CS425/ECE428 Homework 1

Solutions

This assignment has 9 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 $p_1, \dots, p_n$. 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.

Solution:
To ensure failure of two processes is always detected, process p_{i+1} may also send a heartbeat every T seconds to process p_i, with process p_1 sending a heartbeat to p_n. In this case, the bandwidth cost of the modified algorithm is twice as large as the original detection scheme since now neighbor processes send heartbeats to each other.

(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?

Solution:
The timeout should be set to $n(T + \Delta)$, assuming there are n processes in total.

Explanation: Suppose process p_1 receives a heartbeat from process p_n at timestamp t_0 and every process in the system is functional, the latest timestamp when process p_1 receives the next heartbeat from process p_n would be $t_0 + n(T + \Delta)$ given the rule. Therefore, the timeout should be equal to the difference of the two timestamps $(= n(T + \Delta))$.

The maximum detection time would be $\Delta + n(T + \Delta)$.

Explanation: the worst case of failure detection is that process p_n crashes right after it sends a heartbeat to process p_1. With timeout set to $n(T + \Delta)$, it takes at most $\Delta + T_{timeout}$ for process p_1 to detect the failure of process p_n. Therefore, the maximum detection time should be set to $\Delta + n(T + \Delta)$.

(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.

Solution:
Modification: When a process P_i does not receive a heartbeat after its timeout, it may send a token containing the message "recover P_i" to P_{i+2}. Every node passes this token to their neighbor. If P_i receives its recovery token back later, then we know that the crashed process is P_{i+1}.
For example, if P_3 has not received a token within timeout, it sends "recover P_3" to P_5, which passes it to P_6, P_7,..., etc. The recovery token will only get back to P_3 if P_4 was the one that failed. If P_3 does not receive back its the recovery token, then we know that a process other than P_4 is.
Notice that after a failure all processes will time out and each one will launch this detection token. The one that precedes the crashed process will be the only one that identifies the failed process, but once the identification has been made, it can forward the identity of the failed process to others, if needed.

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?

Solution: $(T + \Delta) * 1.01$

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

Solution: $\Delta + (T + \Delta) * 1.01$

(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%?

Solution:
Let the time period of synchronization be T seconds. The accuracy bound right after synchronization is $\frac{RTT}{2}$. Since there is a clock drift, the accuracy bound keeps increasing, until the next synchronization message is received after T seconds. The bound just before synchronization is therefore the highest, and is given by (drift rate) * T + $\frac{RTT}{2}$.

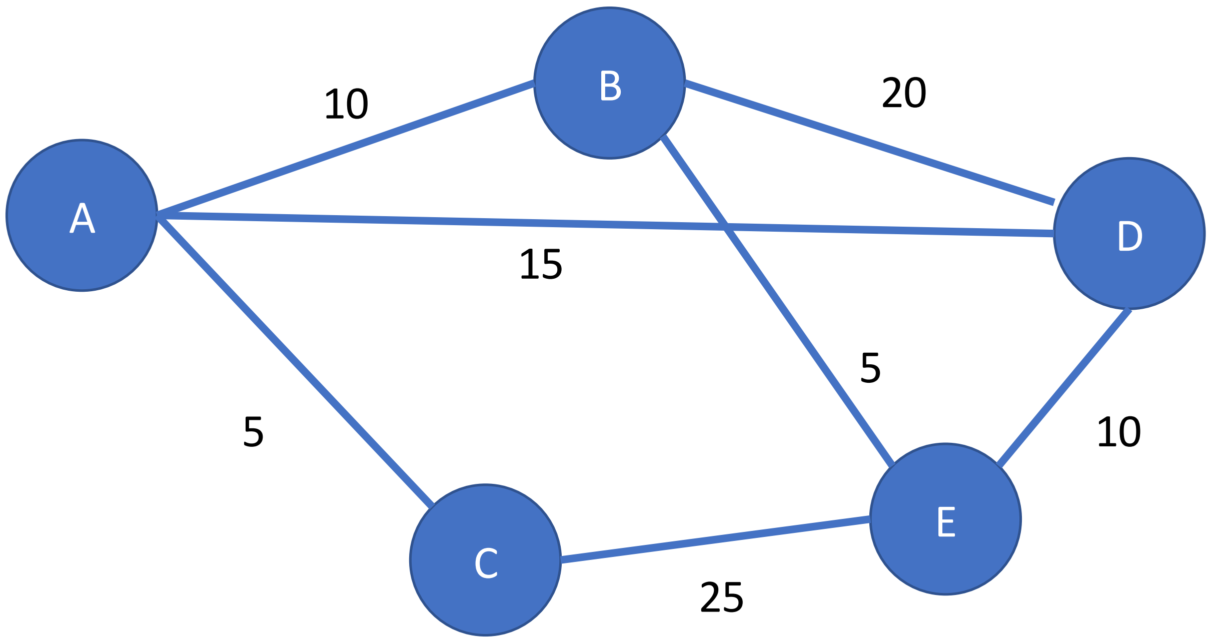
We require this value to be less than or equal to 100ms. To compute the largest allowed T, we set this value to be 100ms. Therefore, we get

$$0.01 * T + \frac{20}{2} = 100$$

$$T = 9000\, ms = 9\,\text{seconds}$$

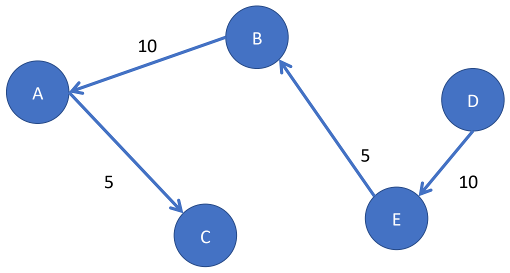
3. You Say Go Slow, I Fall Behind . . . . *18 points*

(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?

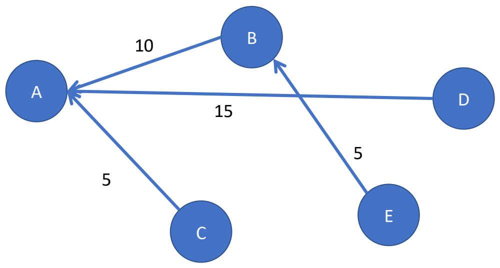


Solution: If server X synchronizes with Y then their skew will be at most 1/2 of the RTT between them. For servers that do not directly synchronize, the skews will be bounded by the the sum of the skews on the path of servers that synchronize. Consequently, want to find a spanning tree on this graph that has the smallest *diameter*, which is the maximum distance between any two nodes along tree edges, with the RTT acting as the tree weights.

There are two solutions that have a minimal diameter. The first has the spanning tree $C \rightarrow A \rightarrow B \rightarrow E \rightarrow D$. The largest distance along this tree is between C and D, measuring $5 + 10 + 5 + 10 = 30$. This means that the worst-case skew between A and D would be 15 ms.

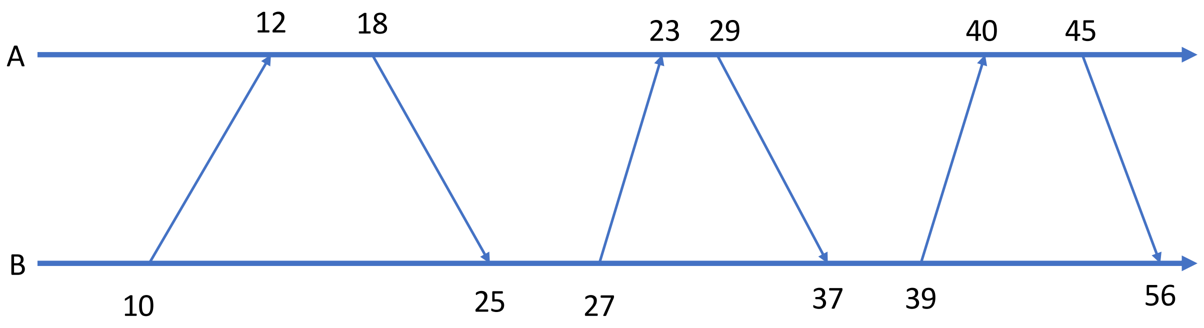


The second solution has a tree rooted at A. A synchronizes with B, C, and D, and E synchronizes with B. The longest distance in this tree is $E \leftarrow B \leftarrow A \rightarrow D$ which is again 30, so the worst-case skew is 15 ms.



In general, a minimum spanning tree algorithm is *not* what we want because it minimizes the sum of *all* the weights, rather than the weights along the longest path. The first solution happens to be a minimum-weight spanning tree as well as a minimum diameter one; the second solution has the sum of weights being 35 but a diameter of 30.

(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.



Solution:
Let t_1, t_2, t_3, t_4, t_5, t_6 be the one-way transmission time of each of the six messages, in the order from left to right. For each pair of messages in opposite directions we can compute the sum of the transmission times as the difference between the time measured on the sending and receiving side. For example, message 1 is sent at time 10 (B) and message 4 is received at time 37 (B), which means that 27 time steps elapsed between these two events. On the other hand,

message 1 was received at time 12 (A) and message 4 was sent at time 29 (A), so there was 17 time steps that elapsed between these two. This defines the equation:

$$27 = t_1 + 17 + t_4$$

or

$$t_1 + t_4 = 27 - 17 = 10$$

Since we know that $t_1 \geq 0$ and $t_4 \geq 0$, we can use this to produce two inequalities:

$$t_1 \leq 10$$

$$t_4 \leq 10$$

Doing this for every pair, we have:

$$t_1 + t_2 = (25 - 10) - (18 - 12) = 9 \quad \Rightarrow \quad t_1, t_2 \leq 9$$

$$t_1 + t_4 = (37 - 10) - (29 - 12) = 10 \quad \Rightarrow \quad t_1, t_4 \leq 10$$

$$t_1 + t_6 = (56 - 10) - (45 - 12) = 13 \quad \Rightarrow \quad t_1, t_6 \leq 13$$

$$t_2 + t_3 = (23 - 18) - (27 - 25) = 3 \quad \Rightarrow \quad t_2, t_3 \leq 3$$

$$t_2 + t_5 = (40 - 18) - (39 - 25) = 8 \quad \Rightarrow \quad t_2, t_5 \leq 8$$

$$t_3 + t_4 = (37 - 27) - (29 - 23) = 4 \quad \Rightarrow \quad t_3, t_4 \leq 4$$

$$t_3 + t_6 = (56 - 27) - (45 - 23) = 7 \quad \Rightarrow \quad t_3, t_6 \leq 7$$

$$t_4 + t_5 = (40 - 29) - (39 - 37) = 9 \quad \Rightarrow \quad t_4, t_5 \leq 9$$

$$t_5 + 5 - 40) = 12 \quad \Rightarrow \quad t_5, t_6 \leq 12$$

Picking the tightest bound for each transmission time, we are left with:

$$t_1 \leq 9 \quad t_2 \leq 3 \quad t_3 \leq 3 \quad t_4 \leq 4 \quad t_5 \leq 8 \quad t_6 \leq 7$$

(6) (c) Based on your answer in part (b), 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?

Solution: Let o be the skew of B with respect to A. In other words, time(B) = time$(A) + o$. We can translate the inequalities above into inequalities in o. For example, message 1 was sent at time 10 according to B's clock, which is time $10 - o$ according to A's clock. So:

$$12 = 10 - o + t_1 \quad \text{or} \quad o = t_1 - 2$$

Combining this wth $0 \leq t_1 \leq 9$, we have

$$-2 \leq o \leq 7$$

Doing the same for the other messages we have:

$$18 + t_2 = 25 - o \Rightarrow o = 7 - t_2 \Rightarrow 4 \leq o \leq 7$$

$$27 - o + t_3 = 23 \Rightarrow o = t_3 - 4 \Rightarrow 4 \leq o \leq 7$$

$$29 + t_4 = 37 - o \Rightarrow o = 8 - t_4 \Rightarrow 4 \leq o \leq 8$$

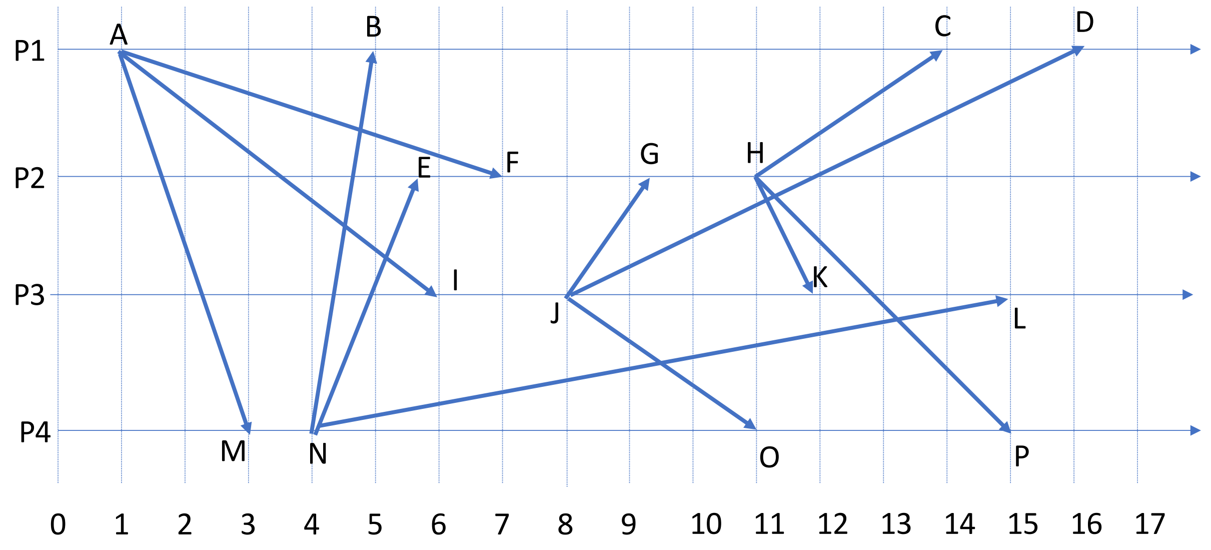


Figure 1: Event timeline for Questions 3 and 4

$$39 - o + t_5 = 40 \Rightarrow o = t_5 - 1 \Rightarrow -1 \leq o \leq 7$$

$$45 + t_6 = 56 - o \Rightarrow o = 11 - t_6 \Rightarrow 3 \leq o \leq 11$$

Again taking the tightest bounds out of all of these inequalities, we have:

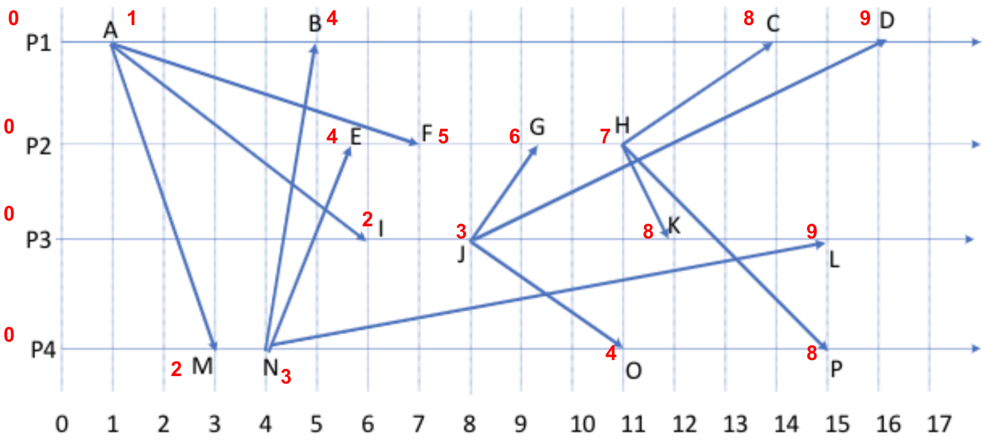
$$4 \leq o \leq 7$$

To minimize the skew we want to change B's clock *back* by 5.5 time units, leaving a skew of at most 1.5.

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.

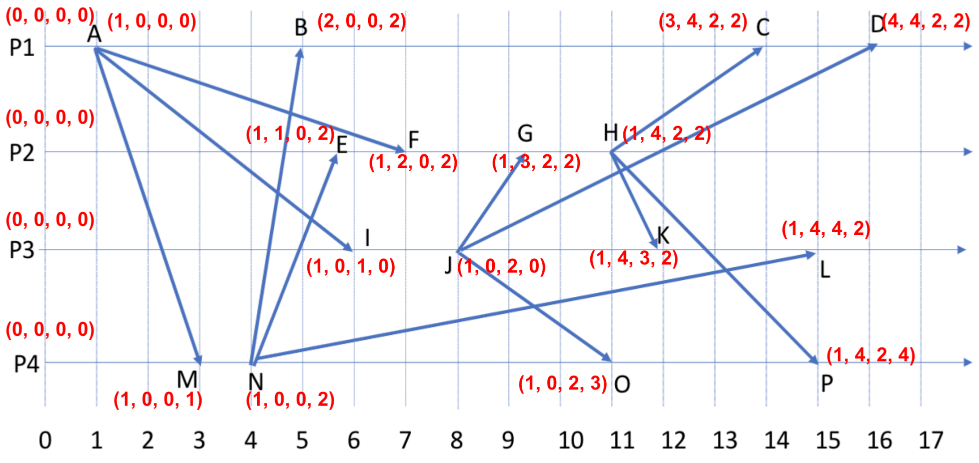
(6) (a) Write down the Lamport timestamp of each event

Solution:



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

Solution:



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

Solution:

(i) Events concurrent with event C: K, L, O, P

(ii) Events concurrent with event I: B, E, F, M, N

(iii) Events concurrent with event O: B, C, D, E, F, G, H, K, L

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.

Solution:

All possible consistent cuts the resulting snapshot could capture:

1. B, F, J, N
2. B, G, J, N
3. B, H, J, N

4. B, F, J, O
5. B, G, J, O
6. B, H, J, O
7. B, H, J, P
8. C, H, J, P
9. D, H, J, P

The frontier for P3 will always be J since the snapshot is initiated by J at time 9. Note that the markers from P3 will reach P2 after G and P4 after O. But it is possible for P1 to get the marker then send its own marker to P2 and/or P4 before events G and O occur. This scenario requires P1 to receive its marker before time C, however. If the marker is received after C then the frontier for P2 must be H and likewise the frontier for P4 must be P. The solution above are all consistent cuts that satisfy this property.