Homework 4

CS425/ECE428 Spring 2021

Due: Thursday, April 1 at 11:59 p.m.

1. 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 $x_1 = 12$, $x_2 = 3$, $x_3 = 5$, $x_4 = 8$, and $x_5 = 10$.

   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 $45ms$. Answer the following questions with respect to local time at the processes' clock since the start of the system.

   (a) (2 points) 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) (2 points) Assume P_2 fails right after unicasting x_2 to P_4 and P_5 but just before it could initiate the unicast of x_2 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_2 fails at right after unicasting x_2 to P_4 but just before it could initiate the unicast of x_2 to any of the other processes. P_4 fails right after it has relayed x_2 to P_1 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_2 fails right after unicasting x_2 to P_4 but just before it could initiate the unicast of x_2 to any of the other processes. P_4 fails right after it has relayed x_2 to P_1 but just before it unicasts it to any other process. P_3 fails right before it could unicast x_3 to any process. When will each of the remaining alive processes decide on a value and what will each of their decided values be?

   (e) (2 points) If it is known that no more than 3 process may fail in the system, what is the smallest value that the *timeout* should be set to for ensuring safety? Now answer Q1d assuming that the timeout is updated to this value.

2. Consider a system of five processes that implement the Paxos algorithm for consensus. P1 and P2 are proposers. P3, P4, P5 are acceptors. Answer the following sub-questions (parts (a) and (b) are independent of one another).

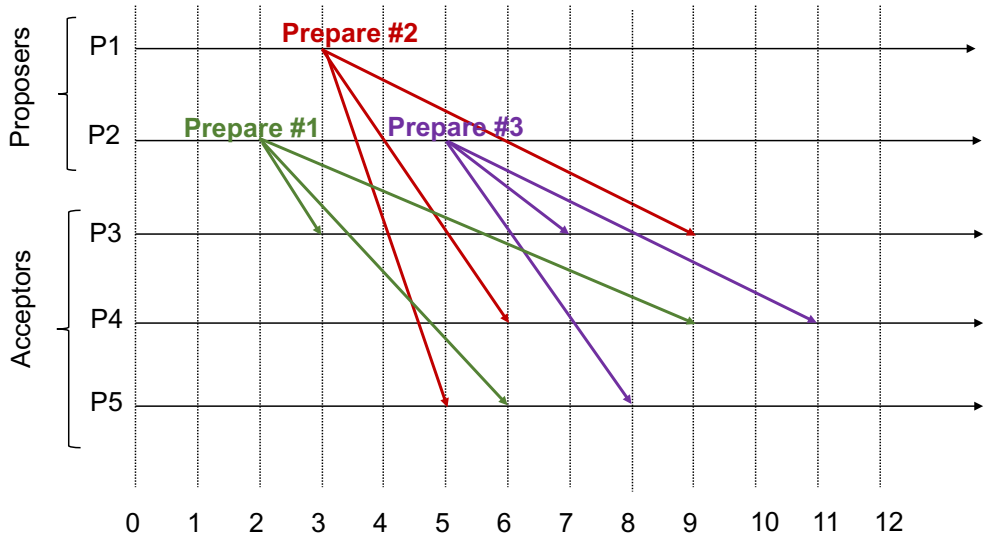

Figure 1: Figure for question 2(a)

(a) (3 points) Refer to Figure 1. P1 send a *prepare* message with proposal number 2, and P2 sends two different *prepare* messages with proposal numbers 1 and 3 respectively to all three acceptors. The responses from P3, P4, and P5 (if any) are not shown in the figure. Assume no other proposals are initiated.

(i) Which processes will reply back to P2's *prepare* message for proposal#1? *(1 points)*

(ii) Which processes will reply back to P1's *prepare* message for proposal#2? *(1 points)*

(iii) Which processes will reply back to P2's *prepare* message for proposal#3? *(1 points)*

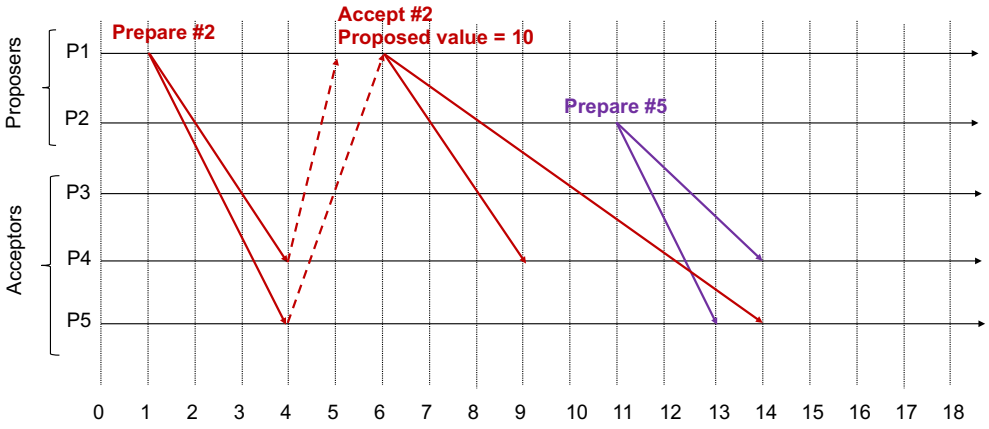

Figure 2: Figure for question 2(b)

(b) (7 points) Now refer to Figure 2. P1 sends a *prepare* message with proposal number 2 to processes P4, and P5, receives their replies, and sends an *accept* message with proposed value of 10 for proposal #2. P2 concurrently sends a *prepare* message with proposal #5, with an initial intention

to propose a value of 15 if it receives sufficient replies. Only a subset of responses from processes P4, and P5 are shown in the figure. Assume no other proposals are initiated.

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

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

(iii) Will P2 send out an *accept* message for its proposal #5? If yes, what will be the corresponding proposed value? *(1 point)*

(iv) Consider the state of the system at time 10 units. Has the proposed value 10 been implicitly decided upon at this point? If yes, explain why? If not, construct a possible scenario that may occur after time 10 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=10units but must comply with what is shown until t=10units. 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. *(2 points)*

(v) Suppose that P1's *accept* request reaches P5 at time 8 units (instead of 14 units). If we now consider the state of the system at time 10 units, has the proposed value 10 been implicitly decided upon? *(1 point)*

(vi) Suppose that P1's *accept* request reaches P4 at time 15 units (instead of 9 unit) and reaches P5 at the original time 14 units. Will P2 send out an *accept* message for its proposal #5? If yes, what will be the corresponding proposed value? *(1 point)*

3. 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. Answer the following questions assuming that the election timeout is chosen uniformly at random from the range [100,400] ms (unless otherwise specified), no processing delay exists, and the one-way delay for all messages between two processes are as shown in Figure 3. Each question below is independent of others.

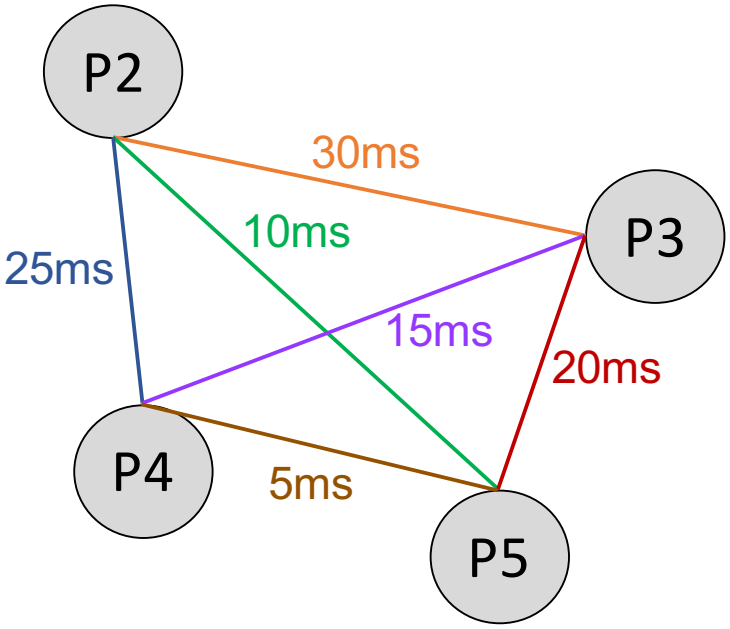

Figure 3: Figure for question 3

(a) (2 points) Suppose P2 and P4 both set their election timeout to 100 ms and call an election for term 2. Assume P3 and P5 have their timeout values set to more than 300 ms. Which candidate

(P2 or P4) will each of the four alive processes vote for? Will a leader be elected for term 2? If yes, which process?

(b) (2 points) Suppose P2 sets its election timeout to 100 ms and calls an election for term 2, and P4 sets its timeout to 120 ms and also calls an election for term 2. Assume P3 and P5 have their timeout values set to more than 300 ms. Which candidate (P2 or P4) will each of the four alive processes vote for? Will a leader be elected for term 2? If yes, which process?

(c) (2 points) Suppose P2 sets its election timeout to 100 ms and calls for an election for term 2. Assume P3 and P5 have their timeout values set to more than 300 ms. What range of timeout values for P4 (within [100,400] ms) will result in a split vote?

(d) (4 points) Suppose P2 sets its election timeout to 100 ms and calls for an election for term 2. What is the probability that another process (among P3, P4 and P5) also calls an election for term 2? [*Hint:* this probability can be computed as (1 - *(probability that neither P3 nor P4 nor P5 call for an election for term 2)*)]

4. Consider the following logs of servers in a Raft cluster:

S_1 1, 1, 1, 2, 2, $\underline{4}$, 4, 6, 6

S_2 1, 1, 1, 2, 2, 3, 3, 3, 5, 5, 5

S_3 1, 1, 1, 2, 2, 2, 2

S_4 1, 1, 1, 2, 2, $\underline{4}$, 4, 6

S_5 1, 1, 1, 2, 2, $\underline{4}$, 4, 4, 4

Each number represents the term that the event is associated with. Answer the following questions based on these log entries.

(a) (3 points) Order the servers from least up-to-date to most up-to-date.

(b) (3 points) List which processes could have been possible leaders for terms 2, 3, 4, 5, and 6. Mentioning just one server for each term is sufficient.

(c) (4 points) Based on the logs, is it *certain* that the underlined entry for term 4 in the logs of S_1, S_4, and S_5 could have been safely committed at some point? If yes, construct the scenario complying with the shown logs, that would have led to the underlined entry being committed. If not, construct a possible scenario complying with shown logs, that would have led to the underlined entry not being committed, and could eventually lead to it being overwritten.

[A scenario will consist of temporal sequence of steps. Each step may correspond to a process winning an election for a term (including which processes vote for it), or to one or more of the leader's entries getting successfully replicated (including which processes replicate the entries), or possible (transient) communication failures between processes resulting in one or more entries not being replicated or a new election being initiated. You may start your sequence of steps from the election of term 3's leader.]