Maekawa 2.0 (Sanders Sec. 5)

Request are timestamped with Totally-Ordered Lamport Timestamps

Message types:
REQUEST, GRANT, RELEASE
FAIL, INQUIRE, RELINQUISH (process can now take back a GRANT if it has received RELINQUISH)

State:
reqQ: priority queue, ordered by timestamps
lockTS: Totally ordered Lamport Timestamp for lock holder’s request
* also keep track of messages that have been received

Initially:
reqQ is empty
lockTS = null
lockHolder = null

When \( p_i \) wants to request resource:
Send REQUEST\((C_i,i)\) to all processes in \( S_i \) (self included)
Wait for GRANT from all in \( S_i \)
Access Resource

When \( p_i \) is done with resource:
Send RELEASE to all processes in \( S_i \) (self included)

When \( p_j \) receives REQUEST\((C_i,i)\) from \( p_i \):
if lockTS = null
    Send GRANT to \( p_i \)
    lockTS = \((C_i,i)\)
    lockHolder = \( p_i \)
else
    Put \((C_i,i)\) in reqQ
    if lockTS < \((C_i,i)\)
        Send FAIL to \( p_i \) (you don’t stand a chance of getting resource right now
        so you should RELINQUISH – give up your locks)
    else
        Send INQUIRE to lockHolder (see if it will give up the lock for \( p_i \))
        Send FAIL to all processes in reqQ with timestamps > \((C_i,i)\) **
        that have not yet been sent a FAIL message
        (you are later in reqQ, so you should RELINQUISH – give up locks)
When $p_i$ receives INQUIRE from $p_j$: *(should I release the lock I am holding for you?)*

- If $p_i$ has received FAIL from any process
  - or if it has sent RELINQUISH to any process and not yet received a new GRANT,
    - send RELINQUISH to $p_j$ *(I give up my lock)*

If an INQUIRE message arrives before it is known whether $p_i$ will succeed or fail to lock all of its quorum members, a reply is deferred until this becomes known, i.e., it receives a single FAIL.

When $p_j$ receives RELINQUISH from $p_i$: *(put locked process back in queue)*

- Add lockTS to reqQ
- lockTS = null
- lockHolder = null
- if queue ≠ empty
  - $(C, k) = \text{dequeue}\text{(reqQ)}$
  - lockTS = $(C, k)$
  - lockHolder = $p_k$
  - send GRANT to $p_k$

When $p_j$ receives RELEASE from $p_i$:

- lockTS = null
- lockHolder = null
- if reqQ ≠ empty
  - $(C, k) = \text{dequeue}\text{(reqQ)}$
  - lockTS = $(C, k)$
  - lockHolder = $p_k$
  - send GRANT to $p_k$

** Missing in Maekawa paper: may lead to deadlock without it**