Homework 3

CS425/ECE428 Spring 2023

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

1. **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. Answer the following questions assuming 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 1. The processes communicate with one-another only through their direct channels (not via other processes).

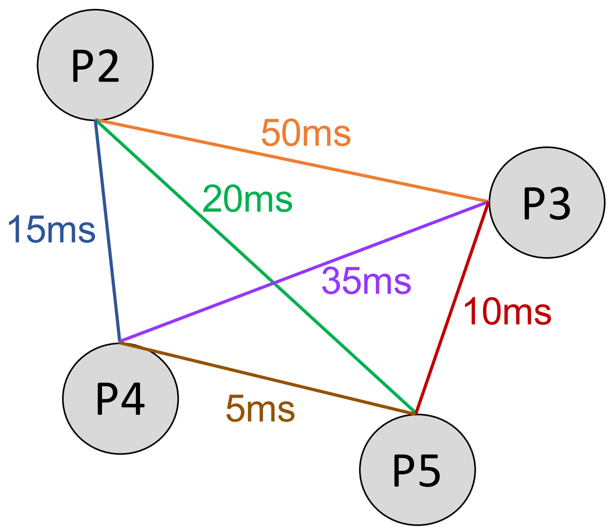


Figure 1: Figure for question 2

Suppose P2 sets its election timeout to 170 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) (2 points) P2 winning the election?

> **Solution:** In order for P2 to win the leader election, there are two cases:
>
> - P2 receives P3's vote: P3's timeout need to be at least 170+50=220 ms
> - P2 receives P4 and P5's vote: we need to ensure that even if P3 starts election, its message should get to P4 and P5 after P2's message. Since P4 will receive P2's message at 185 ms and P5 will receive P2's message at 190 ms, P3's timeout need to be at least max(185-35, 190-10)=180 ms.
>
> By taking the union of two cases, P3's timeout should be within **(180, 500]** ms.

(b) (2 points) P3 winning the election?

> **Solution:** In order for P3 to win the leader election, there are two cases:
>
> - P3 receives P2's vote: P3's timeout need to be at most 170-50=120 ms

- P3 receives P4 and P5's vote: we need to ensure that even if P2 starts election, its message should get to P4 and P5 before P2's message. Since P4 will receive P2's message at 185 ms and P5 will receive P2's message at 190 ms, P3's timeout need to be at most min(185-35, 190-10)=150 ms.

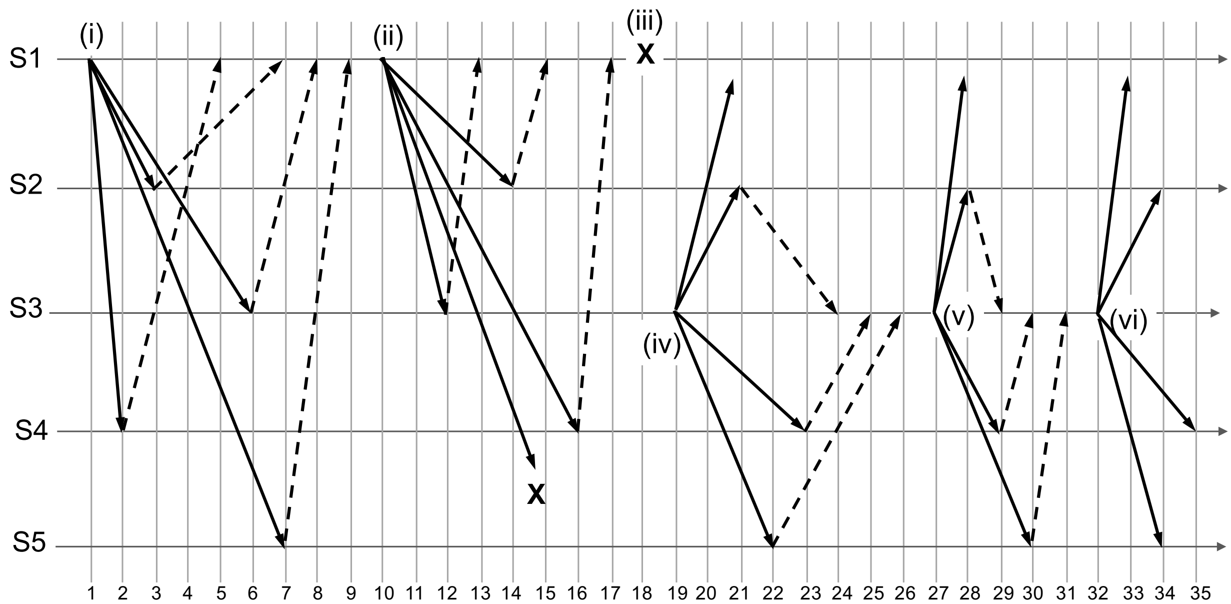
By taking the union of two cases, P3's timeout should be within **[100, 150)** ms.

(c) (2 points) split vote?

Solution: Split vote will happen if P3's timeout is within **(150, 180)** ms.

2. **RAFT timeline**

Consider the timeline below.



It demonstrates a series of events / message exchanges in a Raft cluster. The numbers at the bottom show the real time. Initially all servers start in follower state in term 0. The following key events occur as indicated in the figure:

(i) Server 1's election timeout expires, it sends RequestVote messages for term 1.

(ii) Server 1 receives log entry P from a client and sends AppendEntries messages, including P. Its message to Server 5 is dropped (i.e., never received).

(iii) Server 1 crashes; it remains crashed for the remainder of the execution.

(iv) Server 3's election timeout expires, it sends RequestVote messages for term 2.

(v) Server 3 receives log entry Q from a client and sends AppendEntries messages to nodes, including Q.

(vi) Server 3 sends the next AppendEntries messages to nodes.

List the time that each of the following happen, using the timeline on the diagram (i.e., 1–35), or write "never" if the event never occurs.

(a) (1 point) Server 1 transitions to Leader state.

> **Solution:** At time 7, when server 1 receives votes from the majority

(b) (1 point) Server 4 updates its current term to 1.

> **Solution:** At time 2, when server 4 receives RequestVote from server 1

(c) (1 point) Server 1 *commits* event P in its log.

> **Solution:** At time 15, when server 1 knows the majority of servers have stored event P

(d) (1 point) Server 2 *appends* event P to its log.

Solution: At time 14, when server 2 receives AppendEntries from server 1

(e) (1 point) Server 4 updates its current term to 2.

Solution: At time 23, when server 4 receives RequestVote from server 3

(f) (1 point) Server 3 transitions to Leader state.

Solution: At time 25, when server 3 receives votes from the majority

(g) (1 point) Server 4 *commits* event Q in its log.

Solution: At time 35, since server 3 commits Q at time 30, and server 4 only commits Q after receiving the next AppendEntries

(h) (1 point) Server 4 *commits* event P in its log.

Solution: At time 35

(i) (1 point) Server 5 updates its current term to 2.

Solution: At time 22, when server 5 receives RequestVote from server 3

For the next parts, state the *earliest* time after which the following *must* be true, even if execution after this time proceeded differently than what is shown in the diagram. Assume that once server 1 has crashed, it can never recover.

(j) (1 point) Server 5 *cannot* be elected as leader in term 1.

Solution: At time 3, when the majority of servers have voted for server 1 in term 1

(k) (1 point) Server 2 *cannot* be elected as leader in term 2.

Solution: At time 21, when server 2 has already granted vote for server 3

(l) (1 point) Event P will eventually be committed by at least one server.

Solution: At time 14, when P is stored on a majority of servers Another case (if you assume the majority nodes will fail and never recover): at time 15, when S1 commits event P

(m) (1 point) Event Q will eventually be committed by at least one server.

Solution: At time 29, when Q is stored on a majority of servers Another case (if you assume the majority nodes will fail and never recover): at time 30, when S3 commits event Q Also accepted: at time 28, since S1 remains crashed and after time 28 any 3-server majority will contains the event Q.

3. **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, 1

S_2: 1, 1, 2, 2, 2

S_3: 1, 1

> **Solution:** Valid.
>
> S_1 is the leader for term 1, and it sends the first 2 logs to all servers. After that, S_1 becomes disconnected to the network and it adds the third log to itself. S_2 detects S_1 as failed and starts leader election, S_3 grants vote for S_2, and S_2 becomes the leader. It then adds another 3 logs to itself.

(b) (3 points)

S_1: 1, 1, 1

S_2: 1, 1, 2, 2, 2

S_3: 1, 1, 3

> **Solution:** Invalid.
>
> S_1 must be the leader for term 1 since it has the most number of logs for term 1. Based on log entries, S_2 must be the leader for term 2 and S_3 must be the leader for term 3. But in order for S_3 to be the leader for term 3, it must get vote from at least another server. S_2 cannot grant vote since it's the leader for term 2, and S_1 also cannot grant vote since it has more complete logs than S_3 before S_3 becomes the leader. Thus, this case is invalid.

(c) (3 points)

S_1: 1, 1, 1

S_2: 1, 2, 2, 2

S_3: 1, 1, 1, 3

> **Solution:** Valid.
>
> S_3 is the leader for term 1, and it sends the first log to all servers. It then gets disconnected to the network and adds 2 logs to itself. S_2 detects this failure and it becomes the leader for term 2 by getting the vote from S_1. S_2 then gets disconnected to the network and adds 3 logs to itself. S_3 then gets reconnected to the network and it's elected as the leader for term 3 by getting vote from S_1. S_1 then repairs its logs so it's consistent with S_3's logs. S_3 then gets a log for term 3 and adds to itself.

(d) (3 points)

S_1: 1, 1, 1, 3

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

Solution: Valid.
S_1 is the leader for term 1, and it sends the first log to all servers. It then gets disconnected to the network and adds 2 logs to itself. S_2 detects this failure and it becomes the leader by getting the vote from S_3. S_2 then gets disconnected to the network and adds 3 logs to itself. S_1 then gets reconnected to the network and it's elected as the leader for term 3 by getting vote from S_3. S_1 then gets a log for term 3 and sends it to S_3. S_3 then repairs its logs so it's consistent with S_1's logs.

(e) (3 points)
S_1: 1, 1, 1, 1
S_2: 1, 1, 2, 2
S_3: 1, 1, 2, 3

Solution: Valid.
S_1 is the leader for term 1. It sends out the first 2 logs to everyone then gets disconnected from the network, and it adds 2 logs to itself. After S_1 gets disconnected, S_2 detects the failure and becomes the leader by getting the vote from S_3. S_2 then sends out 1 log to S_3. S_2 then gets disconnected to the network and adds another log to itself. S_3 detects S_2's failure and starts leader election. At this time, S_1 gets reconnected to the network and grants vote for S_3. S_3 then becomes the leader for term 3 and appends a log to itself.

4. **Blockchains**

In a system using a blockchain for distributed consensus, in order to add a block to a chain, a participating node must solve the following puzzle: it must find a value x such that its hash, $H(x||seed)$, is less than T. The hash function is such that a given value of x can uniformly map to any integer in $[0, 2^{256} - 1]$. Assume T is set to 2^{220}.

(a) (2 points) What is the probability that a given value of x, randomly chosen by the participating node, is a winning solution to the puzzle (i.e. $H(x||seed) < T$)?

Solution:

$$P(H(x||seed) < T) = \frac{2^{220}}{2^{256}} = \frac{1}{2^{36}}$$

(b) (2 points) Assume a participating node adopts the standard strategy for solving the puzzle: it randomly picks a value x and checks if it is the winning solution. It keeps repeating this step, until a winning solution is found. Further assume that, for simplicity, the strategy is memoryless (unoptimized), in the sense that a value of x that has already been checked can get re-checked if it is randomly selected again. If the node can hash and check 2^{10} values per second, what is the probability of finding a winning solution within 5 hours? (You may round your answer to five decimal places.)

Solution:

$$\begin{aligned} P(\text{winning within 5hrs}) &= 1 - P(\text{winning after 5hrs}) \\ &= 1 - (1 - \frac{1}{2^{36}})^{3600 \cdot 5 \cdot 2^{10}} \\ &= 0.00027 \end{aligned}$$

(c) (2 points) Assume there are 10000 participating nodes in the system, and that each node starts solving the puzzle at exactly the same time. Assuming the same rate of computing hashes at each node (i.e. 2^{10} values per second), what is the probability that at least one node in the system finds a winning solution in 5 hours? (You may round your answer to four decimal places.)

Solution:

$$
\begin{aligned}
P(\text{at least one winning within 5hrs}) &= 1 - P(\text{no one winning within 5hrs}) \\
&= 1 - (1 - \frac{1}{2^{36}})^{10000 \cdot 3600 \cdot 5 \cdot 2^{10}} \\
&= 0.9316
\end{aligned}
$$