Homework 3

CS425/ECE428 Spring 2026

Due: ~~Friday, March 27~~ Monday, March 30 at 11:59 p.m.

1. **Synchronous Consensus**

   Consider a system of six processes $[P_1, P_2, P_3, P_4, P_5, P_6]$. Each process P_i proposes a value x_i. Let $x_1 = 10$, $x_2 = 5$, $x_3 = 6$, $x_4 = 15$, $x_5 = 9$, and $x_6 = 3$.

   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 $i \in [1, 6]$. 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 6 processes are delivered at a process P_j, P_j sets y_j to the maximum of these six proposed values.
   - If y_j is still *undecided* at time $(t + timeout)$, P_j computes the maximum 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?

   (b) (1 point) Assume P_4 fails right after unicasting x_4 to P_5 and P_6 but just before it could initiate the unicast of x_4 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?

   (c) (2 points) Assume P_4 fails at right after unicasting x_4 to P_5 but just before it could initiate the unicast of x_4 to any of the other processes. P_5 fails right after issuing unicast of x_4 to P_6 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?

   (d) (2 points) Assume P_1 fails right before it could unicast x_1 to any process. P_4 fails right after unicasting x_4 to P_5 but just before it could initiate the unicast of x_4 to any of the other processes. P_5 fails right after issuing unicast of x_4 to P_6 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?

   (e) (1 point) What is the smallest value that the *timeout* should be set to for ensuring safety in this system? [Hint: You need to think about the maximum number of process failures the system must tolerate, and compute timeout based on that, for ensuring safety of consensus.]

   (f) (1 point) Answer Q1c assuming that the timeout is updated to the value in Q1e.

   (g) (1 point) Answer Q1d assuming that the timeout is updated to the value in Q1e.

2. **Paxos**

Consider a system of five processes that implement the Paxos algorithm for consensus. As shown in Figure 1, P1 and P2 are proposers. P3, P4, P5 are acceptors. P1 sends two different *prepare* message to processes P4 and P5. The first one with proposal #4, and an initial intention to propose value 10 if it receives sufficient replies for this proposal. The next one with proposal #6, and an initial intention to propose a value 20 if it receives sufficient replies for this proposal. P2 sends a *prepare* message to P3 and P5 with proposal #5 and an intention to propose a value 15 if it receives sufficient replies for this proposal. Assume no other proposals are initiated.

No responses from processes P3, P4, and P5 are shown in the figure. Assume that, when applicable, any reply from any acceptor (for propose or accept requests from either of the proposers) takes exactly one time unit to reach the proposer. Further assume that, when applicable, the propose sends an *accept* message immediately upon receiving replies from majority of acceptors, and it takes exactly one time unit for the message to reach the corresponding acceptors.

Answer the following sub-questions.

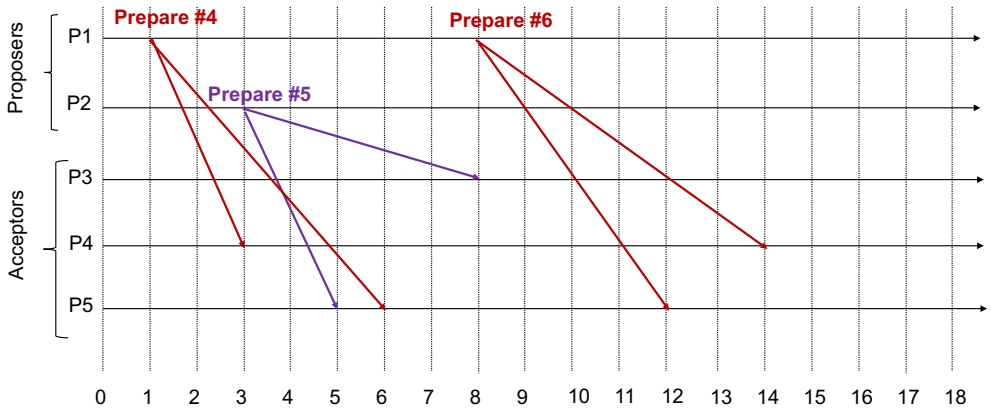

Figure 1: Figure for question 2

(a) (2 points) Which processes will reply back to P1's *prepare* message for proposal #4? Also specify, what value (if any) will the reply contain.

(b) (3 points) Will P1 send out an *accept* message for its proposal #4?

If not, explain why?

If yes, specify the following: (i) At what real time unit will it send that message? (ii) What will be the corresponding proposed value? (iii) Which processes will accept the value?

(c) (2 points) Which processes will reply back to P2's *prepare* message for proposal #5? Also specify, what value (if any) will the reply contain.

(d) (3 points) Will P2 send out an *accept* message for its proposal #5?

If not, explain why?

If yes, specify the following: (i) At what real time unit will it send that message? (ii) What will be the corresponding proposed value? (iii) Which processes will accept the value?

(e) (2 points) Which processes will reply back to P1's *prepare* message for proposal #6? Also specify, what value (if any) will the reply contain.

(f) (3 points) Will P1 send out an *accept* message for its proposal #6?

If not, explain why?

If yes, specify the following: (i) At what real time unit will it send that message? (ii) What will be the corresponding proposed value? (iii) Which processes will accept the value?

(g) (1 point) Do the above sequence of messages lead to an implicitly decided value? If yes, what is the earliest time 't' in the above figure, where decision is (implicitly) reached? In other words, what is the earliest time 't', such that even if the sequence of events differ after time 't' from what is shown in the figure, the decided value remains unchanged? What is this implicitly decided value?

3. **RAFT Leader Election**

Consider a system of 5 processes $\{P1, P2, P3, P4, P5\}$ using Raft's algorithm for leader election. Suppose P1, the leader for term 1, fails and its four followers receive its last heartbeat at exactly the same time. Assume that the election timeout is chosen uniformly at random from the range [100,500] ms (unless otherwise specified), no processing delay exists, and the one-way delay for all messages between two processes are as shown in Figure 2. The processes communicate with one-another only through their direct channels (not via other processes).

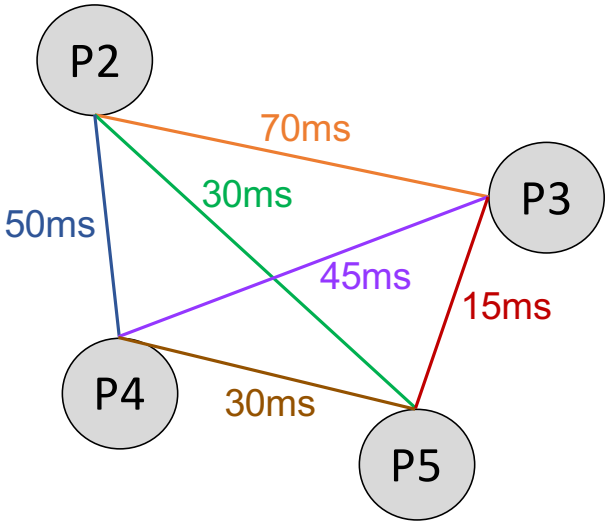

Figure 2: Figure for question 3

Suppose P2 sets its election timeout to 200 ms and calls for an election for term 2. Assume P4 and P5 have their timeout values set to more than 400 ms. What range of timeout values for P3 (within [100,500] ms) will certainly result in:

(a) (1 point) P2 winning the election for term 2?

(b) (1 point) P3 winning the election for term 2?

(c) (1 point) split vote for term 2?

4. **RAFT Log Consensus**

Consider a system of three servers $\{S_1, S_2, S_3\}$ wanting to achieve log consensus using the Raft algorithm. For each sub-part below, state whether the shown snapshot of log entries at each server could arise from a valid run of the Raft algorithm. If yes, construct a scenario that would lead to these log entries in Raft's execution. If not, explain what makes the entries invalid.

Each number in the shown log entries represents the Raft term that the corresponding event is associated with.

For the valid log entries, the scenario you construct should include, for each term: which server gets elected as the leader, which servers vote for it, and which log entries does it append / replicate at each server.

(a) (3 points)

S_1: 1, 1, 3
S_2: 1, 1,
S_3: 1, 1, 2

(b) (3 points)

S_1: 1, 1, 1, 1
S_2: 1, 1, 2,
S_3: 1, 2, 2, 2

(c) (3 points)

S_1: 1, 1, 1
S_2: 1, 1, 2, 2
S_3: 1, 1, 3

(d) (3 points)

S_1: 1, 1, 1
S_2: 1, 1, 1, 3
S_3: 1, 1, 2