Saya Mahipala
CSE-B

Assignment-1

1) Suzuki-kasami's broadcast algorithm in IOT network.

Suzuki-kasami's algorithm is a token-based distributed mutual exclusion algorithm designed to ensure that only one process enters the Critical Section (CS) at a time in a distributed system such as an IOT network. It uses a unique token and a broadcast mechanism for requesting access to the CS.

Basic Idea

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

System Model Assumptions

- System consists of N sites ($S_1, S_2, \ldots S_n$)
- Each site has one process.
- Communication is via message passing.
- Channels are reliable and processes do not fail.

Data Structures Used

Each site maintains:

1. Request Number Array (RN[1...N])
   - RN[i][j] = largest sequence number received so far from site S_j.

- Used to identify latest (2) requests and discard outdated

2. Token Structure, which contains:
   - LN[1...N]: LN[j] = sequence number of the request from Sj that was last served.
   - Queue Q : Holds IDs of sites waiting for the token.

Working of the algorithm

1. Requesting the critical section.
   - When site si wants to enter CS and does not have the token:
     ⇒ It increments its request number
     RN[i][i] = RN[i][i] + 1
     ⇒ Broadcasts REQUEST(i, RN[i][i]) to all other sites.

2. Receiving a REQUEST Message
   When site Sj receives REQUEST(i,n):
   - It updates:
     RN[j][i] = max(RN[j][i], n)
   - If Sj has the token and is not in CS, it checks:
     - If RN[j][i] = LN[i] + 1, then si has a pending request Sj sends the token to Si.

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 (3)

- After existing CS, site checks for other pending requests.
- for each site S_j, if $RN[i][j] = LN[i]+1$, 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 Distinguishing outdated and current requests.
   - Sequence numbers in REQUEST messages ensure 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.

⇒ Disadvantages
- Broadcast causes high message overhead in large networks.
- Token loss can halt the system.