Quiz 5

Graded

Student

Boning Li

Total Points

25 / 30 pts

Question 1

Question 1

8 / 8 pts

✔ **– 0 pts** Correct

– 1 pt Works but suboptimal in messages sent

– 3 pts Partial or underspecified

– 8 pts No response

– 0 pts Click here to replace this description.

Question 2

Question 2

4 / 8 pts

– 0 pts Correct

– 1 pt Correct response (NO) but unknown should not be a final value

✔ **– 4 pts** Incorrect (should be NO)—even though no process can have voted abort at this point, agreement can still be violated if commit is sent while uncertain processes exist, then all but uncertain processes crash and remaining uncertain processes decide abort

– 5 pts Correct response (NO) but weak justification

– 7 pts Doesn't seem to answer question

Question 3

Question 3

7 / 7 pts

✔ **– 0 pts** Correct

– 2 pts Fails to identify violation in execution

– 4 pts Violation not demonstarted

– 6 pts No execution given

– 7 pts No reponse

Question 4

Quesiton 4

6 / 7 pts

− 0 pts Correct

✔ **− 1 pt** Asynchronous reliable messaging does not require claimed total order violation to be possible

− 3 pts Integrity violation is avoidable

− 4 pts Execution not given

− 5 pts Processes misbehaving outside of the system model does not demonstrate that the algorithm misbehaves

− 70 pts No response

Distributed Systems and Algorithms — CSCI 4510/6510

Quiz 4

November 11, 2024

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

Instructions:

- You will have **50 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 (8 points). Consider the Byzantine Generals Problem in system where it is known which sites are traitors, i.e., the set of site IDs for the treacherous sites is provided as input to the algorithm. Design an algorithm that solves the Byzantine Generals problem when this information is available. Assume the system model is the same as for Lamport's algorithm. Your solution should use oral messages, and it should send as few messages as possible.

Since the traitors are known ~~for~~ by loyal generals, we can use one round to solve the problem.

① If the commander is traitor, ~~that means the initial command~~ all loyal generals will know this because the recipient knows who sent the message. In this case, all loyal generals agree on R.

② If the commander is loyal, all loyal generals simply follow this initial command, either A or R.

N = number of generals including commander

This algorithm only requires $\boxed{N-1 \text{ messages}}$ in one round.

It satisfies IC1 because all loyal generals follow the same command.

It satisfies IC2 because all loyal generals agree on R when commander is traitor, and it satisfies IC1.

Question 2 (8 points). Recall that the Three-Phase Commit (3PC) algorithm is non-blocking, meaning that if processes crash, the remaining processes can reach agreement and decide.

Dr. Science proposes the following modified version of (TR4) in the Termination Protocol in 3PC: if any active process is in the pre-commit state, the coordinator decides "commit" and sends "commit" to all processes. All other parts of the algorithm remain the same.

Does the 3PC algorithm with Dr. Science's modification satisfy agreement? If yes, argue why. If not, describe an execution of the modified algorithm that violates agreement. Assume the system model is the same as in the notes.

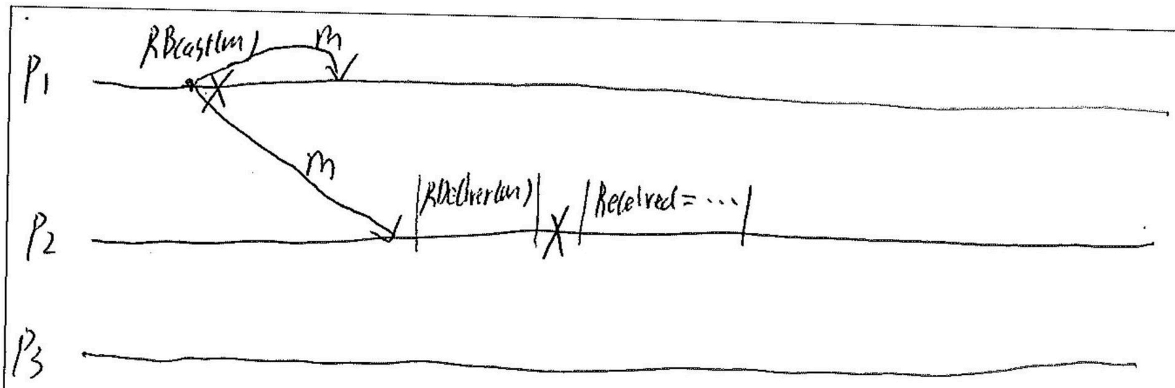
The only case that some process is in pre-commit state is that all voted "commit" in phase 1. ~~It's possible that the coordinator~~ When TR4 happens, that means some processes are in "pre-commit" state and some are in "uncertain" state. None of them would abort. So, when an active process sends "commit" to others, they will decide on "commit" as long as they don't crash. Therefore, no two processes decide on different values. The answer is $\boxed{\text{YES}}$.

After modifying TR4, we can still claim that it satisfies agreement. If the coordinator fails before sending all "commit"s, that means some processes are "commit" while others are "pre-commit". In this case TR4 will ensure all correct processes will ~~decide~~ decide on "commit".

Question 3 (7 points). Recall that the Diffusion Algorithm guarantees the three properties of Reliable Broadcast: validity, agreement, and integrity. In class, we showed that it also guarantees a stronger agreement property called *uniform agreement*: if any process delivers a message m, then all correct processes eventually deliver m.

We also studied a variation of the Diffusion Algorithm where the order of the deliver and relay steps are reversed:
on $recv(m)$:
. if $m \notin Received$
$RDeliver(m)$
$Received = Received \cup \{m\}$
if $senderID \neq self$
$send(m)$ to all processes (can exclude self)

Prove that this variation does not guarantee the uniform agreement property by giving an execution of the algorithm that violates this property.



There are three processes P_1, P_2 and P_3. Now, P_1 broadcasts m but crashes before sending m to P_3. P_2 receives m and invokes RDeliver(m). However, P_2 crashes before executing Received = ... ~~H~~ In this case, m will not be sent to P_3. Clearly, P_2 has delivered m and P_3 is correct. However, P_3 won't deliver m eventually. This violates the uniform agreement.

Question 4 (7 points). Consider the atomic broadcast algorithm we studied in class in an asynchronous system with reliable messaging and crash failures. Prove that this algorithm does not guarantee atomic broadcast in this system model by giving an execution of the algorithm that violates one of the four required properties. Be sure to indicate which property is violated and why.

Assume there are two processes P_1 and P_2, and a sequencer.
P_1 and P_2 want to broadcast m_1 and m_2, respectively. They send m_1 & m_2 to sequencer, and then the sequencer attaches a seqNum to each of them. Assume it assigns 1 to m_1 and 2 to m_2. It sends $(m_1, 1)$ to P_1 and P_2 firstly, and then sends $(m_2, 2)$ to P_1 and P_2. As the system is asynchronous, it's possible that P_1 receives $(m_1, 1)$ first while doesn't receive $(m_2, 2)$ for a long time. Similarly, P_2 receives $(m_2, 2)$ first and doesn't receive $(m_1, 1)$ for a loooooooooooong time. In this case, P_1 will deliver m_1 first but P_2 will deliver m_2 first.
This violates the property of "total order".

Byzantine Agreement with Oral Messages

An algorithm that solves the Byzantine Generals problem satisfies the following properties.

- (IC1) All loyal lieutenants obey the same order.
- (IC2) If the commander is loyal, then every loyal lieutenant obeys the order it sends.

System Model: synchronous; reliable communication, N process, M are faulty.

Properties of Oral Messages:

- Every message that is sent is received correctly.
- The recipient of a message knows who sent it.
- The absence of a message can be detected.

Theorem: There is no solution for the Byzantine Generals Problem (with oral messages) for $N < 3M + 1$.

Lamport's Algorithm for Byzantine Agreement with Oral Messages (with at most m traitors

function $majority(v_1, \ldots, v_{n-1})$:
  return majority value in $\{v_1, \ldots, v_{n-1}\}$ or RETREAT if no majority exists.

Case: $OM(0)$

1. The commander sends its value v to every lieutenant.
2. Each lieutenant uses the value received from the commander, or RETREAT if no value is received.

Case: $OM(m)$, $m > 0$

1. The commander sends its value to all $n - 1$ lieutenants.
2. For each i, let v_i be the value Lieutenant i receives from the commander, or RETREAT if no value is received. Lieutenant i acts as the commander in $OM(m-1)$ to send the v_i to each of the $n-2$ other lieutenants.
3. For each i, and each $j \neq i$, let v_j be the value Lieutenant i received from Lieutenant j in step (2) (using $OM(m-1)$), or RETREAT if no value is received. Lieutenant i uses the value $majority(v_1, v_2, \ldots, v_{n-1})$.