Quiz 1

Graded

Student

Boning Li

Total Points

25 / 30 pts

Question 1

Question 1

6 / 6 pts

- ✔ **– 0 pts** Correct

Question 2

Question 2

4 / 6 pts

- ✔ **– 2 pts** Answer is correct (NO) but not firmly demonstrated by justification/counterexample

Question 3

Question 3

10 / 10 pts

- ✔ **– 0 pts** Correct

Question 4

Question 4

5 / 8 pts

- ✔ **– 3 pts** One answer not computed (algebra shown)

Distributed Systems and Algorithms — CSCI 4510/6510
Quiz 1
September 12, 2024

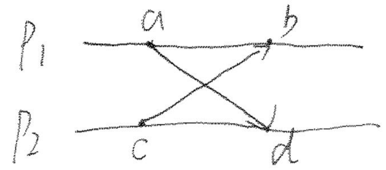
RCS ID: libl9 @rpi.edu **Name:** Boning Li

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 (6 points). Recall that in Lamport's Logical Clock algorithm, each process p_i has an integer c that is initialized to 0. Suppose instead, each process's clock is initialized to a random number between 0 where N is the number of processes in the system. The algorithm remains otherwise unchanged.

Does this modified algorithm always satisfy the weak clock condition, i.e., if $e \to f$ then $C(e) < C(f)$? Answer or NO and justify your answer.



C_1 initialized as ~~0~~ 2 $C(a)=3$ $C(b)=4$

C_2 initialized as ~~0~~ 1. $C(c)=2$ $C(d)=$ ~~3~~ 4

If e and f occur on the same process P_i, then $C(e) < C(f)$ due to increment.

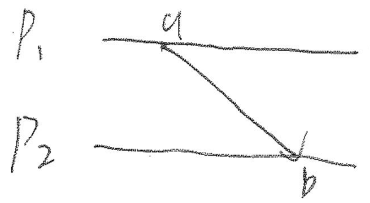
If e occurs at P_i and f occurs at P_j, then $C(f) = \max(C(e)+1, C_j(f)+1)$.
as a result, $C(e) < C(f)$.

Therefore, this modified algorithm always satisfies the weak clock condition.
The answer is YES.

Question 2 (6 points). Recall that in Vector Clock algorithm, each process p_i has a vector clock VT_i that is initialized to the **0** vector. Suppose instead, each process's clock is initialized to be a vector of random numbers between 0 and N, where N is the number of processes in the system. The algorithm remains otherwise unchanged.

Does this modified algorithm always satisfy the strong clock condition, i.e., $e \to f$ if and only if $C(e) < C(f)$? Answer YES or NO and justify your answer.

In case where e and f occur on the same p_i, $e \to f$ gives $C(e) < C(f)$ since $e.VT(i) < f.VT(i)$ due to increment, and for all other $j \neq i$, $e.VT(j) \le f.VT(j)$ due to inheritance. When e occurs on p_i and f on p_j, consider the following case.



initially $V_1 = (2, 2)$

initially $V_2 = (0, 0)$

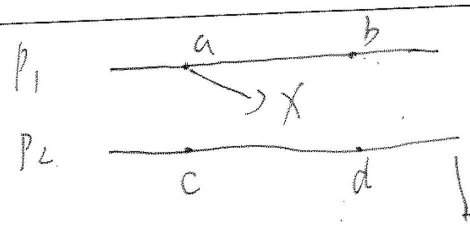
On event a, ~~V(a)~~ $a.VT = (3, 2)$ and is sent to P_2.

Upon receiving m on event b, $b.VT = \max[(0, 1), (3, 2)] = (3, 2)$

Then we have $a \to b$ but $a.VT = b.VT$.

Therefore, the answer is NO.

Question 3 (10 points). All of the space-time diagrams I have drawn in class illustrate *reliable* comm
i.e., every message that is sent is eventually received. Consider a distributed system in which commun
unreliable; some messages may be sent but never received (messages may be lost). Does Lamport's Logi
algorithm always satisfy the weak clock condition in this system: if $e \to f$ then $C(e) < C(f)$? Justify your



If e and f occur on the same process p_i, then $C(e) < C(f)$ ~~dute~~ due to the increment, since local events don't require communication.

If e ~~oo~~occurs on p_i and f ~~oo~~occurs on p_j, then $e \to f$ states that e causally affects f. When e is a send of m, and f is the receive of m. In the case where ~~f~~ m arrives at p_j, we have $C(e) < C(f)$ based on the algorithm. However, if m get lost, there won't be a receive event f on p_j, since p_j has no idea that p_i sends it a msg. In this case, there's no event f and no such relation $e \to f$ exists. As a result, as long as we have relation $e \to f$, $C(e) < C(f)$ will hold. Lamport's Logical clock algorithm still satisfies the weak condition.

Question 4 (8 points). Suppose the push algorithm is used to set the time at process p_i's physical clock. The communication link between the time server S and process p_i is synchronous, with a minimum latency of 10 and a maximum latency of 30. The time server S sends its message to p_i at time $t = 100$. Answer the following questions and show your work.

1. When p_i receives the message, what does it set its clock to?
2. What is the resulting accuracy of p_i's clock?

① Assume the round-trip time recorded by p_i is $T_{RT} \geq 20$

The ~~minimum~~ earlist time ~~required by~~ for the message to arrive at p_i is 110.

The latest time of arrival is $100+30 = 130$, or $T_{RT} - min = T_{RT} - 10$

A reasonable ~~set~~ setup could be $\frac{130+110}{2} = 120$

If $T_{RT} - 10 < 30$, the interval becomes $[t+min, t+T_{RT}-min] = [110, T_{RT}+100]$

③ In this case, reasonable setup should be $t + \frac{T_{RT}}{2} = 100 + \frac{T_{RT}}{2}$

② Given the time interval, the resulting accuracy is

$$\pm\left(\frac{T_{RT}}{2} - min\right) = \pm\left(\frac{T_{RT}}{2} - 10\right)$$