# Quiz 4

● Graded

**Student**

Boning Li

**Total Points**

26 / 30 pts

**Question 1**

## Question 1

9 / 10 pts

- **– 0 pts** Correct (NO, YES) with good justification + execution
- ✓ **– 1 pt** Correct (NO, YES), justification casework incomplete (consider both/all proposal orders)
- **– 2.5 pts** Correct (NO, YES), justification poor or irrelevant
- **– 2.5 pts** pt2 incorrect (should be NO, YES)—$A_1$ may fail to receive messages or crash
- **– 4 pts** pt1 incorrect (should be NO, YES)
- **– 6.5 pts** Incorrect (should be NO, YES)—for pt2, notice that $A_1$ may fail to receive messages or crash
- **– 8 pts** Incorrect with poor or irrelevant justification
- **– 8 pts** pt1 incorrect, and does not adequately describe executions
- **– 9 pts** Incorrect and pt1 does not describe an execution
- **– 10 pts** Unclear which question is being answered, justification doesn't seem relevant to either

Question 2

## Question 2

**7** / 10 pts

**– 0 pts** Correct

**– 1 pt** Correct response (NO), minor flaws in justification

**– 2 pts** Correct response (NO), justification confused about T/T'

**– 4 pts** Correct response (NO), justification irrelevant or answers a wrong question

✓ **– 3 pts** Convoluted failure scenario; this case should actually be handled by reattempting the election if you receive ok but no coordinator within T', so it does take longer but you shouldn't adjust T' for it

**– 5 pts** Incorrect (should be NO), note that multiple elections *don't* need to run in sequence, and the initiator always gets an ok from the process that will ultimately identify itself as coordinator

**– 5 pts** Incorrect (should be NO), the <3p cases are actually special, it turns out that T' does not actually need to be increased once it's nonzero

**– 7 pts** Incorrect (should be NO), total number of messages is not relevant to T'

**– 7 pts** Clearly misinterpreted question—yes, you can choose a gratuitous T' for the 6p system and a tighter T' from the 5p system, but that's silly and meaningless

**– 9 pts** Incorrect—bully algorithm does not use ring topology

**– 10 pts** Incorrect or missing

Question 3

## Question 3

**10** / 10 pts

✓ **– 0 pts** Correct

**– 7 pts** Mostly incorrect but has some relevant information

**– 10 pts** Missing or incorrect

**– 2 pts** Some mistakes in answer

**– 2 pts** Answer recalls some aspects of proof, but is missing the big picture

**– 1 pt** Minor mistakes in answer

# Distributed Systems and Algorithms — CSCI 4510/6510
Quiz 4
November 11, 2024

RCS ID: libl9 @rpi.edu Name: Bohing Li

**Instructions:**

- You will have **45 minutes** to complete this quiz. Please do not start until told.
- Write your RCS ID and name in the blanks at the top of this cover sheet.
- Put away notes, laptops, and other electronic devices. Cheating on a quiz will result in an **immediate F in the course** and a report will be filed with the Dean of Students.
- Read each question carefully several times before beginning to work and especially before asking questions.
- Write your answers clearly and completely inside the box.

**Question 1 (10 points).** Consider the Synod algorithm with 3 acceptors: $A_1$, $A_2$, and $A_3$. For each example below, indicate whether it represents a possible state of the acceptors' (*accNum*, *accVal*) variables at a single wall clock time $t$. If so, describe an exchange of messages between proposers and acceptors that results in that state. If not, explain why not.

1. $A_1 = (4, \text{'apple'})$, $A_2 = (5, \text{'pecan'})$, $A_3 = (4, \text{'apple'})$
2. $A_1 = (\perp, \perp)$, $A_2 = (3, \text{'pumpkin'})$, $A_3 = (2, \text{'blueberry'})$

1. ~~Yes. Assume $P_1$ initiates the~~ ~~proposal~~ ~~prepare (4), which is received and then by $A_1$, $A_2$ and $A_3$. Upon receiving these promises $(\perp, \perp)$, $P_1$ sends accept(4, 'apple') to $A_1$, $A_2$, and $A_3$. Now, all of them have (accNum, accVal) = (4, 'apple')~~ NO As $A_1$ and $A_3$ have already agree on the value 'apple', there's no way for another proposer to propose a distinct value 'pecan' because it won't be accepted by the majority $(\frac{2}{3})$.

2. Yes Assume initially, $A_2$ is down. $A_1$ and $A_3$ have $(\perp, \perp)$. Now, $P_1$ sends prepare(2) to $A_1$, $A_2$, and $A_3$. $A_1$ and $A_3$ ~~res~~ responds with promise$(\perp, \perp)$. Aftere receiving promises, $P_1$ sends accept (2, 'blueberry') to $A_1$, ~~and~~ $A_2$ & $A_3$. $A_1$ is down and $A_3$ receives the msg, ~~a~~and then updates (accNum, accVal) = (2, 'bluebery'). Now, $A_1$ and $A_2$ recover while $A_3$ is down. Let $P_2$ sends prepare (3) ~~'pumpkin'~~ to all them, and receives promise $(\perp, \perp)$ from $A_1$ and $A_2$. $P_2$ sends accept (3, 'pumpkin') to all acceptors. Assume $A_1$ is down again, and only $A_2$ receives accept. So, only ~~P~~ $A_2$ updates (accNum, accVal) = (3, 'pumpkin'). Finally, after $A_1$ & $A_3$ recovers, their records are $A_1 = (\perp, \perp)$, $A_2 = (3, \text{'pumpkin'})$,

$A_3 = (2, \text{'bluebery'})$

**Question 2 (10 points).** Recall that in the Bully Algorithm, after a process receives an 'OK' message, it waits for $T'$ time to receive a 'coordinator' message. If it does not receive a 'coordinator' message within this time, the process starts a new election. Note that the value of $T'$ is the same for every process in the system.

Can a system with five processes use a smaller value for $T'$ than a system with six processes? Answer YES or NO and justify your answer.

Yes. We know the system is synchronus, so there's a upper bound of communication.

Imagine one case that the largest process fails after sending OK. For example, we have a system with 6 processes, from $P_1$ to $P_6$. $P_1$ initializes the electon and receives ok from others. Then, when $P_5$ sends election(5), $P_6$ fails. Then $P_5$ waits for $T'$ time and then fails. $P_4$ will repeat this process until all of them fails. Finally, $P_1$ will be the leader. Follow this approach, it's possible to set up a smaller $T'$ for a system with fewer processes.

**Question 3 (10 points).** To prove the FLP theorem, we first proved Lemma 2 and Lemma 3:

**Lemma 2**: Assume $\mathcal{P}$ is totally correct in spite of one fault. Then $\mathcal{P}$ has a bivalent initial configuration.

**Lemma 3**: Assume $\mathcal{P}$ is totally correct in spite of one fault. Let $C$ be a bivalent configuration, and let $e = (p, m)$ be an event that can be applied to $C$. Let $\hat{\mathrm{C}}$ be the set of configurations reachable from $C$ without applying $e$, and let $\hat{\mathrm{D}} = \{e(\mathcal{E}) \mid \mathcal{E} \in \hat{\mathrm{C}}$ and $e$ can be applied to $\mathcal{E}\}$. Then $\hat{\mathrm{D}}$ contains a bivalent configuration.

We then used these two lemmas to prove the theorem: no consensus protocol is totally correct in spite of one fault. Briefly explain (in "layman's terms"), how these two lemmas were used to prove the theorem. Be sure to discuss what role process failures (or the lack thereof) play in the proof of the theorem.

Given Lemma 2, we know that P has a bivalent initial config if one may fail. A bivalent configuration can lead to another bivalent ~~under~~ with one fault.

This means, if there's one fault, and we assume P is totally correct, then P's initial configuration must be bivalent. If the sequence of events is carefully chosen, the protocol can continuously go into a bivalent configuration. It will never terminate, which contradicts the fact that P is totally correct (agreement, validity & termination). Therefore, P cannot be totally correct.

Why process failures are important: The proof is based on bivalence, as discussed above. The process failure is the key point that introduces bivalence to the system. If every config is univalent, the failure will lead to a contradiction that neighbor configs may go from 0-valent to 1-valent, or ~~the~~ inversely. This is impossible. Therefore, a config with one fault must be bivalent.