Homework 2

CS425/ECE428 Spring 2026

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

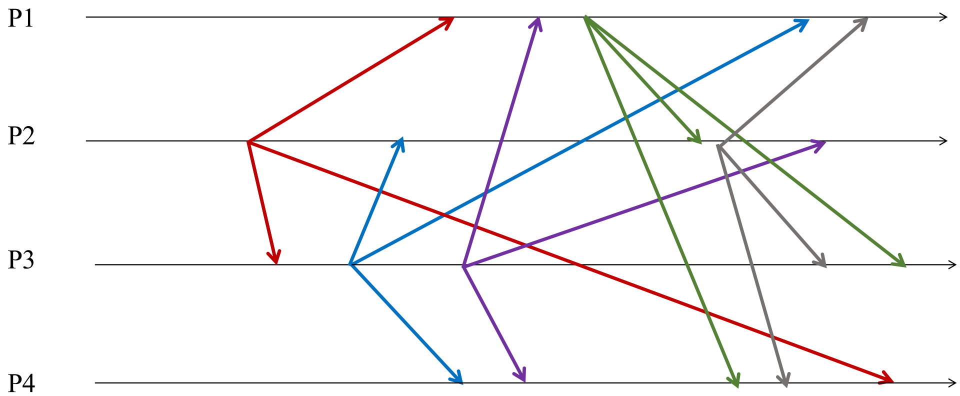

Figure 1: Figure for question 1

1. In the execution in Figure 1, four processes form a multicast group, sending messages to each other. To simplify the picture, messages sent by each process to itself are not shown but assume that such messages are received *and delivered* instantaneously (at the same time when the sender issues the multicast). As response to the question below, you may use printed or hand-drawn figure with hand-drawn responses, or digitally edit the figure from the homework PDF.

   (a) (3 points) Identify the messages that are buffered at the processes to ensure *FIFO-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 FIFO-ordered multicast. (To identify the instant of time draw an arrow that begins at the time when the message is received by the multicast protocol at the process to the time at which the message may be delivered to the application. It's ok for you to draw somewhat overlapping arrows for buffered messages that get delivered around the same time – your arrows need not indicate the specific ordering among them.)

   (b) (6 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 by the multicast protocol at the process to the time at which the message may be delivered to the application. It's ok for you to draw somewhat overlapping arrows for buffered messages that get delivered around the same time – your arrows need not indicate the specific ordering among them.)

2. (4 points) Consider an execution of ISIS algorithm for total-ordered multicast. The following lists a subset of messages in a process's queue, with their corresponding priorities:

   - Message A: proposed priority 3
   - Message B: proposed priority 4
   - Message C: final priority 7
   - Message D: proposed priority 10
   - Message E: final priority 15

   In the final total ordering of the messages above, which of the following could be true (select all that apply by writing the appropriate choice numbers):

   (i) A gets delivered before B.

   (ii) A gets delivered after B.

   (iii) A gets delivered after D.

   (iv) A gets delivered after E.

   (v) C gets delivered before B.

   (vi) C gets delivered after B.

   (vii) C gets delivered after D.

   (viii) C gets delivered after E.

3. Consider a distributed system of five processes $\{P_1, P_2, P_3, P_4, P_5\}$. Each process needs mutually exclusive access to a critical section. Assume that each process wishes to enter a critical section only once. Table 1 lists the time when each process first makes a blocking call to "enter" the critical section (since the start of the system). It also lists the time each process spends in the critical section after "enter" returns, before calling "exit".

Process ID	Time when "enter" is called (since start of system)	Time spent in critical section after "enter" returns, before calling "exit"
P_1	100ms	40ms
P_2	30ms	30ms
P_3	5ms	25ms
P_4	200ms	10ms
P_5	25ms	5ms

Table 1: Timings for Q3

For each of the subparts below, assume that the one-way network delay between any two different processes is fixed at 10ms (i.e. it takes exactly 10ms for a message to go from P_i to P_j, when $i \neq j$). The network delay for any message that a process P_i sends to itself is zero. Other than the network delays and the time spent in critical section, assume all other processing takes negligible amount of time.

(a) (5 points) Suppose the system uses the central server algorithm for mutual exclusion, electing P_2 as the leader. The leader grants requests in the order in which it receives them. When will each process start executing its critical section?

(b) (5 points) Now suppose that the system uses ring-based algorithm for mutual exclusion, with the ring structured as shown below (P_1 to P_2 to P_3 to P_4 to P_5 to P_1).

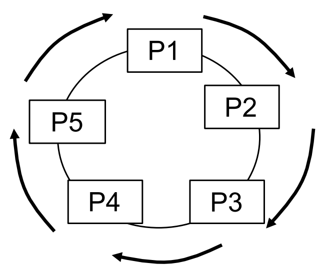

Figure 2

At time 0ms (when the system starts up), the token is at P_1. As specified above, the network delay for passing the token from a given process to its ring successor is 10ms. When will each process start executing its critical section?

(c) (5 points) Now suppose the processes use the Ricart-Agrawala algorithm for mutual exclusion. Assume all processes start off with a Lamport timestamp of zero, and no other events occur at the processes outside of the events that are part of the algorithm. As mentioned above, the one-way network delay between any two different processes is 10ms. The id of process P_i is i (this is used for lexicographical tie-breaking when two requests have the same Lamport timestamps). When will each process start executing its critical section?

4. Consider a system of 8 processes with ids 1 to 8 (i for P_i), that uses the Bully algorithm for leader election. The process with the highest id among the alive processes must be elected as the leader. Initially, all 8 processes are alive and P_8 is the leader. Then P_8 fails. P_3 detects this and initiates an election run at time 500ms from the start of the system. P_3 is the only process that knows of P_8's failure, and other processes are not aware of P_8's failure.

   Assume a failed process does not recover again, no messages are dropped, and the one-way message transmission time between any pair of processes is exactly 16ms. The first timeout (that is set after sending an "election" message) is accordingly set to 32ms, and the second timeout (that is set after receiving a "disagree" message) is set to 64ms. The processing time at each process is negligible.

   An election finishes when all the alive processes know who the new leader is.

   Respond to the following questions.

   (a) (1 point) If no other process fails, when will the election finish? Your response should be in terms of time in milliseconds since the start of system (so greater than 500ms, since the election starts at time 500ms).

   (b) (7 points) How many total messages will be sent and received by each alive process during the election run in the above scenario (i.e. when no other process fails)?

Process	**Sent**	**Received**
P_1		
P_2		
P_3		
P_4		
P_5		
P_6		
P_7		

   (c) (2 points) Now suppose that P_7 fails during the election at time 524ms. When will the election finish? Your response should be in terms of time in milliseconds since the start of system (so greater than 500ms, since the election starts at time 500ms).

   (d) (2 points) Next suppose that P_7 fails during the election at time 540ms. When will the election finish? Your response should be in terms of time in milliseconds since the start of system (so greater than 500ms, since the election starts at time 500ms).