Homework 3

CS425/ECE428 Spring 2020

Due: Thursday, March 12 at 11:59 p.m.

1. Consider the following modification of the Bully algorithm: The initiating node (which we assume does not fail) sends an Election message only to the process with the highest id. If it does not get a response after a timeout, it then sends an Election message to the process with the second highest id. If after another timeout it gets no response, it tries the third highest id, and so on. If no higher numbered processes respond, it sends a Coordinator message to all lower-numbered processes.

   (a) (2 points) What should a process do when it receives an Election message in order to minimize turnaround time?

   For the following parts, consider a distributed system of 6 processes that uses the modified Bully algorithm for leader election. Initially all 6 processes are alive and P_6 is the leader. Then P_6 fails, P_3 detects this, and initiates the election. Assume one-way message transmission time is T, and timeout is set using the knowledge of T.

   (b) (2 points) If no other node fails during the election run, how many *total* messages will be sent by *all* processes in this election run?

   (c) (2 points) If no other node fails during the election run, how long will it take for the election to finish?

   (d) (2 points) Now assume that right after P_3 detects P_6's failure and initiates the election, P_5 fails. How many *total* messages will be sent by *all* processes in this election run?

   (e) (2 points) For the above scenario (where P_5 fails right after P_3 initiates election upon detecting P_6's failure), how long will it take for the election to finish?

2. Consider a system of N process that are arranged in a ring, with each process having a ring successor and a predecessor, and a communication channel only to its ring successor. Each process P_i has a unique id i. A process P_i maintains a value x_i (these values may not be unique across processes).

   (a) (6 points) Consider a problem where each process P_k is required to set the value of an output variable y_k (initialized to *undecided*) to $\min_{i=1}^{N}(x_i)$. The safety condition for the problem requires that, at any point in time, the variable y_k at process P_k $\forall k \in [1, N]$ is either *undecided* or $\min_{i=1}^{N}(x_i)$.

   A distributed algorithm designed for the above problem works as follows:

   - A process P_i initiates the algorithm by sending $(propose, x_i)$ to its ring successor.
   - When a process P_j receives $(propose, x)$ from its ring predecessor:
     - if $x < x_j$, it forwards $(propose, x)$ to its successor.
     - if $x > x_j$, it sends $(propose, x_j)$ to its successor.
     - if $x = x_j$, it concludes that $x = x_j$ is the minimum value, and sends $(decided, x)$ to its successor.
   - When a process P_j receives $(decided, x)$, it sets $y_j = x$ and forwards $(decided, x)$ to its successor (if it had not already done so in the past). Once P_j sets y_j, it ignores any subsequent *decided* messages.

   Multiple processes may initiate the above algorithm simultaneously. Assume no process fails and the communication channel delivers all messages correctly and exactly once.

   Does the algorithm described above guarantee safety condition for the problem? If yes, prove how. If not, (i) describe a scenario where safety is violated, and (ii) suggest modifications to the algorithm that would guarantee the safety condition.

   (b) (4 points) Now consider a modified problem, where each process P_k must decide on a value y_k which is $\Sigma_{i=1}^{N} x_i$. Design a ring-based algorithm for this problem, which follows the constraint that processes are arranged in a ring, with each process having a ring successor and a predecessor, and a communication channel only to its ring successor. You may assume that no process fails and all messages are delivered correctly and exactly once. Multiple processes may initiate your algorithm simultaneously.

3. 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 $x_1 = 2$, $x_2 = 10$, $x_3 = 4$, $x_4 = 7$, and $x_5 = 3$.

   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 $i \in [1, 5]$. 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 (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 $(t + \textit{timeout})$, 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 when P_i begins the multicast of x_i. A message sent from a process to any other process takes exactly T time units (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 $2T + \epsilon$, where $0 < \epsilon < T$ is a small positive value.

   (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_1 fails right after unicasting x_1 to P_3, but just before it could initiate the unicast of x_1 to any of the other processes. When will each of the remaining alive processes decide on a value and what will each of their decided values be?

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

   (d) (2 points) Assume P_1 fails right after unicasting x_1 to P_3 but just before it could initiate the unicast of x_1 to any of the other processes, and P_5 fails right before it could unicast x_5 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?

4. Consider a system of five processes that implement the Paxos algorithm for consensus. Answer the following sub-questions, each of which is unrelated to the other two sub-questions.

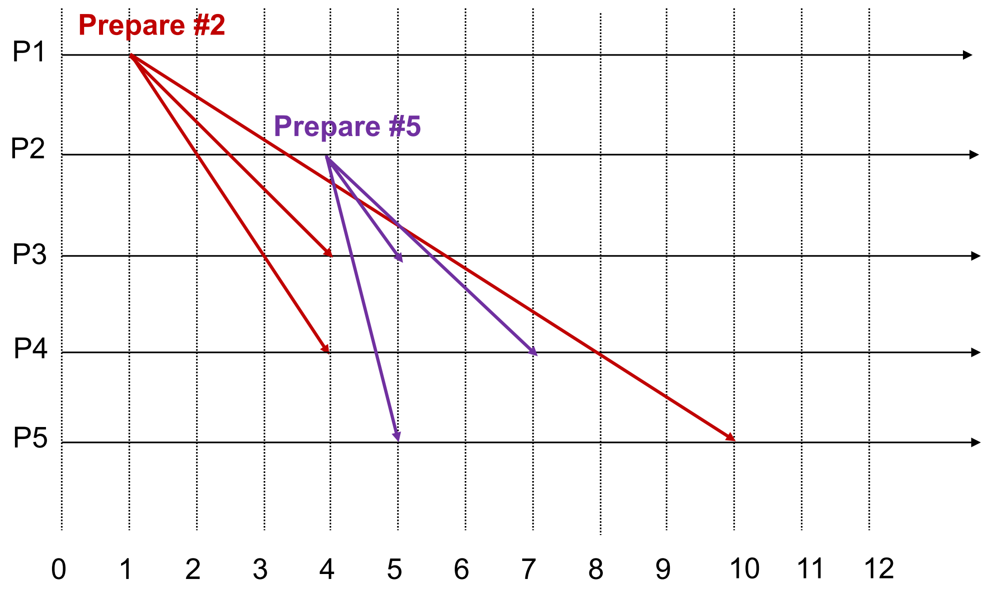

Figure 1: Figure for question 4(a)

(a) (3 points) Refer to Figure 1. P1 and P2 send a *prepare* message with proposal numbers 2 and 5 respectively to processes P3, P4, and P5. The responses from processes 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 P1's *prepare* message? *(1.5 points)*

(ii) Which processes will reply back to P2's *prepare* message? *(1.5 points)*

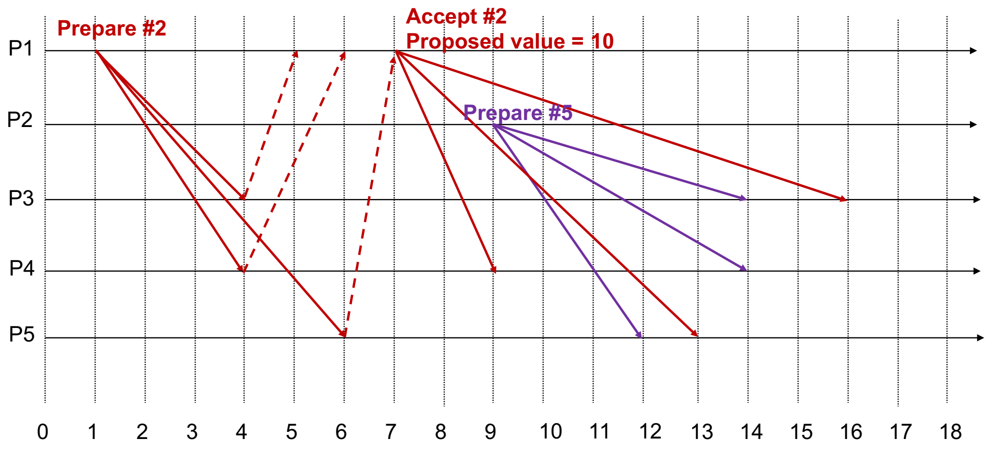

Figure 2: Figure for question 4(b)

(b) (4 points) Now refer to Figure 2. P1 sends a *prepare* message with proposal number 2 to processes P3, P4, and P5, receives their replies, and sends an *accept* message with proposed value of 10 (and 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 P3, P4, and P5 are shown in the figure. Assume no other proposals are initiated.

(i) Which processes will *accept* P1's proposal? *(1.5 points)*

(ii) Which processes will reply back to P2's *prepare* message? *(1.5 points)*

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

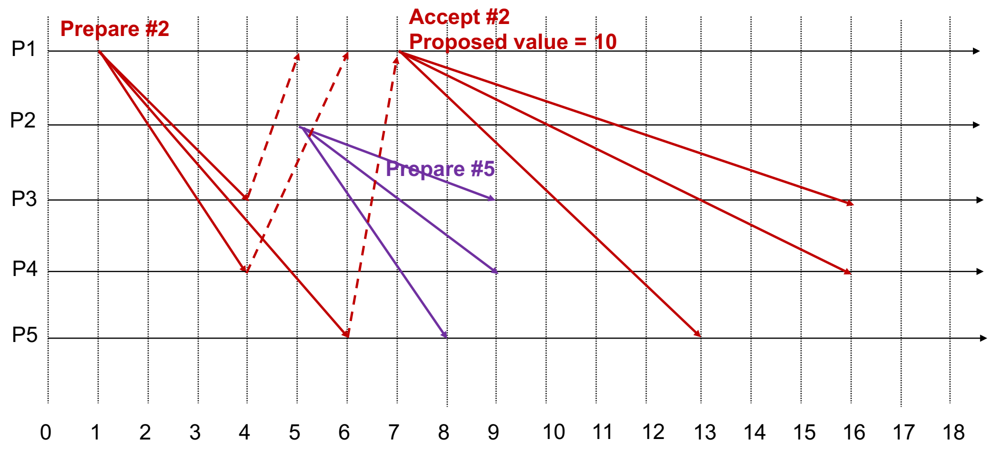

Figure 3: Figure for question 4(c)

(c) (3 points) Now refer to Figure 3. P1 sends a *prepare* message with proposal number 2 to processes P3, P4, and P5, receives their replies and sends an *accept* message with proposed value of 10 (and 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 P3, 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 message? *(1 point)*

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