



VIMAL JYOTHI
ENGINEERING COLLEGE
JYOTHI NAGAR, CHEMPERI - 670632, KANNUR D.T., KERALA
An ISO 9001:2008 Certified Institution

Assignment Cover Page

Name of the student :- SREELAKSHMI - T			
PRN ..V.M.L.22.C.S.1.69.....		Admission No...8394	
Subject Name :- DISTRIBUTED COMPUTING		Subject Code:-	
Assignment Title/No : 1			
Name of the faculty: MANJU MISS			
Assignment Submitted on			
Late submission rules : Max mark will reduced to 50% for 1-5 working day's delay, no mark will be awarded thereafter.			
I am hereby confirming that this assignment is my own and I haven't adopted any unfair means in any steps of its preparation to enhance my performance in this assignment.			
Date :		Sreelakshmi - T Sign with Name	
Assignment subdivision	Maximum Mark	Marks awarded	Remarks
A			
B			
C			
Feed back/suggestions :			
Name and sign of the faculty			

11) Suzuki-Kasami's Broadcast algorithm in IoT network.

Ans - Suzuki-Kasami's Broadcast Algorithm

It's a token based distributed mutual exclusion algorithm designed to ensure the only one process enter the critical section at a time in distributed system - such as an IoT network. It uses unique token & broadcast mechanism for requesting access to critical section (CS).

→ Basic idea :-

- * Unique token circulates among all sites
 - * A site can enter CS only if it holds the token.
 - * If it does not have token, it broadcasts a request to other sites.
 - * The site holding the token sends it to requester.
- This ensures mutual exclusion because only one token exist

→ System Model Assumption.

- * System consists of N sites (s_1, s_2, \dots, s_n)
- * Each site has one process.
- * Communication is via message passing
- * Channels are reliable & processes do not fail.

→ Data Structures Used :-

i) Request Number Array ($RN[1 \dots N]$)

- $RN[i][j]$ = largest sequence number rec'd from s_j
- Used to identify latest request & discard outdated ones.

ii) Token structure, which contains :

- $LN[1 \dots N]$ = seq. no of request from s_j (last served)
- Queue Q : Hold ID's of sites waiting for token.

⇒ Working of Algorithm:

1. Requesting the critical section.

when site S_i wants to enter cs and does not have token:-

* It increments its request number $\Rightarrow RN[i][j] + 1$

* Broadcasts REQUEST($i, RN[i][j]$) to all other sites.

2. Receiving Request Message.

• When site S_j receives REQUEST(i, n):

* It updates:- $RN[j][i] = \max(RN[j][i], n)$

* If S_j has the token and is not in cs, it checks:-

* if $RN[j][i] = LN[i] + 1$, then S_i has a pending request

* S_j sends the token to S_i

3. Executing the Critical Section

• Site enters cs only when it receives the token

• After finishing cs:

• It updates:

$LN[i] = RN[i][i]$ (marks its request as served).

4. Token Passing Rule

• After exiting cs, site checks for other pending requests:

* For each site S_j , if $RN[i][j] = LN[j] + 1$, then S_j is waiting.

* Add S_j to queue Q if not already present.

• If Q is not empty:

* Remove the site ID at the head of Q

* Send the token to that site.

→ Handling Key Design Issues: -

1. Distinguishing outdated & current requests

- Sequence numbers in REQUEST message ensures old request are ignored.
- If received number is less than stored RN value - request is outdated.

2. Identifying sites with pending requests

- Done using condition:

$$RN[j] = LN[j] + 1$$

- Ensures only sites with unserved requests are queued

→ Advantages: -

- Simple and Efficient for moderate-sized systems.
- No need for timestamps or global clock.
- Ensure mutual exclusion, fairness, and no starvation

→ Disadvantages: -

- Broadcast causes high message overhead in large networks
- Token loss can halt the system (needs recovery mechanism)
- Not ideal for highly dynamic IoT environments - with frequent node failures.