1 Lamport's Distributed Mutual Exclusion Protocol is Safe

Assume that Lamport's distributed mutual exclusion protocol is not safe. Thus there is an execution of the protocol in which two distinct processes, say P_i and P_j, are in their critical sections simultaneously, say at time t (see Figure 1). Let their critical section requests be denoted by R_i and R_j, respectively. Further, let their overlapping critical sections be denoted by CS_i and CS_j, respectively. For the critical section CS_i, we use $CS_i.begin$ and $CS_i.end$ to denote its start and finish events. We can likewise define $CS_j.begin$ and $CS_j.end$. Finally, let $\langle ts_i, i\rangle$ and $\langle ts_j, j\rangle$ denote timestamps of requests R_i and R_j, respectively.

Without loss of generality, assume that $\langle ts_i, i\rangle < \langle ts_j, j\rangle$. For P_j to enter critical section, both L1 and L2 must be true at $CS_j.begin$. For L1 to be true at $CS_j.begin$, P_j must have received a message m_i from P_i whose timestamp is greater then $\langle ts_j, j\rangle$. There are three different cases depending on when P_i sends m_i.

1. P_i sends m_i before it generates R_i: In this case, clearly, P_i's logical clock value at the time it sends m_i is at least ts_j. Thus, P_i's logical clock value at the time it generates R_i is greater than ts_j, which contradicts the assumption that $\langle ts_i, i\rangle < \langle ts_j, j\rangle$.

2. P_i sends m_i after it leaves its critical section: In this case, $CS_i.end \rightarrow m_i.send \rightarrow m_i.receive \rightarrow CS_i.begin$. This implies that CS_i ends before CS_j starts and the two critical sections do not overlap, which contradicts the assumption that P_i and P_j are in their critical sections simultaneously.

3. P_i sends m_i after it generates R_i but before leaves its critical section CS_i: In this case, at the time P_i generates R_i, it will send a REQUEST message to P_j containing the request R_i. Since all channels are FIFO, P_j will receive P_i's REQUEST message before it will receive m_i. Immediately on receiving the REQUEST message, P_j will insert $\langle ts_i, i\rangle$ into its priority queue. P_i's request will stay in P_j's queue at least until time t. This implies at the time P_j enters its critical section, P_i's request is still in P_j's queue, which contradicts with the assumption that L2 is true for P_j at $CS_j.begin$.

Since all three cases lead to a contradiction, we can conclude that Lamport's distributed mutual exclusion protocol is safe.

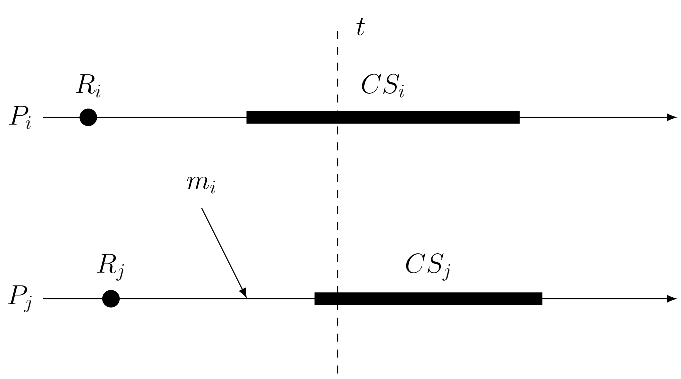

Figure 1: Two processes P_i and P_j are in their critical sections simultaneously.

2 Ricart and Agrawala's Distributed Mutual Exclusion Protocol is Safe

Assume that Ricart and Agrawala's distributed mutual exclusion protocol is not safe. Thus there is an execution of the protocol in which two distinct processes, say P_i and P_j, are in their critical sections simultaneously, say at time t (see Figure 2). We use the same notation as before.

Without loss of generality, assume that $\langle ts_i, i\rangle < \langle ts_j, j\rangle$. For P_j to enter critical section, P_j must have received the REPLY message from P_i for its request R_j. There are three different cases depending on when P_i sends the REPLY message.

1. P_i sends the REPLY message before it generates R_i: In this case, clearly, P_i's logical clock value at the time it sends the REPLY message is at least ts_j. Thus, P_i's logical clock value at the time it generates R_i is greater than ts_j, which contradicts the assumption that $\langle ts_i, i\rangle < \langle ts_j, j\rangle$.

2. P_i sends the REPLY message after it leaves its critical section: In this case, $CS_i.end \rightarrow$ REPLY$.send \rightarrow$ REPLY$.receive \rightarrow CS_i.begin$. This implies that CS_i ends before CS_j starts and the two critical sections do not overlap, which contradicts the assumption that P_i and P_j are in their critical sections simultaneously.

3. P_i sends the REPLY message after it generates R_i but before leaves its critical section CS_i: In this case, P_i sends the REPLY message while its own request is still pending (not yet completed). This contradicts the fact that, as per the protocol, if P_i has a pending critical section request, then it cannot send a REPLY message to a request with larger timestamp.

Since all three cases lead to a contradiction, we can conclude that Ricart and Agrawala's distributed mutual exclusion protocol is safe.

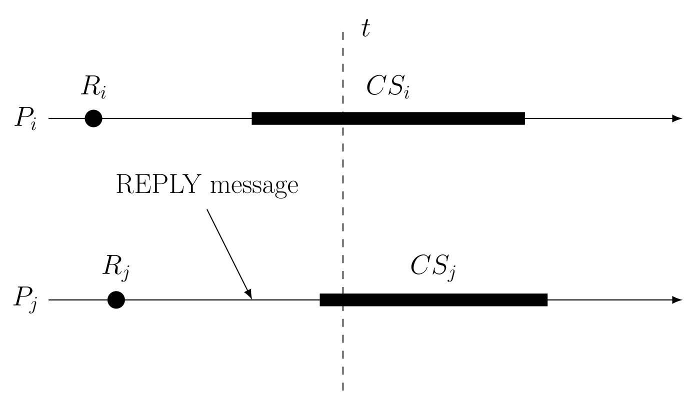

Figure 2: Two processes P_i and P_j are in their critical sections simultaneously.