Maekawa 2.0 (Sanders Sec. 5)

State:
- reqQ: priority queue, ordered by Totally Ordered Lamport Timestamps
- lockHolder: who p_i has sent GRANT to by not yet received RELEASE or YIELD from
- lockTS: timestamp of lockHolder’s request
* Also, keep track of messages received

Initially:
- reqQ is empty
- lockHolder = null
- lockTS = null

When p_i wants to request resource:
- Send REQ(i, TS_i) to all processes in S_i
- Wait for GRANT messages from all processes in S_i
  - Note: if p_i sends YIELD to any process in S_i after receiving GRANT from it, it must wait for a new GRANT from that process.
- Access Resource

When p_i is done with resource:
- Send RELEASE to all processes in S_i

When p_j receives REQ(i, TS_i) from p_i:
- if lockHolder = null
  - Send GRANT to p_i
  - lockHolder = p_i
  - lockTS = TS_i
- else
  - Put request (i, TS_i) in reqQ
  - if lockTS < TS_i
    - Send FAIL to p_i
  - else
    - Send INQUIRE to lockHolder
    - Send FAIL to all processes in reqQ with timestamps > TS_i

When p_j receives RELEASE from p_i:
- lockHolder = null
- lockTS = null
- if reqQ ≠ empty
  - (lockHolder, lockTS) = dequeue(reqQ)
  - send GRANT to lockHolder

When p_j receives INQUIRE from p_i:
- If it has received FAIL from any process
  - send YIELD to p_i
When \( p_j \) receives FAIL from \( p_i \):
   If it has received \textbf{INQUIRE} from any process \( p_k \) and not yet sent \textbf{YIELD} to \( p_k \)
   send \textbf{YIELD} to \( p_k \)

When \( p_j \) receives YIELD from \( p_i \):
   Add \((\text{lockHolder}, \text{lockTS})\) to \( \text{reqQ} \)
   \text{lockHolder} = \text{null}
   \text{lockTS} = \text{null}
   \textbf{if} \ \text{reqQ} \neq \text{empty}
       \text{(lockHolder, lockTS)} = \text{dequeue(reqQ)}
       \text{send GRANT to lockHolder}