Solution to Homework 2

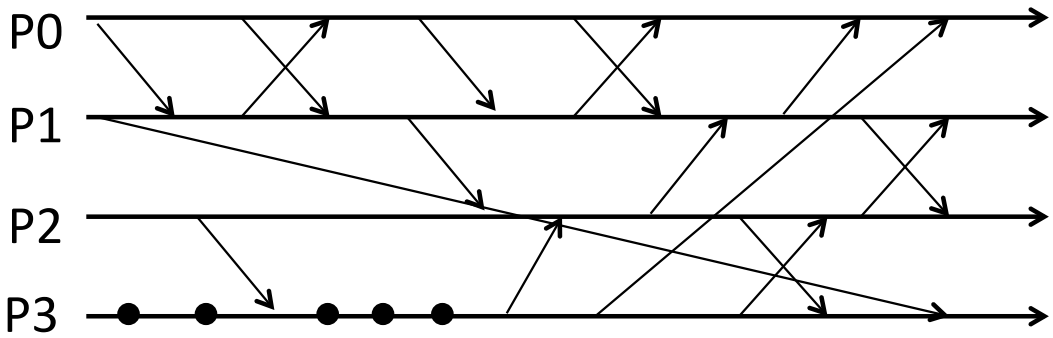
Problem 1. (Chapter 14 Problem 4 in Cbook)
A client attempts to synchronize with a time server. It records the round-trip times and timestamps returned by the server in the table below. Which of these times should it use to set its clock? To what time should it set it? Estimate the accuracy of the setting with respect to the server's clock. If it is known that the time between sending and receiving a message in the system concerned is at least 8 ms, do your answers change?

Round-trip (ms)	Time (hr:min:sec)
22	10:54:23.674
25	10:54:25.450
20	10:54:28.342

Answer: The client should choose the minimum round-trip time of 20 ms = 0.02 s as it gives the result with highest accuracy. It then estimates the current time to be 10:54:28.342 + 0.02/2 = 10:54:28.352. The accuracy is $\pm$ 10 ms.

If the minimum message transfer time is known to be 8 ms, then the setting remains the same but the accuracy improves to $\pm$ 2 ms.

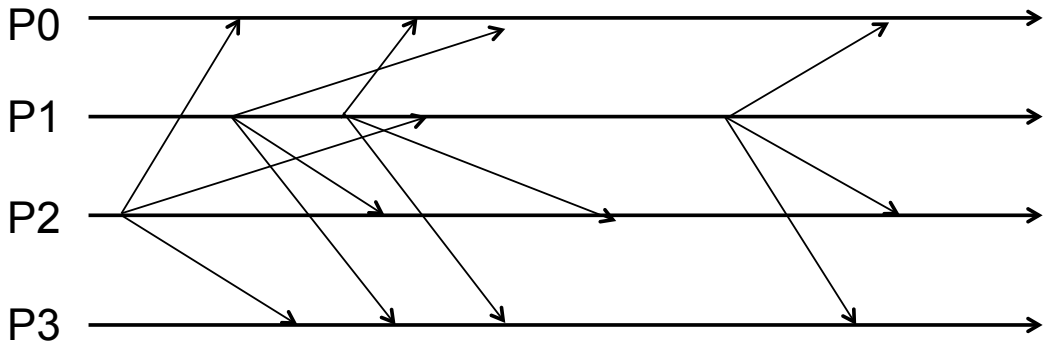
Problem 2. In the following figure, all processes, P0, P1, P2, and P3, start with Lamport timestamps or vector timestamps (as applicable) containing all zeroes. Arrows between processes denote message transmission. List the Lamport timestamps of all events.



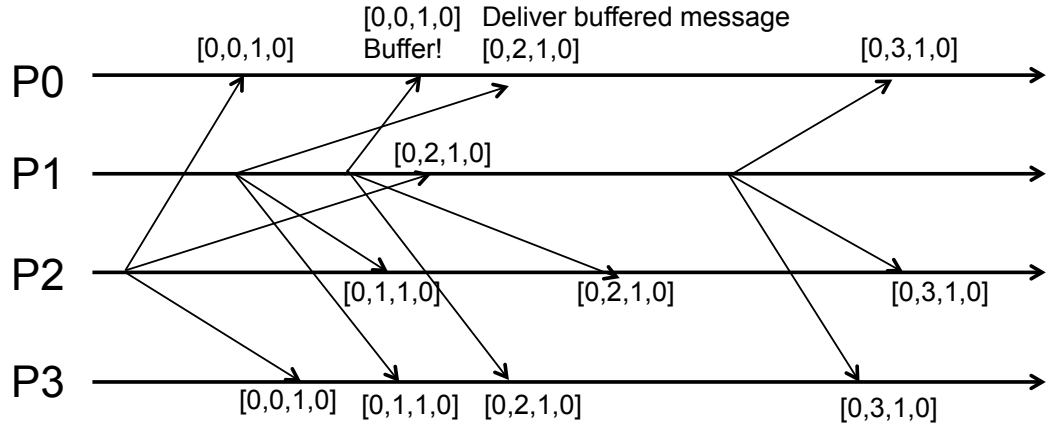
Answer:
P0: 1, 2, 4, 5, 6, 8, 12, 13
P1: 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13
P2: 1, 6, 8, 9, 10, 11, 12, 13
P3: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12

Problem 3. Repeat Problem 2 for vector timestamps instead of Lamport timestamps (List the vector timestamps of all events).
Answer:
P0: (1,0,0,0), (2,0,0,0), (3,3,0,0), (4,3,0,0), (5,3,0,0), (6,7,0,0), (7,10,4,7), (8,10,4,8)
P1: (0,1,0,0), (1,2,0,0), (1,3,0,0), (2,4,0,0), (2,5,0,0), (4,6,0,0), (4,7,0,0), (5,8,0,0), (5,9,4,7), (5,10,4,7), (5,11,4,7), (5,12,7,9)
P2: (0,0,1,0), (2,5,2,0), (2,5,3,7), (2,5,4,7), (2,5,5,7), (2,5,6,9), (2,5,7,9), (5,11,8,9)
P3: (0,0,0,1), (0,0,0,2), (0,0,1,3), (0,0,1,4), (0,0,1,5), (0,0,1,6), (0,0,1,7), (0,0,1,8), (0,0,1,9), (2,5,5,10), (2,5,5,11)

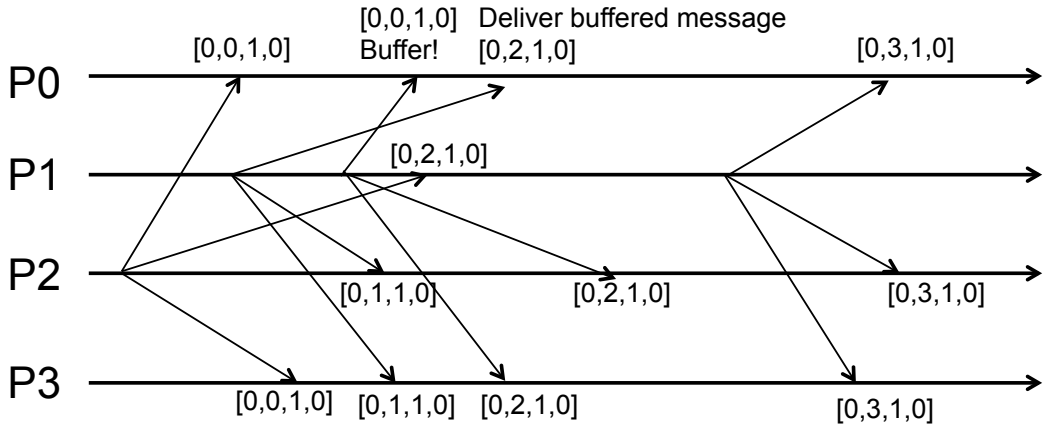
Problem 4. Given the following multicast timeline, assume that the processes use the FIFO Ordering multimcast algorithm, mark the timestamps at the point of each multicast send and each multicast receipt. Also mark multicast receipts that are buffered, along with the points at which they are delivered to the application.



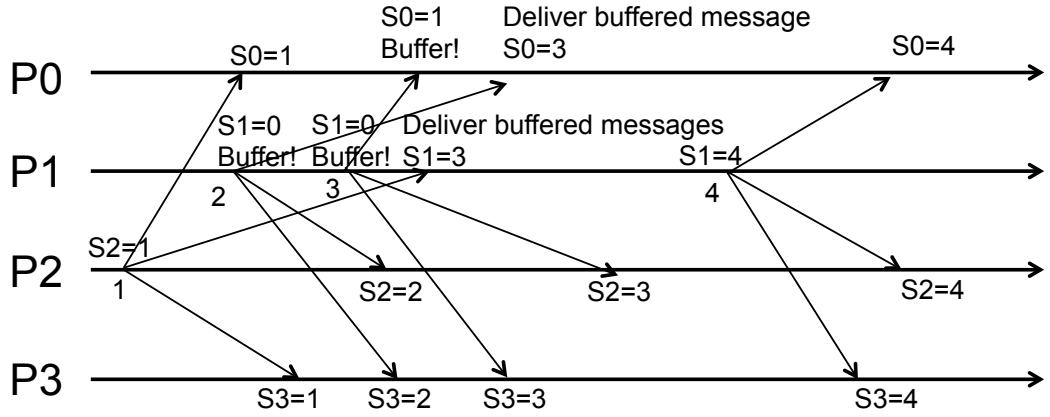
Answer:



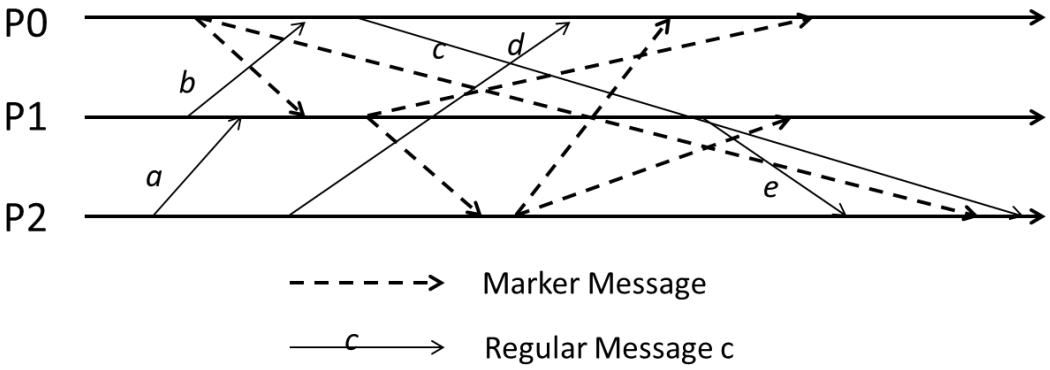
Problem 5. Repeat Problem 4 with the causal ordering multicast algorithm.
Answer:



Problem 6. Repeat Problem 4 with the total ordering multicast algorithm using a sequencer. Assuming that the sequencer receives the multicast instantaneously after the multicast is sent.
Answer:



Problem 7. In the figure below, a-e are regular application messages, with S(a) denoting the send event of a and R(a) denoting its receipt event. Markers shown as dotted lines. Note that Markers don't count as events. Use the Chandy-Lamport global snapshots algorithm to mark the entire global snapshot collected.



Answer:
P0: Initial state, C10: {b}, C20: {d}
P1: R(a), C01: {}, C21: {}
P2: S(d), C12: {}, C02: {}

Problem 8. Suppose there are N processes in the distributed system, and Ricard-Agrawala's algorithm is used for distributed mutual exclusion. How many messages are exchanged before a process can enter the critical section?
Answer: $2 \times (N - 1)$.

Problem 9. Assume that there are 16 nodes in a distributed system numbered 1, 2, ..., 16 and that Maekawa's algorithm is used to enforce mutual exclusion. Show an example situation where deadlock may occur.
Answer: Suppose nodes' quorums are constructed, and that node 1's quorum Q_1 includes nodes {1, 2, 3, 4, 5, 9, 13}, node 4's Q_4 quorum includes nodes {1, 2, 3, 4, 8, 12, 16}. If node 1 sends requests to Q_1, and node 4 sends requests to Q_4 at the same time to access the critical section at the same time, node 2 and node 3 will receive two requests at (nearly) the same time. If node 2 decides to reply to node 1 and queues the request from node 4, while node 3 decides to reply to node 4 and queues the request node 1, neither node 1 nor node 4 can proceed with the execution, and we have a deadlock.

Problem 10. Consider a modified version of the Ricart-Agrawala's algorithm that is claimed to be faster. It is faster because it does not require each process to wait for N-1 replies. Instead, it waits for only N/2 +1 replies before entering the critical section. The reasoning is that since this is a majority of processes, it is not possible that more than one process has received a majority of replies. The rest of the algorithm remains unchanged. Does this algorithm guarantee mutual exclusion?

Answer: No. In Ricart-Agrawala's algorithm, a receiving process replies as long as it is not currently in the critical section or it is requesting access but its request has lower priority. When a requesting process gets N/2 +1 replies, it only indicates N/2 +1 processes are not in the critical section or have requests with higher priority. However, it is entirely possible that some other process, who hasn't replied yet, is currently in the critical section.

Problem 11. Suppose there are 9 processes, p1 through p9. All processes know the id and addresses of all the other processes. The process with the highest id should always be the leader. Initially, p9 is the leader. However, after a power outage, all odd numbered processes (p1, p3, ...) fail. The failure of leader p9 is only detected by p4. How many messages are exchanged in order to elect the new leader using the bully algorithm?
Answer: p4 starts the election by sending the "election" message to p5, p6, p7, p8, p9 (5 messages). p4 receives "answer" messsages from p6 and p8 (2 messages). p6 sends "election" message to p7, p8, and p9 (3 messages). p6 receives "answer" message from p8 (1 message). p8 sends "election" message to p9 (1 message). After timeout, p8 elects itself as leader and sends "coordinator" message to p1, p2, p3, p4, p5, p6, p7 (7 messages). In total, 19 messages are exchanged.