Token-based algorithms

- A unique token is shared among all sites.
- A site is allowed to enter its CS if it possesses the token.
- Token-based algorithms use sequence numbers instead of timestamps.
- Every request for the token contains a sequence number and the sequence numbers of sites advance independently.
- A site increments its sequence number counter every time it makes a request for the token.
- A primary function of the sequence number is to distinguish between old and current requests.

Example

- SUZUKI-KASAMI'S BROADCAST ALGORITHM

Suzuki–Kasami Algorithm for Mutual Exclusion in Distributed System

- **Suzuki–Kasami algorithm** is a token-based algorithm for achieving mutual exclusion in distributed systems.

- This is modification of Ricart–Agrawala algorithm, a permission based (Non-token based) algorithm which uses **REQUEST** and **REPLY** messages to ensure mutual exclusion.

- *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.

If a site attempting to enter a CS does not have the token, it broadcasts a REQUEST message for the token to all
other sites.

A site that possesses the token sends it to the site that sends the REQUEST message.

If the site possessing the token is executing the CS, it sends the token only after it has exited the CS.

A site holding the token can repeatedly enter the critical session until it sends the token to some other site.

Datastructures used:-

- n array of integers **RN[1…N]** **// Request Array**
  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]** **// Token Array**
  This array is used by the token. **LN[J]** is the sequence number of the request that is recently executed by site S_j.
- //ie Si requests to Sj and that requests are executed in Sj
- 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 for Mutual Exclusion in Distributed System

Algorithm:

- **Requesting the Critical section:(Si requests for Sj and Sj fulfils the request of Sj by giving token which is used to access a critical section)**

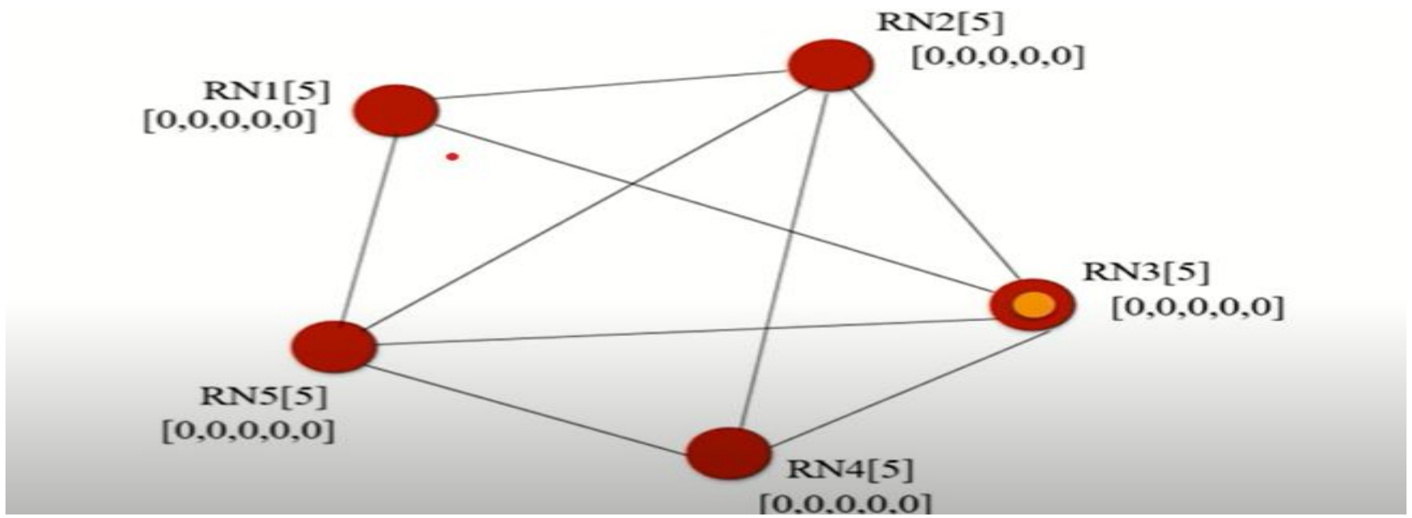
1.When a site S_i wants to enter the critical section and it does not have the token then it increments its sequence number **RN_i[i]** and sends a request message **REQUEST(i, sn)** to all other sites in order to request the token. Here **sn** is update value of **RN_i[i]**

2.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**).

3.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 for Mutual Exclusion in Distributed System

- **Executing the critical section:**
  - Site S_i executes the critical section if it has acquired the token.
- **Releasing 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 updation, 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.


RN1[5]
[0,0,0,0,0]
RN2[5]
[0,0,0,0,0]
RN3[5]
[0,0,0,0,0]
RN5[5]
[0,0,0,0,0]
RN4[5]
[0,0,0,0,0]