Homework 1

CS425/ECE428 Spring 2023

Due: Wednesday, Feb 15 at 11:59 p.m.

1. Consider a distributed system of four processes as shown in Figure 1. The system is synchronous, and the minimum and maximum network delays (in seconds) between each pair of processes are shown in the figure as $[min, max]$ against each channel. Assume no messages are lost on the channel, and the processing time at each process is negligible compared to network delays.

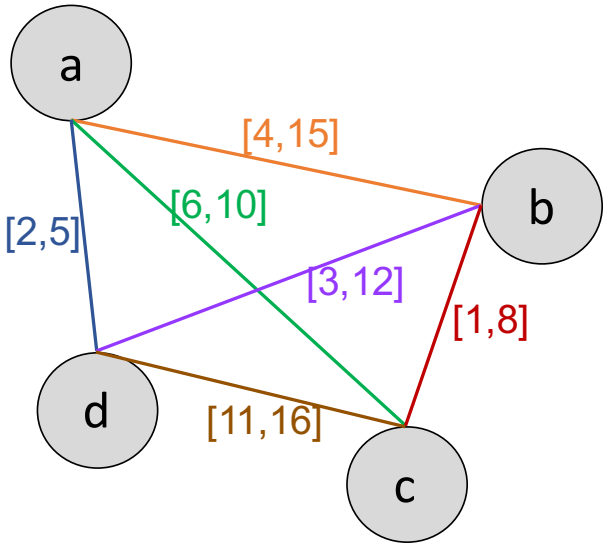


Figure 1: Figure for question 1.

(a) (3 points) Consider an all-to-all heartbeat protocol, where each process sends a heartbeat to each other process periodically every T=100s, and each process sets a timeout (computed appropriately from the known bounds on network delay) to detect failure of other processes.

Suppose that process a crashes. For every other process, calculate how long it will take to detect a's failure, in the worst case.

> **Solution: For process b - 126 seconds, For process c - 114 seconds, For process d - 108 seconds.**
> Explanation (Complete detail not required for full credit): First, we calculate the appropriate value of the timeout. The maximum time difference between two consecutive heartbeats would be when the first heartbeat reaches from process a to process b with the *minimum* delay, and the subsequent message reaches from process a to process b with the maximum delay. Hence, the timeout value should be set as heartbeat period + max delay - min delay. Now, if process a fails immediately after sending out a heartbeat, this would cause a worst case delay. The worst case failure detection time would hence be equal to the timeout value + max network delay. (Because, the timeout value would only be in effect after receiving the latest heartbeat which can take *max network delay* time). Therefore, for process **b**, worst case failure detection time is 126 seconds. For process c, worst case failure detection time is 114 seconds. For process d, worst case failure detection time is 108 seconds.

(b) (3 points) Now consider a small extension of the protocol in in Q1(a) – as soon as a process detects that another process p has crashed via a timeout, it sends a notification to all other processes about p's failure. Suppose that process a is the only process that crashes. For every other process, calculate how long it will take to detect a's failure, in the worst case.

Solution: For process b - 120 seconds, For process c - 114 seconds, For process d - 108 seconds.
Explanation (Complete detail not required for full credit): From part (a), process d detects the failure first at 108 seconds. Process c can get the message from d in the worst case after 124 seconds but this is more than the original failure detection time for c which was 114 seconds. Therefore, process c's worst case failure detection time also will not change. However, process b will get a message from process d in the worst case after 108 + 12 = 120 seconds which is less than 126 seconds. Therefore, worst case failure detection time for process b reduces to 120 seconds.

(c) (2 points) If it is known that *no more than two* processes may crash simultaneously, how would you redesign the heartbeat protocol described in Q1(a) to minimize bandwidth usage, without increasing the worst case time taken to detect the failure of a process by at least *one* alive process. [Hint: do we really need *all-to-all* heartbeats?]

Solution: If it is known that no more than two processes crash simultaneously, then it is sufficient for each process to send heartbeats to 2 other processes instead of 3. For maintaining worst case time, these two processes must be the ones with minimum failure detection time as calculated by part (a). (This is the only detail required for full credit, writing the exact two processes is not required)

(d) (2 points) Assuming the modification in Q1(c), list the minimal set of processes a must send heartbeats to, so as to minimize the worst case time taken to detect failure of a by at least *one* alive process.

Solution: a must send its heartbeat to the two processes which have the minimum worst case failure detection time. From part (a), process a should send heartbeats to processes c and d.

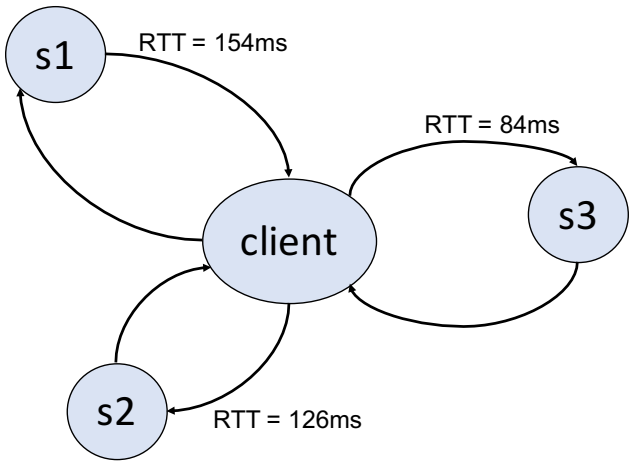


Figure 2: Figure for question 2(a).

2. (a) (4 points) Consider Figure 2. The client has an option of using any of the three authoritative sources of real time (s1, s2, or s3) for external synchronization via Cristian algorithm. The round-trip times (RTT) between the client and the three servers are shown in the figure. Assume that the observed RTT to each server remains constant across all synchronization attempts.

(i) Which server should the client choose to achieve the lowest accuracy bound right after synchronization? What is the value of this bound, as estimated by the client right after synchronization? [1 point]

Solution: Choose S3, and bound = 42ms.
Explanation (Not required for full credit): According to Christian's algorithm, the accuracy bound is directly proportional to the RTT. Therefore, the client should choose S3 to achieve the lowest accuracy bound. The value of the bound is given by $\frac{RTT}{2}$ which equals 42ms for S3.

(ii) If the client's local clock drifts at the rate of 10μs every second, what is the smallest frequency (or the longest time-period) at which the client must initiate synchronization with the server it chose in part (i), so as to maintain an accuracy bound within 250ms at all times. [3 points]

Solution: Ans: 20800 seconds
Explanation: We want the maximum skew between the clocks of the server and client to be 250ms. So, if T_s is the frequency of synchronization, then after T_s seconds, the clocks would have drifted by $0.01 \times T_s$. Also, the bound on synchronisation by part (a) is 42ms. Therefore, we want $0.01 \times T_s + 42 \leq 250$. For the smallest frequency of synchronisation, we want T_s to be maximum i.e. when the equality holds. Solving the above equation, we get $T_s = 20800$ seconds.

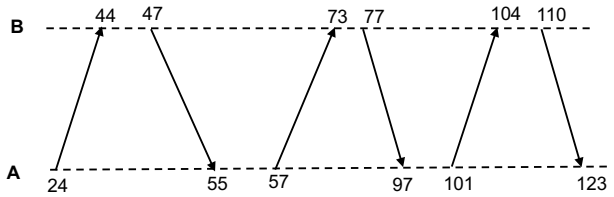


Figure 3: Figure for question 2(b).

(b) (6 points) Consider the series of messages exchanged between two servers A and B as shown in Figure 3. The local timestamps (in seconds) at the servers when sending and receiving each message are shown in the figure.

(i) Assume NTP's symmetric mode synchronization, where the send and receive timestamps for each message are recorded by both servers. Given A's knowledge of the send and receive timestamps for all six messages, what is the lowest synchronization bound (as estimated by A) with which A can compute its offset relative to B? What is the corresponding estimated offset value? [Hint: A may use *any* pair of messages exchanged between the two servers, and not just two consecutive messages to compute offsets.] *(4 points)*

Solution: Notice the hint, we can use *any* pair of messages to synchronize. $T_{Br} - T_{As}$ for the three sets of messages are 20, 16 and 3 seconds respectively. $T_{Ar} - T_{Bs}$ for the three sets of messages are 8, 20 and 13 seconds respectively. The synchronization bound is given by -

$$\frac{d_i}{2} = \frac{(t+t_i)}{2}.$$

Therefore, the lowest synchronization bound would be when $t+t_i$ has the least value which happens when we select the third message from A to B and the first message from B to A.

This gives us the lowest synchronization bound of 5.5 seconds. **(11 seconds or d_i is also acceptable)**. The corresponding offset is given by the relation -

$$o_i = \frac{(T_{Ar} - T_{Bs}) - (T_{Br} - T_{As})}{2} \implies o_i = \frac{8-3}{2} = 2.5 \text{ seconds.}$$

(ii) Now assume that A uses the same series of messages for synchronization via Cristian algorithm: messages sent from A to B are requests, and messages from B to A are responses carrying the timestamp when B received the last request. What is the tightest synchronization bound (as estimated by A) with which A can compute its offset relative to B? *(2 points)*

Solution: Pick the last pair of messages, and bound = 11 seconds.
Explanation: If we are using Christian's algorithm, then the tightest bound would be achieved by the messages whose RTT is the least, since the bound is directly proportional to RTT and is given by $\frac{RTT}{2}$. In this example, the least RTT is for the last pair of messages, and the bound is given by $\frac{123-101}{2} = 11$ seconds.

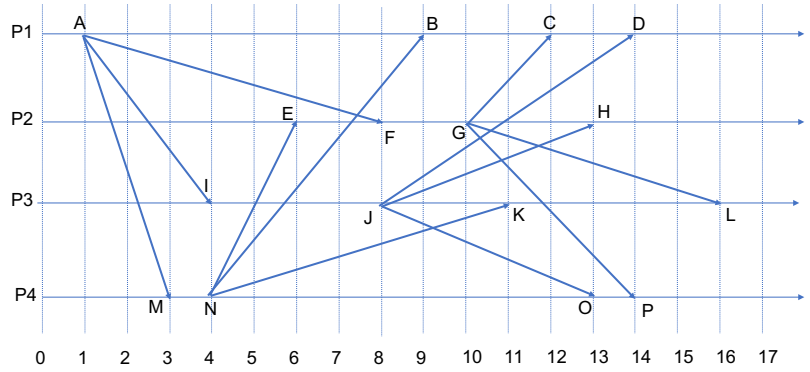
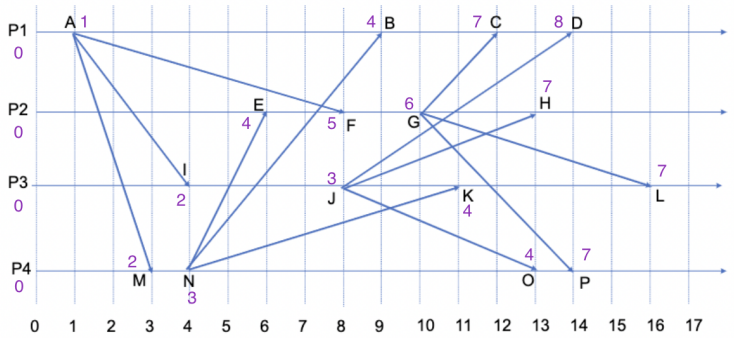


Figure 4: Timeline for questions 3 and 4.

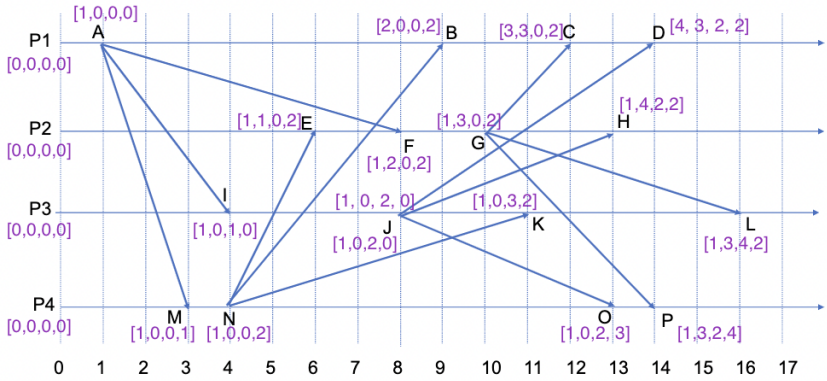
3. The timeline in Figure 4 shows 16 events (A to P) across four processes. The numbers below indicate real time.

(a) (2 points) Write down the Lamport timestamp of each event.

Solution:



(b) (4 points) Write down the vector timestamp of each event.



Solution:

(c) (2 points) List all events considered concurrent with (i) B, (ii) N

Solution: The events concurrent with B are E, F, G, H, I, J, K, L, O, P. The events concurrent with N are I and J.

4. (a) (4 points) Consider the timeline and events in Figure 4 again. Suppose that P3 initiates the Chandy-Lamport snapshot algorithm at (real) time 7. Assuming FIFO channels, write down *all* possible consistent cuts that the resulting snapshot could capture. You can describe each cut by its frontier events.

Solution: If P3 initiates the snapshot at time 7, then the frontier event at P3 would be I. For process P1, the first snapshot message has to arrive before D but after A (since channels are FIFO). So, the frontier event can be either A, B or C. For process P2, the frontier event similarly, can be E, F or G. For process P4, the frontier event can only be N. Therefore, all possible consistent cuts possible are - [A, E, I, N], [A, F, I, N], [A, G, I, N], [B, E, I, N], [B, F, I, N], [B, G, I, N] and [C, G, I, N]. (Note that [C, E, I, N] and [C, F, I, N] are not consistent)

(b) (4 points) Write *all* possible states of the incoming channels at P3 and at P4 that the above snapshot could record. You can denote each message by its send and receive event ids.

Solution: At P3,
Incoming channel from P1: Always Empty
Incoming channel from P2: Either empty or contains $G - L$.
Incoming channel from P4: Always have $N - K$.
At P4,
Incoming channel from P1: Always empty
Incoming channel from P2: Either empty or contains $G - P$,
Incoming channel from P3: Always empty.

5. (a) (2 points) Consider the timeline of events {A,B,...F} across two processes as shown in Figure 5. List all possible linearizations for this system that includes each event.

Solution: There are 4 possible linearizations for this system - [A, B, D, E, F, C], [A, D, B, E, F, C], [A, D, E, B, F, C] and [A, D, E, F, B, C].

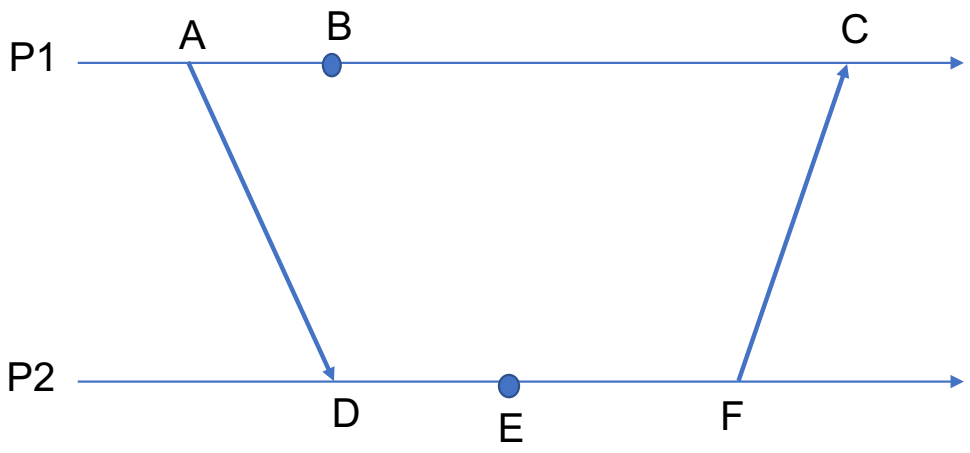


Figure 5: for question 5

(b) (2 points) What is the total number of consistent global states that can be possibly captured for the above system? Identify each of them by the frontier events of the corresponding cuts.

Solution: There are 10 possible consistent global states that can be captured for the above system. The frontier events corresponding to these states are - [], [A], [A, D], [A, E], [A, F], [B], [B, D], [B, E], [B, F] and [C, F].