Quiz 3

● Graded

Student

Boning Li

Total Points

29 / 30 pts

Question 1

Question 1

8 / 8 pts

✔ **– 0 pts** Correct under implicit FIFO assumption (5 does not pass token to 1, then request it, with request arriving before token)

Question 2

Question 2

8 / 8 pts

✔ **– 0 pts** Correct

Question 3

Question 3

8 / 8 pts

YES

✔ **– 0 pts** Correct

Question 4

Question 4

5 / 6 pts

Part 1

✔ **– 1 pt** The algorithm satisfies the properties, but the justification has errors

---

Part 2

✔ **– 0 pts** Correct

Distributed Systems and Algorithms — CSCI 4510/6510
Quiz 3
October 21, 2024

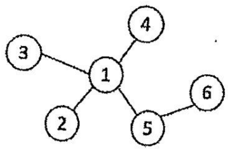
RCS ID: Lib19 @rpi.edu **Name:** Boning Li

Instructions:

- You will have **55 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** 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). Recall that in Raymond's algorithm for mutual exclusion, processes send explicit request messages for the token, and a process can access the resource only when it holds the token.

Consider an execution of Raymond's algorithm in the tree network on the left.



1. What is the maximum number of entries that any process can have in its request queue (at any time)? Assume that when a process requests the resource, it does not request it again until after it has accessed the resource.
2. Suppose the token is at process 6, and process 6 is accessing the resource. Give an execution of Raymond's algorithm that results in a request queue of this maximum length at some process. Explain which messages are sent up until the point the queue achieves the maximum length.

1. The maximum number is 4. Assume 1 is accessing the resource and will keep using it for a long time. Then, all the remaining nodes send a request. The queue of ~~a~~ 1 would be like $\boxed{2|3|4|5}$. The queue of 5 can be $\boxed{5|6}$

2. Initially, 6 has the token and is accessing the resource.
 1). 1 wants the resource, so it sends a request to 5 and sets its queue = $\boxed{1}$
 2). 5 receives 1's req and sends a req to 6. It sets its queue = $\boxed{1}$
 3). 6 receives 5's req and sets its queue as $\boxed{5}$
 4). 2 wants the resource. It sends a req to 1.
 5). 1 receives 2's req and sets its queue = $\boxed{1|2}$. 1 doesn't request for 5.
 6). 3 wants the resource. It sends a req to 1.
 7). 1 receive's 3's req and sets its queue = $\boxed{1|2|3}$
 8). 4 wants the resource. It sends a req to 1.
 9). 1 receives 4's req. It sets its queue = $\boxed{1|2|3|4}$

Question 2 (8 points). Recall that in the deadlock-free version of Maekawa's mutual exclusion algorithm, if process p is currently locked for a request from process q, and it receives a request with a higher timestamp from process r, it will send a FAIL message to process r to let process r know that another process has p's lock. If a process receives both a FAIL message and an INQUIRE message, it will relinquish its locks.

Consider a 9-node network with grid quorums. Give an example execution where a process receives a FAIL message but does not receive a ~~RELINQUISH~~ Inquiry message (and therefore, does not give up its locks), and this process eventually is granted access to the resource.

1 2 3

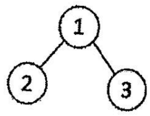
4 5 6

7 8 9

$S_1 = \{1, 2, 3, 4, 7\}$, $S_2 = \{1, 2, 3, 5, 8\}$

Assume P_3 is currently locked for a request from P_1, whose T.O. Lampon is (2, 1). Now, P_2 also wants the resource and sends a request to all of its quorums, including P_3. ~~Its~~ Its T.O. Lampore is (3, 2). Then, P_3 will send a FAIL msg to P_2. Assume ~~o~~ no other processes want the resource. After P_1 finishes using the resource, ~~P_3 will~~ it will send RELEASE msg to all its quorums, and P_3 will grant P_2 to use the resource. Finally, P_2 is granted access to the resource.

Question 3 (8 points). In class, we studied the Chandy-Lamport global snapshot algorithm in a network that is a complete graph, i.e., any process can send a message to any other process. Suppose instead, the communication network topology is as shown in the figure to the right, where p_1 and p_2 can exchange messages, and p_1 and p_3 can exchange messages, but p_2 and p_3 cannot send messages to one another.



Does the Chandy-Lamport algorithm generate a consistent global state in this network? Answer YES or NO and justify your answer.

YES. Since the communication between 2 and 3 is indirect now, they have to ~~go throu~~ use 1 as a proxy. Assume 3 doesn't send msg to 2. In this case the Chandy-Lampore algorithm will generate a consisten global state. The same follows if 2 doesn't send any msg to 3. Now, if 2 and 3 communicates through 1. the communication can be divided into two parts: 2 communicates with 1, and 1 communicates with 3. In this case, we can safely add an edge between 2 and 3, and this also gives a consistent global state. As a result, the Chandy-Lampore algorithm still generates a consisture global state in this network.

Question 4 (6 points). An algorithm that solves the Two Generals Problem satisfies the following three properties:

- **Agreement:** No two processes decide on different values.
- **Validity:**
  1. If all processes have input 0, then 0 is the only decision value.
  2. If all processes have input 1 and all messages are received, then 1 is the only decision value.
- **Termination:** All processes decide after a finite number of messages are sent.

We learned that there is no algorithm that solves the Two Generals Problem in a system with unreliable communication (and no process failures).

1. Give an algorithm for the Two Generals Problem that satisfies agreement and validity in this system model. Justify your answer.
2. Give an algorithm for the Two Generals Problem that satisfies agreement and termination in this system model. Justify your answer.

1. Initially, both nodes send their input to ~~the~~ each other. ~~The~~ After receiving each other's message, both of them will send back a value ack. ~~(A,1)~~ For ~~a~~ example, ack(A,1) means that B receives the message from A that A's input is 1. Any node will ~~s~~ continously sent its input if an ack is not received after a timeout interval. After receiving value ack, they will send a final ack to each other following the same manner. After both final acks are received. The value will be ~~obta~~ obtained using logical AND. This ensures they both agree on the same output. If both are 0s, or 1s, the AND will give 0 ~~or~~ or 1, respectively.

2. ~~Assume the algorithm terminates after N rounds, and each round both nodes send a msg to each other. Initially, both nodes send their value to each other. In the following, Initially, both nodes send their value to each other. In the following rounds, each of them sends ack1, ack2, ... ack(n-1) to the other.~~ ~~they send cummulative cummulative ack to the other. For example, if A sends ack1, then B responds with ack 2.~~ ~~If all acks are well-received~~ ~~In the final round, they~~ The algorithm only takes one round. Each of them send their value to each other. Then, both of them output 0. This ensures they always ~~a~~ decide on 0 and it terminates within one round.