Homework 1

CS425/ECE428 Spring 2022

Due: Wednesday, Feb 9 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.

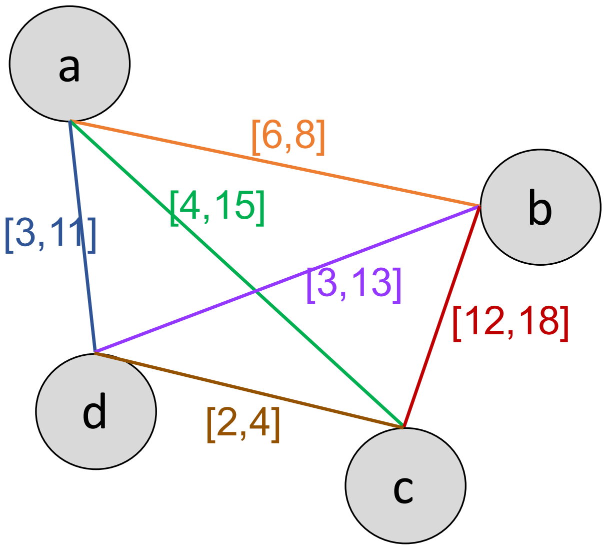

Figure 1: Figure for question 1.

   (a) (3 points) Consider an all-to-all heartbeat protocol, where each process sends a heartbeat to each other process periodically every T=50s, 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.

   (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.

   (c) (2 points) If it is known that *at most two* processes may crash within a few hours of each other, 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?]

   (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.

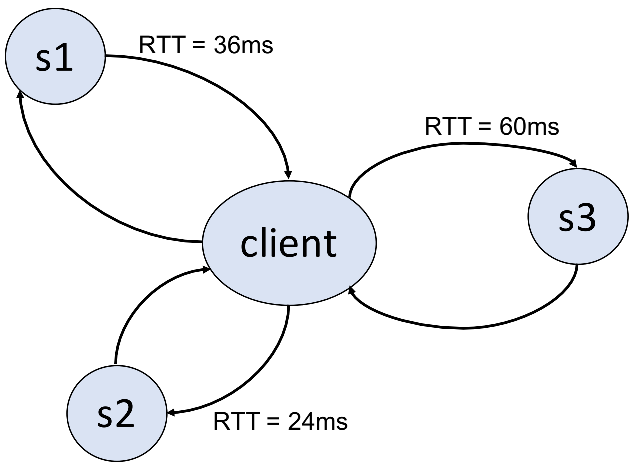

Figure 2: Figure for question 2(a).

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]

   (ii) If the client's local clock drifts at the rate of $3\mu s$ 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 90ms at all times. [3 points]

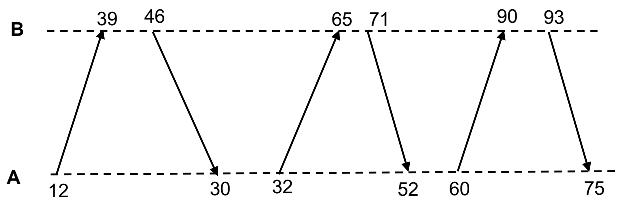

Figure 3: Figure for question 2(b).

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

   (i) Assume 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)*

   (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)*

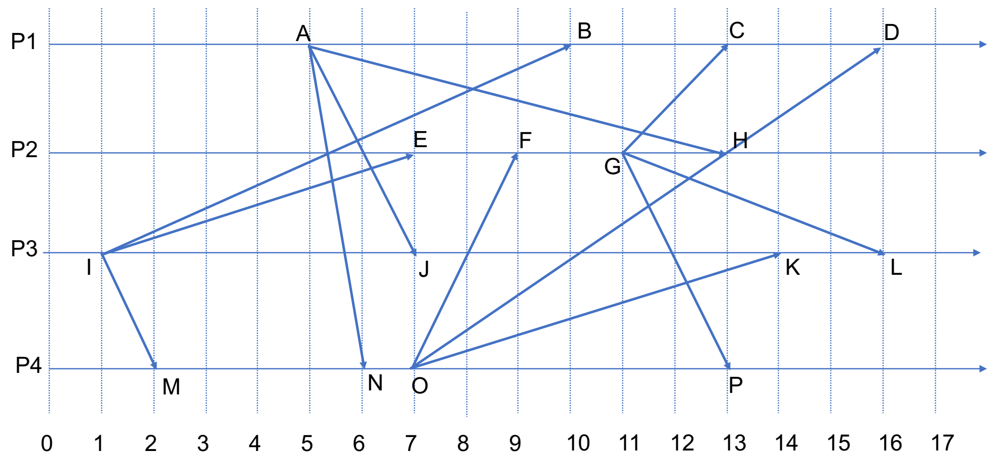

Figure 4: Timeline for questions 3 and 4.

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.
   (b) (4 points) Write down the vector timestamp of each event.
   (c) (4 points) List all events considered concurrent with: (i) A, (ii) F, (iii) K, and (iv) N.

4. (a) (4 points) Consider the timeline and events in Figure 4 again. Suppose that P2 initiates the Chandy-Lamport snapshot algorithm at (real) time 8. Assuming FIFO channels, write down *all* possible consistent cuts that the resulting snapshot could capture. You can describe each cut by its frontier events.
   (b) (6 points) Write *all* possible states of the incoming channels at P1 and at P3 that the above snapshot could record. You can denote each message by its send and receive event ids.