SUZUKI–KASAMI ALGORITHM FOR MUTUAL EXCLUSION

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.

Each requests for critical section contains a sequence number. This sequence number is used to distinguish old and current requests.

Data structure and Notations:

- An array of integers **RN[1…N]**
  A site S_i keeps $\mathbf{RN_i[1…N]}$, where $\mathbf{RN_i[j]}$ 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

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 $\mathbf{RN_i[i]}$ and sends a request message **REQUEST(i, sn)** to all other sites in order to request the token.
    Here **sn** is update value of $\mathbf{RN_i[i]}$
  - When a site S_j receives the request message **REQUEST(i, sn)** from site S_i, it sets $\mathbf{RN_j[i]}$ to maximum of $\mathbf{RN_j[i]}$ and **sn** i.e $\mathbf{RN_j[i]}$ = max($\mathbf{RN_j[i]}$, **sn**).
  - After updating $\mathbf{RN_j[i]}$, Site S_j sends the token to site S_i if it has token and $\mathbf{RN_j[i]}$ = **LN[i]** + 1
- **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]** = $\mathbf{RN_i[i]}$ to indicate that its critical section request $\mathbf{RN_i[i]}$ has been executed
  - For every site S_j, whose ID is not prsent in the token queue **Q**, it appends its ID to **Q** if $\mathbf{RN_i[j]}$ = **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

Message Complexity:
The algorithm requires 0 message invocation if the site already holds the idle token at the time of critical section request or maximum of N message per critical section execution. This N messages involves

- (N – 1) request messages
- 1 reply message

Drawbacks of Suzuki–Kasami Algorithm:

- **Non-symmetric Algorithm:** A site retains the token even if it does not have requested for critical section. According to definition of symmetric algorithm "No site possesses the right to access its critical section when it has not been requested."

Performance:

- Synchronization delay is 0 and no message is needed if the site holds the idle token at the time of its request.
- In case site does not holds the idle token, the maximum synchronization delay is equal to maximum message transmission time and a maximum of N message is required per critical section invocation.