CS425/ECE428 Homework 1

Due: 11:59 p.m. on Friday, September 17, 2021

This assignment has 3 pages and 5 questions, worth a total of 60 points. Solutions must be submitted via Gradescope. Solutions must be typed, not hand-written, but you may include hand-drawn diagrams.

You must acknowledge any sources you used to arrive at your solutions, other than the course materials and textbook. If you work in a group on homework assignments, please list the names of your collaborators, but make sure to write your own solution.

1. Ring Around the Rosy . . . . *12 points*

   Consider a ring-based failure detection scheme with processes $p_1, \ldots, p_n$. Process p_i sends a heartbeat every T seconds to process p_{i+1}, with process p_n sending a heartbeat to p_1.

(6) (a) This failure detection mechanism is not complete because it does not tolerate failure of two neighboring processes. How would you modify it so that failure of two processes is always detected? Be explicit in your description and compare the bandwidth cost of your modified algorithm to the original ring-based failure detection.

Your modification should still maintain a ring structure for the processes.

(4) (b) Suppose now that we set a p_i only sends a heartbeat to p_{i+1} after receiving a heartbeat from p_{i-1}, and then waiting T seconds. I.e.:

1. p_1 sends a heartbeat to p_2
2. p_2 receives the heartbeat, waits T seconds, sends a heartbeat to p_3
3. p_3 receives the heartbeat, waits T seconds, sends a heartbeat to p_4
4. . . .

Assume that there is exactly one such heartbeat (or token) circulating in the ring at any time. What should the timeout be set to, given a maximum one-way delay of Δ, to ensure accurate detection? What would be the maximum detection time?

(2) (c) The token scheme above ensures that a failure of one process is detected by every other process, but it does not directly reveal *which* process failed. Sketch out how you would modify the algorithm so that the identity of the failed process can be discovered. The modification should preserve the token-based design described above.

2. Xtreme Drift . . . . *6 points*

   Usually drift is small enough so that it only needs to be corrected for occasionally, but this questions considers larger drift.

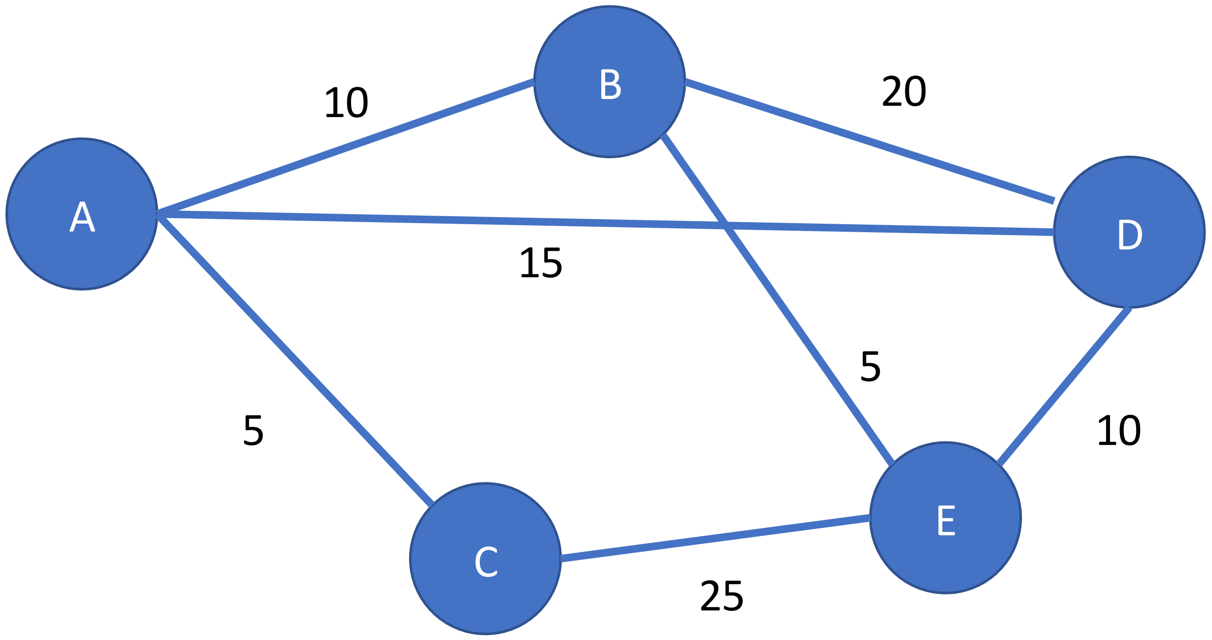
(2) (a) Consider a heartbeat protocol in a system with a maximum one-way delay of Δ and a period of T. Suppose that the clock drift is bounded by 1% (i.e., clocks lose / gain at most 1s every 100s relative to each other). What should be the value of the timeout to ensure accurate detection?

(2) (b) What would be the maximum detection time?

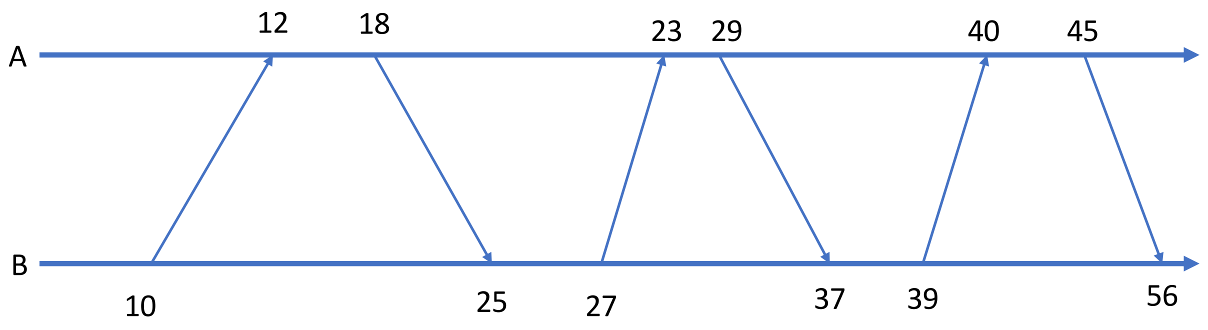
(2) (c) Client A uses Cristian's algorithm to synchronize with server B. If the RTT between A and B is 20 ms, how often should Cristian's algorithm be re-run to keep the clocks synchronized to within 100 ms, if the maximum drift rate is 1%?

3. You Say Go Slow, I Fall Behind . . . . *18 points*

(6) (a) Consider the following diagram that lists five servers (A, B, C, D, E) and the RTT for each link between them. Describe which servers should synchronize with which other servers in order to minimize the worst-case skew between any pair of servers. What would the worst-case skew be in this case?



(6) (b) Consider the following diagram that lists transmission and reception time of NTP messages between server A and B in each server's clock. Assuming no clock drift, what is an upper bound on the one-way transmission time of each of the six messages? Hint: you may want to consider the entire diagram to come up with your answer, not just adjacent message pairs.



(6) (c) Based on your answer in part (a), what is the maximum skew of server B with respect to server A? What is the minimum? By how much would you change the time of server B and in which direction?

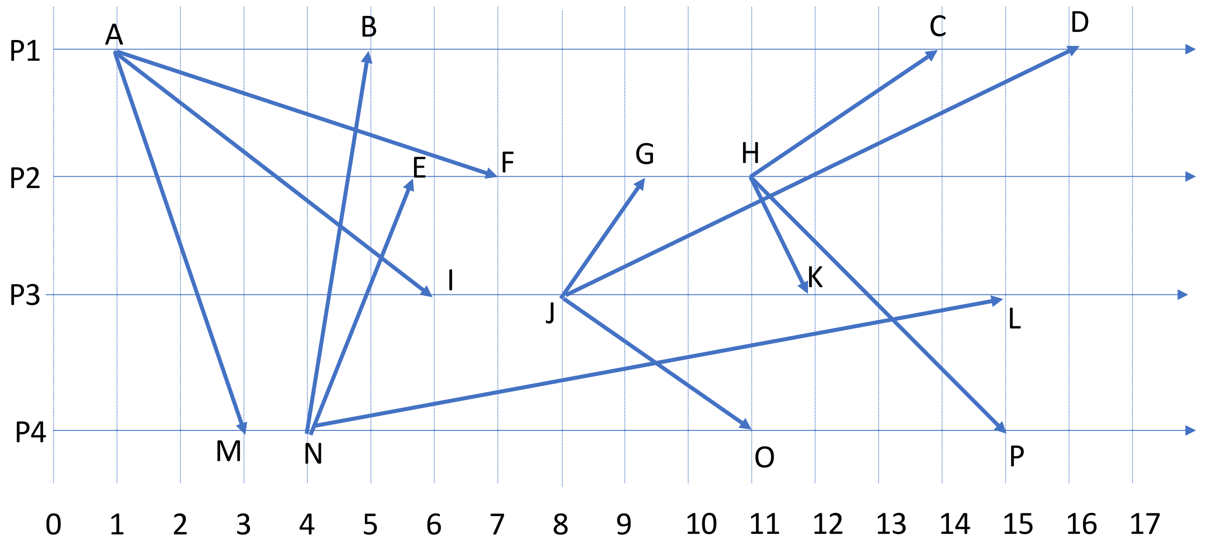


Figure 1: Event timeline for Questions 3 and 4

4. Teach Me How To Be Sensible and Keep The Timestamps Logical .... *18 points*
   The timeline in Figure fig. 1 shows 16 events (A to P) across four processes. The numbers below indicate real time.

(6) (a) Write down the Lamport timestamp of each event

(6) (b) Write down the vector timestamp of each event

(6) (c) List all events concurrent with event (i) C (ii) I (iii) O

5. Sna-sna-snapshots everybody! .... *6 points*
   Consider the timeline and events in Figure 4 again. Suppose that P3 initiates the Chandy-Lamport snapshot algorithm at (real) time 9. Assuming FIFO channels, write down all possible consistent cuts that the resulting snapshot could capture. You can describe each cut by its frontier events.