Homework 1

CS425/ECE428 Spring 2020

Due: Thursday, Feb 13 at 11:59 p.m.

1. Consider a distributed system of five processes as shown in Figure 1. The system is synchronous, and the minimum and maximum network delays (in seconds) between process a and each of the other processes are shown in the figure as $[min, max]$ against each channel from a. Message delivery is guaranteed.

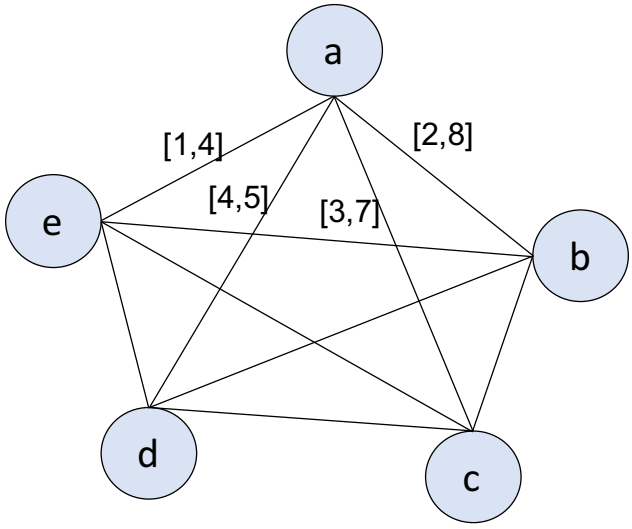

Figure 1: Figure for question 1.

   (a) (3 points) Consider an all-to-all heartbeat protocol, where each process sends a heartbeat to each other process periodically every T=20s, and each process sets a timeout (computed from the known bounds on network delay) to detect failure of other processes.

   Suppose that process a crashes. For each other process, calculate what is latest time it will take to detect a's failure, in the worst case.

   (b) (3 points) If the all-to-all heartbeat protocol is replaced with an all-to-all ping-ack protocol, where each process sends a ping to each other process every 20s. When receiving a ping, each process immediately responds with an ack. Failure is detected by setting a timeout for the ack using known bounds on network delay.

   Again, suppose that process a crashes. For each other process, calculate what is the latest time it will take to detect a's failure, in the worst case.

   (c) (2 points) If it is known that no more than three 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 condition in Q1(c), list the minimal set of nodes 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. [2 points]

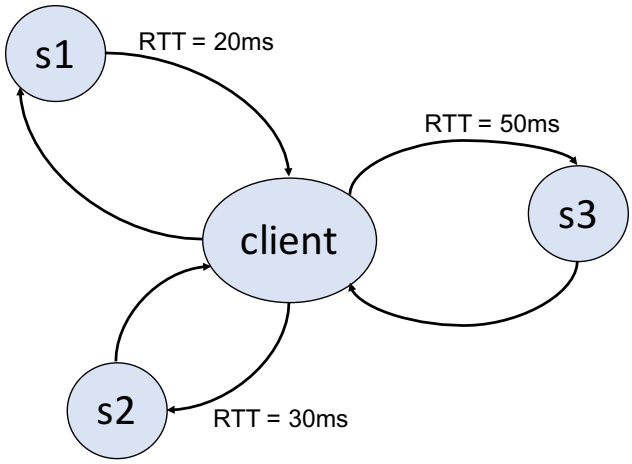

Figure 2: Figure for question 2(b).

2. (a) (4 points) Consider the figure in 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. If the client's local clock drifts at the rate of 2μs every second, what is the smallest frequency (or the longest time-period) at which the client must initiate synchronization with its server of choice, in order to maintain an accuracy bound of 100ms.

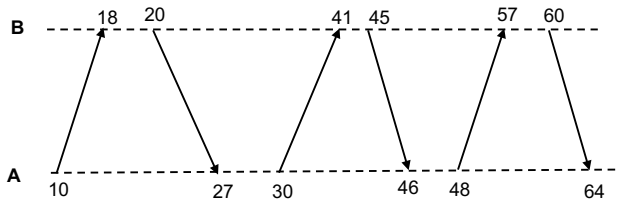

Figure 3: Figure for question 2(b).

(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 tightest accuracy 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 accuracy bound (as estimated by A) with which A can compute its offset relative to B? *(2 points)*

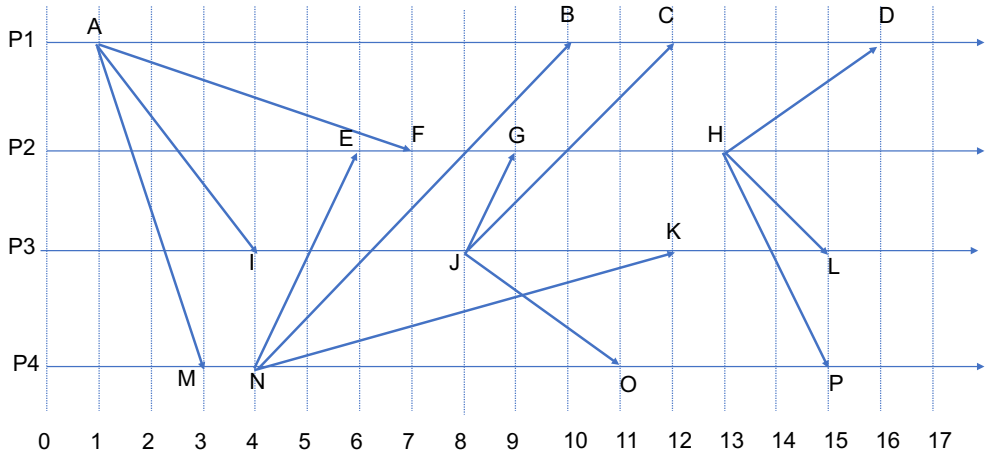

Figure 4: Timeline for questions 3 and 4.

3. (a) (3 points) Looking at fig. 4, write down the Lamport timestamp of each event.

   (b) (4 points) Using fig. 4, write down the vector timestamp for each event.

   (c) (3 points) List all the concurrent events.

4. (a) (6 points) Looking at fig. 4, suppose that P2 initiates the Chandy-Lamport snapshot algorithm at time 11. Write down *all* possible consistent cuts that the resulting snapshot could capture. (You can describe each cut by its frontier events.)

   (b) (4 points) Answer the following questions:

      (i) Provide a real-world usecase of clock synchronization, explaining why it needs clock synchronization, and whether there are other alternatives that can be used instead.

      (ii) Provide a real-world usecase for computing global snapshots, explaining why it needs global snapshots.