Quiz 2

● Graded

Student

Boning Li

Total Points

27 / 30 pts

Question 1

Question 1

10 / 10 pts

✔ **– 0 pts** Correct

Question 2

Question 2

10 / 10 pts

✔ **– 0 pts** Correct - solid justifications

Question 3

Question 3

7 / 10 pts

Correct response (NO)

✔ **– 3 pts** Correct response (NO), but justification is incorrect - execution does not follow algorithm

Distributed Systems and Algorithms — CSCI 4510/6510
Quiz 2
September 30, 2024

RCS ID: lib19 @rpi.edu **Name:** Boning 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** 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). In class, we studied the pull algorithm (Cristian's algorithm) for physical clock synchronization in an asynchronous system, where each communication link has a minimum delay *min*. Suppose the communication link from p to S has minimum delay min_{pS}, and the communication link from S to p has minimum delay min_{Sp}. Consider a pull algorithm in this system, where the process p sends a request to S for the current time at S, and p can measure the round trip time T_{RT} until the response from S is received.

1. What should p set its clock to when it receives the response? Show your work.
2. What is the accuracy of p's clock after it sets it? Show your work.

$$P \underset{min_{sp}}{\overset{min_{ps}}{\rightleftarrows}} S$$

The earliest point p could receive the time t is $t+min_{sp}$

The latest point is $t+T_{Rt}-min_{ps}$

The time interval is $[t+min_{sp},\ t+T_{Rt}-min_{ps}]$

$t+T_{RT}-\cancel{min_{ps}}\,min_{ps}-t-min_{sp} = \cancel{t}\,T_{RT}-min_{ps}-min_{sp}$

① As a result, p should set its clock to

$$t+min_{sp}+\frac{T_{RT}-min_{ps}-min_{sp}}{2} = t+\frac{T_{RT}+min_{sp}-min_{ps}}{2}$$

② The accuracy is $\pm\left(\frac{T_{RT}-min_{ps}-min_{sp}}{2}\right)$

Question 2 (10 points). Consider the matrix clocks in the Wuu-Bernstein algorithm for the Replicated Log Problem. For each statement below, indicate whether the statement is TRUE or FALSE. If the statement is TRUE, provide a justification. If the statement is FALSE, describe a counter-example.

1. For any process p_i, it always holds that $T_i(i,k) \geq T_i(j,k)$ for $j = 1 \ldots N$, $k = 1 \ldots N$.
2. For any pair of processes p_i and p_j, $i \neq j$, it always holds that $T_i(i,i) \geq T_j(i,i)$.

① ~~No.~~ ~~The following~~ $\boxed{\text{Yes}}$ The ith row of T_i refers to the direct knowledge of site i about ~~how~~ events in other sites, while other rows refers to the indirect knowledge.

For $j=i$, we have $T_i(i,k) = T_i(j,k)$ for all k.

For $j \neq i$, $T_i(i,k)$ will be updated on ~~each~~ each receive event. We have

$$T_i(i,k) = \max[T_i(i,k), T_j(j,k)]$$

The indirect knowledge ~~was~~ is updated with other indirect knowledge.

$$T_i(r,s) = \max[T_i(r,s), T_j(r,s)]$$

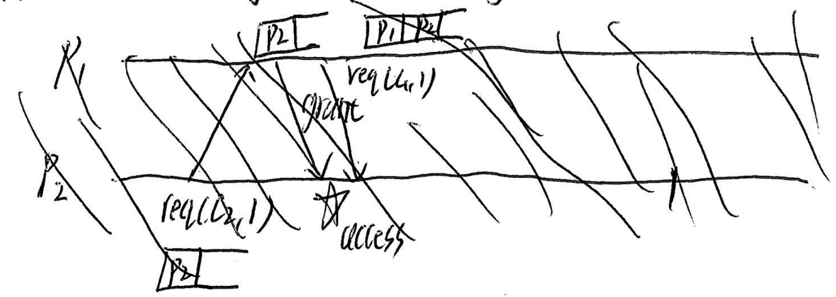
Therefore, $T_i(i,k) \geq T_j(j,k)$ for $j, k = 1 \cdots N$

② $\boxed{\text{Yes}}$ $T_i(i,i)$ is recorded by i locally, so it always reflects the latest event ~~at~~ occuring at i, while $T_j(i,i)$ is an indirect knowledge about site i at site j. If site i sends a message to j everytime an update is made, then $T_i(i,i) = T_j(i,i)$. If there are many local events at i while i never tells j or any other site, $T_i(i,i) > T_j(i,i)$.

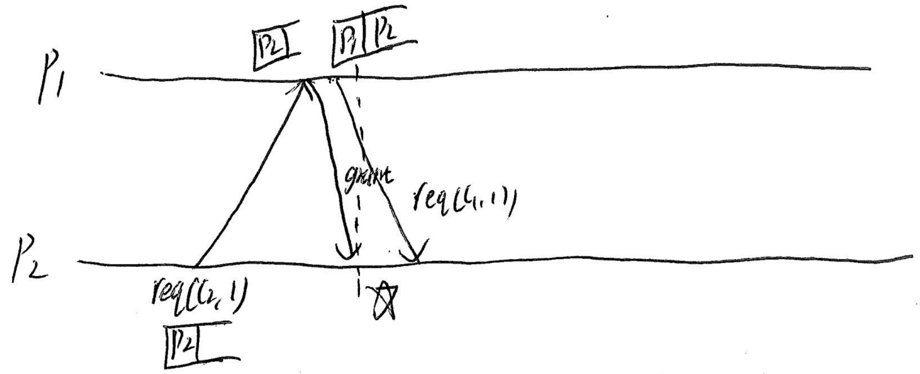
Therefore, $T_i(i,i) \geq T_j(i,i)$

Question 3 (10 points). Recall that in Lamport's mutual exclusion algorithm, a process accesses the resource when it has received GRANT from every other process and its request is at the head of its own priority queue. Suppose p_i has just started accessing the resource. Is it necessarily the case that p_i's request is at the head of of the priority queue at every process? Answer YES or NO and justify your answer.

No. Think of the following counter-example.



In this case P2 has just started accessing the resource, but the priority queue of P1 is |P1|P2|



P2 will start as long as it receives grant. However, it's possible that a request is on the way when grant arrives. In that case, P1 has already put itself at the head of queue and P1 is waiting for a grant.