MAEKAWA'S ALGORITHM FOR MUTUAL EXCLUSION

Maekawa's Algorithm is quorum based approach to ensure mutual exclusion in distributed systems. As we know, In permission based algorithms like Lamport's Algorithm, Ricart-Agrawala Algorithm etc. a site request permission from every other site but in quorum based approach, A site does not request permission from every other site but from a subset of sites which is called **quorum**.

In this algorithm:

- Three type of messages (**REQUEST**, **REPLY** and **RELEASE**) are used.
- A site send a **REQUEST** message to all other site in its request set or quorum to get their permission to enter critical section.
- A site send a **REPLY** message to requesting site to give its permission to enter the critical section.
- A site send a **RELEASE** message to all other site in its request set or quorum upon exiting the critical section.

The construction of request set or Quorum:
A request set or Quorum in Maekawa's algorithm must satisfy the following properties:

1. $\forall i \ \forall j : i \neq j, 1 \leq i, j \leq N :: R_i \cap R_j \neq \emptyset$

   i.e there is at least one common site between the request sets of any two sites.

2. $\forall i : 1 \leq i \leq N :: S_i \in R_i$
3. $\forall i : 1 \leq i \leq N :: |R_i| = K$
4. Any site S_i is contained in exactly K sets.
5. $N = K(K - 1) + 1$ and $|R_i| = \sqrt{N}$

Algorithm:

- **To enter Critical section:**
  - When a site S_i wants to enter the critical section, it sends a request message **REQUEST(i)** to all other sites in the request set $\mathbf{R_i}$.
  - When a site S_j receives the request message **REQUEST(i)** from site S_i, it returns a **REPLY** message to site S_i if it has not sent a **REPLY** message to the site from the time it received the last **RELEASE** message. Otherwise, it queues up the request.

    .

- **To execute the critical section:**
  - A site S_i can enter the critical section if it has received the **REPLY** message from all the site in request set $\mathbf{R_i}$
- **To release the critical section:**
  - When a site S_i exits the critical section, it sends **RELEASE(i)** message to all other sites in request set $\mathbf{R_i}$
  - When a site S_j receives the **RELEASE(i)** message from site S_i, it send **REPLY** message to the next site waiting in the queue and deletes that entry from the queue
  - In case queue is empty, site S_j update its status to show that it has not sent any **REPLY** message since the receipt of the last **RELEASE** message

Message Complexity:
Maekawa's Algorithm requires invocation of $3\sqrt{N}$ messages per critical section execution as the size of a request set is $\sqrt{N}$. These $3\sqrt{N}$ messages involves.

- $\sqrt{N}$ request messages
- $\sqrt{N}$ reply messages
- $\sqrt{N}$ release messages

Drawbacks of Maekawa's Algorithm:

- This algorithm is deadlock prone because a site is exclusively locked by other sites and requests are not prioritized by their timestamp.

Performance:

- Synchronization delay is equal to twice the message propagation delay time
- It requires $3\sqrt{n}$ messages per critical section execution.