Homework 2

CS425/ECE428 Spring 2023

Due: Wednesday, Mar 8 at 11:59 p.m.

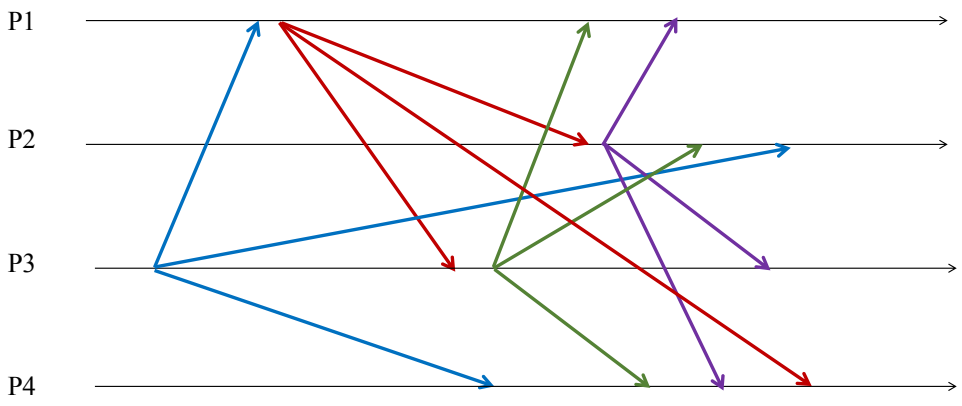

Figure 1: Figure for question 1

1. In the execution in Figure 1, processes send messages to each other to implement *causal multicast*. To simplify the picture, messages sent by each process to itself are not shown but assume that such messages are received and delivered instantaneously. For the questions below, you may use printed or hand-drawn figure with hand-drawn responses, or digitally edit the figure from the homework PDF.

   (4 points) Identify the messages that are buffered at the processes to ensure causally-ordered multicast delivery (Circle the receive event for the buffered messages to identify those messages.) For each message buffered as above, determine the earliest instant of time at which the message may be delivered, while ensuring causally-ordered multicast. (To identify the instant of time draw an arrow that begins at the time when the message is received to the time at which the message may be delivered.)

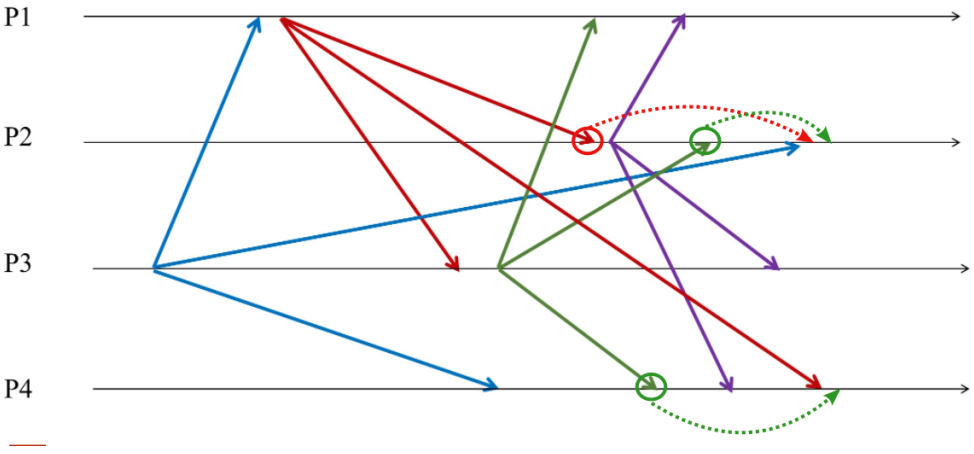

Figure 2: Figure for Q1 solution.

Solution: Three messages will be buffered as shown in Fig 2.

2. For each of the statements below, identify whether it is *true* or *false*. If it is false, present a counter-example. If it is true, prove why.

   (a) (3 points) If the processes in a system use R-multicast (where, upon receiving a message for the first time, they immediately deliver it and re-multicast it to other processes), and each *channel* follows FIFO order, then causal ordering is satisfied.

   Solution:
   True.
   Proof by contradiction: Suppose this is not true. This means that multicast of A happened before multicast of B, but A was delivered after B at some process. Consider process P that initiates multicast of B. By rules of causal ordering, P must have (i) multicast A before B if P is the original sender of both, or (ii) delivered A (and re-multicast it as per R-multicast rules) before multicasting B if A's original sender was another process. In both of these, given FIFO channels, A will be received (and immediately delivered as per R-multicast) before B by other processes in the group. Hence, contradiction.

   (b) (3 points) A FIFO + total ordered multicast is also causal ordered, given that all processes in the system communicate only via the multicast messages, and do not exchange any other messages outside of multicasting.

   Solution:
   True.
   Proof by contradiction: Consider a FIFO+total order multicast that does not imply causal order. This means that multicast of message m_i at process p_i happened before multicast of message m_j at p_j, but m_i was delivered after m_j at some process p_k. If $p_i = p_j$, FIFO order is violated which breaks our initial assumption, so p_i and p_j are different processes. For a causal ordering to be established between m_i and m_j there must be a series of messages $\{m_1, m_2, \ldots, m_n\}$ at processes $\{p_1, p_2, \ldots, p_n\}$ such that m_i is delivered before multicast (and delivery) of m_1 at p_1, m_1 is delivered before multicast (and delivery) of m_2 at p_2, ..., m_n is delivered before multicast (and delivery) of m_j at p_j. By total ordering rule, all processes must then deliver m_i before m_1 before m_2 ... before m_n before m_j. Therefore, m_i must be delivered before m_j. Hence a contradiction.

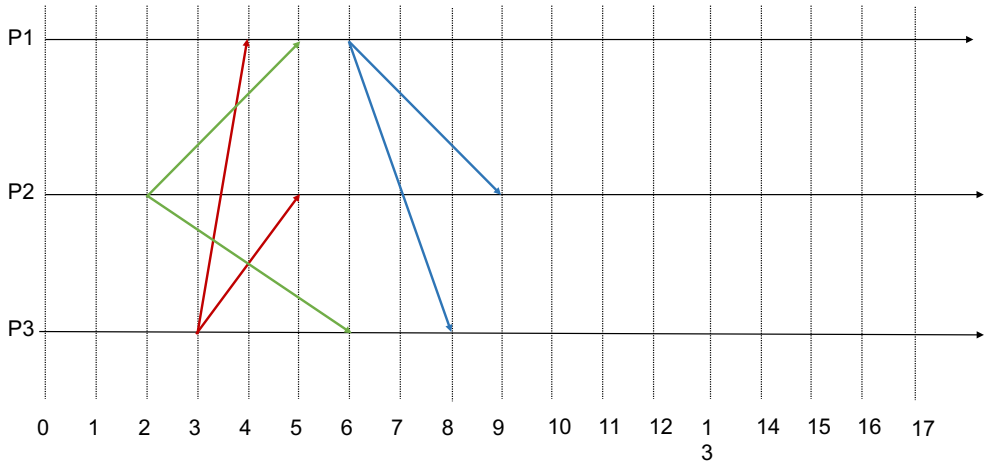

Figure 3: Figure for question 3

3. Figure 3 shows three process P1, P2, and P3 (with ids 1, 2, and 3 respectively) implementing the Ricart-Agrawala (RA) algorithm for mutual exclusion. The lines indicate requests for accessing the critical section (CS) made by each process – blue, green, and red requests are from P1, P2, and P3 respectively. Other than the replies to CS requests (not shown in the figure), no other messages are exchanged between the processes. The timeline indicates real time. Assume that any reply sent for a CS request reaches the requesting process after exactly one (real) time unit. Further assume that any process that enters the CS, spends 3 (real) time units in it.

   (4 points) At what (real) time will each process enter its critical section?

   Solution: P1 will enter the critical section at 15th time unit.

   P2 will enter the critical section at 7th time unit. P3 will enter the critical section at 11th time unit.

4. Consider a run of the Bully algorithm, with 6 processes. Initially all 6 processes are alive and P_6 is the leader. Then P_6 fails and P_3 detects this and initiates the election. Assume no other process fails during the election run, and that there is no processing delay. Also assume that P_3 is the only node that knows that P_6 has failed (and therefore does not send any messages to it). Other nodes are not aware of P_6's failure. One-way transmission time is $T = 10ms$ (and timeout values set using the knowledge of T).

   (a) (1 point) When will the election finish?

   Solution: i) P3 sends "Election" messages to P4 and P5 which takes 1 T time

   ii) P4, sends a "Disagree" message to P3 and also simultaneously start sending "Election" messages to P5 and P6. P5 sends a "Disagree" message to P3 and also simultaneously starts sending "Election" messages to P6.This set of events happens in 1 T time.

   iii) Now P5 sends a "Disagree" message to P4 and also waits for P6's response.This will take 1 T time.

   iv) Now P5 will announce itself as leader and send coordinator messages to P1, P2, P3, and P4. This will take 1 T time.

   So a total of 4 T time will be taken, which in this case will be 4*10ms = 40 ms

   (b) (5 points) How many total messages will be sent and received by each process in above scenario?

Solution:

Process	Sent	Received
P_1	0	1
P_2	0	1
P_3	2	3
P_4	3	3
P_5	7	2

5. Consider a system of five processes $[P_1, P_2, P_3, P_4, P_5]$. Each process P_i proposes a value x_i. Let $x_1 = 3$, $x_2 = 10$, $x_3 = 6$, $x_4 = 7$, and $x_5 = 5$.

   Each process P_k must decide on an output variable y_k (initialized to *undecided*), setting it to one of the proposed values x_i for $i \in [1, 5]$. The safety condition requires that at any point in time, for any two processes P_j and P_k, either y_j or y_k is *undecided*, or $y_j = y_k$ (in other words, the decided value must be same across all processes that have decided).

   A consensus algorithm is designed for the above problem that works as follows:

   - Each process R-multicasts its proposed value at the same time $t = 0ms$ since start of the system (as per their local clocks).
   - As soon as proposed values from all 5 processes are delivered at a process P_j, P_j sets y_j to the minimum of the proposed values it received from the five processes.
   - If y_j is still *undecided* at time $(t + timeout)$, P_j computes the minimum of the proposed values it has received so far and sets y_j to that value.
   - Once a process P_j decides on y_j, it does not update y_j's value, and ignores future proposals (if any are received).

   Assume that all clocks are perfectly synchronized with zero skew with respect to one-another. The proposed value x_i of a process P_i gets self-delivered immediately at time $t = 0ms$ when P_i begins the multicast of x_i. A message sent from a process to any other process takes exactly $T = 20ms$ (and this value is known to all processes). All communication channels are reliable. Processes may fail, but a failed process never restarts.

   Suppose the *timeout* value for the above algorithm is set to $50ms$. Answer the following questions with respect to local time at the processes' clock since the start of the system.

   (a) (1 point) Assume no process fails in the system. When will each process decide on a value and what will each of their decided values be?

   Solution:
   In 20 ms, Every process will decide on the value 3, as it is the minimum among all proposed values.

   (b) (1 point) Assume P_1 fails right after unicasting x_1 to P_2 and P_3 but just before it could initiate the unicast of x_1 to any of the other processes. When will each of the alive processes decide on a value and what will each of their decided values be?

   Solution: P2 and P3 will decide on the value of 3 at 20 ms. P4 and P5 will decide on the value of 3 at 40 ms.

   (c) (2 points) Assume P_1 fails at right after unicasting x_1 to P_2 but just before it could initiate the unicast of x_1 to any of the other processes. P_2 fails right after it has relayed x_1 to P_3 but just

before it unicasts it to any other process. When will each of the alive processes decide on a value and what will each of their decided values be?

Solution: P3 will decide on the value of 3 at 40 ms. P4 and P5 will decide on the value of 5 at 50 ms because they will timeout.

(d) (2 points) Assume P_5 fails right before it could unicast x_5 to any process. P_1 fails right after unicasting x_1 to P_2 but just before it could initiate the unicast of x_1 to any of the other processes. P_2 fails right after it has relayed x_1 to P_3 but just before it unicasts it to any other process. When will each of the remaining alive processes decide on a value and what will each of their decided values be?

Solution: P3 will decide on the value of 3 at 50 ms because it will time out. P4 will decide on the value of 6 at 50 ms because it will time out.

(e) (2 points) What is the smallest value that the *timeout* should be set to for ensuring safety in this system?

Solution: The minimum timeout should be 80 ms.
Say P1 fails after sending x1 to P2, which will take 20 ms.
P2 fails after relaying x1 to P3, which will take 20 ms.
P3 fails after relaying x1 to P4, which will take 20 ms.
P4 relays x1 to P5, which will take 20 ms.

In this scenario, the alive processes P4 and P5 will agree on the same value, so the timeout should be 80 ms.

(f) (2 points) Answer Q5c assuming that the timeout is updated to the value in Q5e.

Solution: P3 will decide on the value 3 at 40 ms. P4, P5 will decide on the value 3 at 60 ms.

(g) (2 points) Answer Q5d assuming that the timeout is updated to the value in Q5e.

Solution: P3 will decide 3 at 80 ms as it will timeout waiting for x5. P4 will decide 3 at 80 ms as it will timeout waiting for x5.

6. Consider a system of five processes that implement the Paxos algorithm for consensus. As shown in Figure 4, P1 and P2 are proposers. P3, P4, P5 are acceptors. P1 sends a *prepare* message with proposal number 2 to processes P3 and P5, receives their replies, and sends an *accept* message with proposed value of 20 for proposal #2. P2 concurrently sends a *prepare* message with proposal #6, with an initial intention to propose a value of 25 if it receives sufficient replies. Only a subset of responses from processes P3 and P5 are shown in the figure. Assume no other proposals are initiated. Answer the following sub-questions.

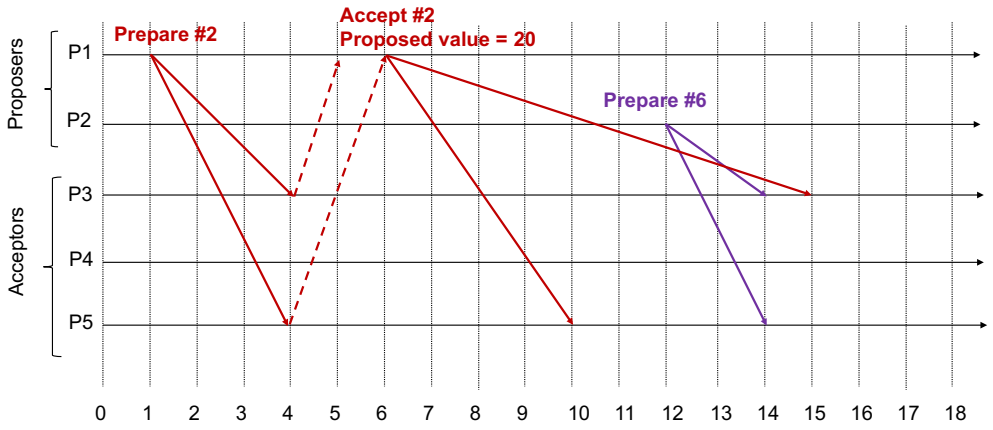

Figure 4: Figure for question 6

(a) (1 point) Which processes will *accept* P1's proposal?

Solution: P5 will accept P1's proposal.

(b) (1 point) Which processes will reply back to P2's *prepare* message?

Solution: Both P3 and P5 will reply to P2's proposal. P5 replies with a value of 20 where as P3 replies without any value.

(c) (2 points) Will P2 send out an *accept* message for its proposal #6? If yes, what will be the corresponding proposed value?

Solution: As mentioned in solution 5.b, P2 will get messages from P5 with a value of 20 and from P3 without any value. So P2 will pick the value 20 according to the algorithm.

(d) (2 points) Consider the state of the system at time 11 units. Has the proposed value 20 been implicitly decided upon at this point? If yes, explain why? If not, construct a possible scenario that may occur after time 11 units where the system ends up deciding on a different proposed value. The scenario would involve a temporal sequence of events that may differ from the one shown in the figure beyond time t=11units but must comply with what is shown until t=11units. An event in such a sequence may include a prepare/accept request sent by a proposer or received by an acceptor, or a prepare reply received by the proposer at some time unit.

Solution: No, because if P2's prepare message is sent to P3 and P4, the system's accepted value is 25. You can see Fig 5.

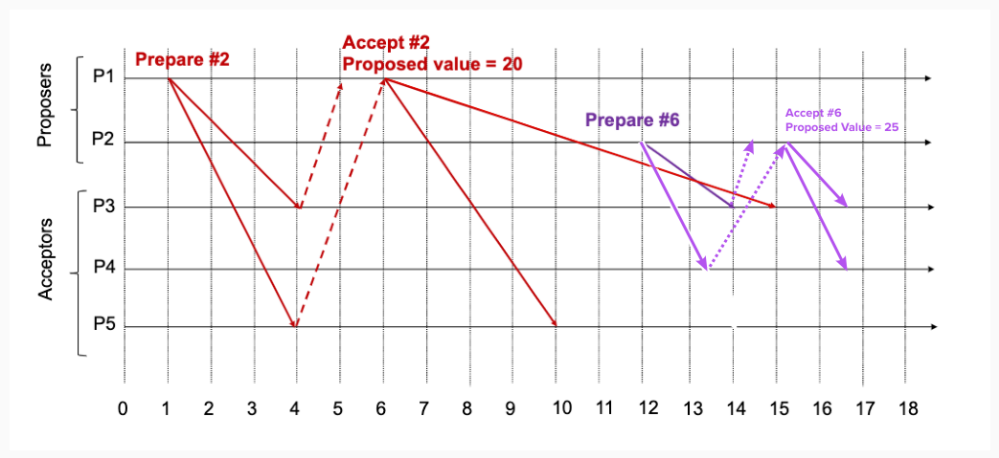

Figure 5: Figure for 6d where system's accepted value is 25

(e) (1 point) Suppose that P1's *accept* request reaches P3 at time 9 units (instead of 15 units). If we now consider the state of the system at time 11 units, has the proposed value of 20 been implicitly decided upon?

Solution:
The value is implicitly fixed now, so no matter how we send the prepare messages of P2, the value will not change.

(f) (1 point) Suppose that P1's *accept* request reaches P5 at time 16 units (instead of 10 units) and reaches P3 at the original time 15 units. Will P2 send out an *accept* message for its proposal #6? If yes, what will be the corresponding proposed value?

Solution: Yes, and the corresponding accepted value will be 25.