SUZUKI–KASAMI's BROADCAST ALGORITHM

- 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.

Requesting the critical section:

(a) If requesting site S_i does not have the token, then it increments its sequence number, $RN_i[i]$, and sends a REQUEST(i, sn) message to all other sites. ("sn" is the updated value of $RN_i[i]$.)

(b) When a site S_j receives this message, it sets $RN_j[i]$ to $max(RN_j[i], sn)$. If S_j has the idle token, then it sends the token to S_i if $RN_j[i] = LN[i] + 1$.

Executing the critical section:

(c) Site S_i executes the CS after it has received the token.

Releasing the critical section: Having finished the execution of the CS, site S_i takes the following actions:

(d) It sets $LN[i]$ element of the token array equal to $RN_i[i]$.

(e) For every site S_j whose i.d. is not in the token queue, it appends its i.d. to the token queue if $RN_i[j] = LN[j] + 1$.

(f) If the token queue is nonempty after the above update, S_i deletes the top site i.d. from the token queue and sends the token to the site indicated by the i.d.

Fig : Suzuki–Kasami's broadcast 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, s_n) to all other sites in order to request the token.
- Here s_n is update value of $RN_i[i]$
- When a site S_j receives the request message REQUEST(i, s_n) from site S_i, it sets $RN_j[i]$ to maximum of $RN_j[i]$ and s_ni.e$RN_j[i] = max(RN_j[i], s_n)$.

After updating $RN_j[i]$, Site S_j sends the token to site S_i if it has token and $RN_j[i] = LN[i] + 1$

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 Si 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 prsent in the token queue Q, it appends its ID to Q if $RN_j[j] = LN[j] + 1$ 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

Correctness

Mutual exclusion is guaranteed because there is only one token in the system and a site holds the token during the CS execution.

Theorem: A requesting site enters the CS in finite time.

Proof: Token request messages of a site Si reach other sites in finite time.

Since one of these sites will have token in finite time, site S_i 's request will be placed in the token queue in finite time.

Since there can be at most N − 1 requests in front of this request in the token queue, site S_i will get the token and execute the CS in finite time.

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.

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.