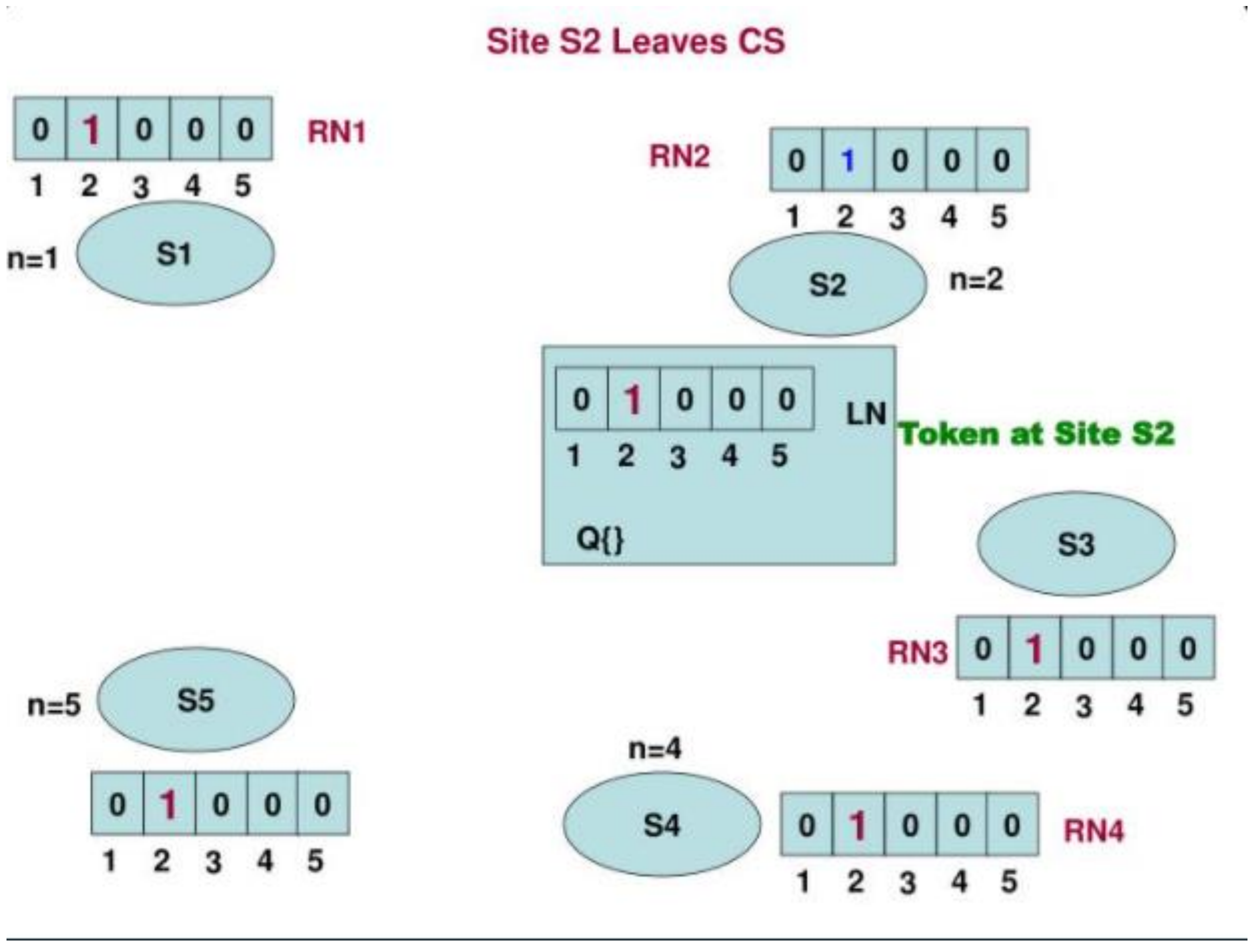
SUZUKI-KASAMI ALGORITHM EXAMPLE-2



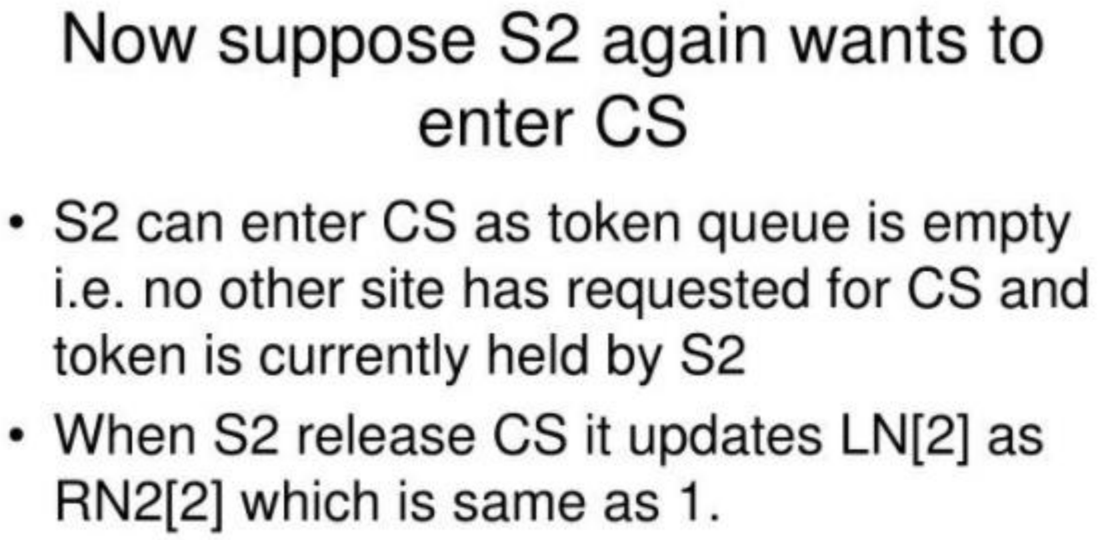
SUZUKI-KASAMI ALGORITHM EXAMPLE-2



SUZUKI-KASAMI ALGORITHM EXAMPLE-2



SUZUKI-KASAMI ALGORITHM EXAMPLE-2



Now suppose S2 again wants to enter CS

- S2 can enter CS as token queue is empty i.e. no other site has requested for CS and token is currently held by S2
- When S2 release CS it updates LN[2] as RN2[2] which is same as 1.

ELECTION ALGORITHMS

Thanks