Homework 1

CS425/ECE428 Spring 2019

Due: Thursday, Feb 7 at 11:55 p.m.

1. (a) (3 points) Consider an asynchronous distributed system with unbounded message delay (though guaranteed message delivery), and perfectly synchronized clocks. p_i and p_j implement a heartbeat protocol, where p_j sends a heartbeat every T time units, and p_i declares p_j as failed when it does not hear a heartbeat for $T + \Delta$ time units. What is the worst-case detection time for p_j having failed? (*Hint:* it will depend on the number of heartbeats sent.)

   Solution: After n heartbeats, the worst-case detection time is $(n + 1)\Delta + T$.

   Suppose the first heartbeat is sent at time T; p_i will wait for this heartbeat until time $T + \Delta$. If it arrives at exactly this time, p_i will wait for the next heartbeat, which will be sent at time T, until time $2T + 2\Delta$. Therefore, the k-th heartbeat, sent at time kT, can arrive as late as $k(T + \Delta)$ without p_i declaring p_j as having failed. If a failure occurs right after the n-th heartbeat is sent at nT, p_i can receive that heartbea at $n(T + \Delta)$ and then wait until $(n + 1)(T + \Delta)$ before declaring p_j as failed. $(n + 1)(T + \Delta) - nT$ gives us the worst-case detection time.

   If we assume the first heartbeat is sent at time $t = 0$ this value is $(n + 1)\Delta + 2T$.

   (b) (2 points) Can you suggest a modification to the protocol to make the worst-case detection time bounded regardless of the number of heartbeats?

   Solution: The delays will be accumulated in the scenario of (a). Instead of resetting the wait period every time p_i hears from p_j, it should expect a heartbeat every T time units within a timeout of Δ. That is, p_i declares p_j as failed when it does not hear a heartbeat within kT and $kT + \Delta$ time units, when $k \in \mathbb{N}$. In this case, the worst-case detection time is $T + \Delta$.

   Other answers may be accepted as long as it does not make modifications on the problem setting or hurt the accuracy too much.

   (c) (4 points) Consider N processes using a ring ping-ack protocol. (I.e., p_i sends a ping to p_{i+1} who sends an acknowledgment back.) Pings are sent every T time units, and a timeout is set to Δ. Assume again we are in an asynchronous network, and let the probability that a round-trip time exceeds Δ be p.

   What is the probability that at least one alive process will be declared as having failed within a time period T? Calculate the value for $p = 0.01$ and $N = 100$.

   Solution: 0.6340.

   P[at least one alive processes declared as failed] $= 1 - P$[all alive processes are safe] $= 1 - (1 - 0.01)^{100} \approx 0.6340$.

   What if we declare failure only after k missed acks?

   Solution: $1 - (1 - (0.01)^k)^{100}$.

   The protocol makes false decisions for k times on the same process.

   (d) (2 points) Consider a ping-ack protocol in a *synchronous* network with a minimum one-way delay

of 10 ms and a maximum delay of 100 ms. Let $T = 1$ s and assume no processing delays and perfectly synchronized clocks.
What should your timeout value be? What is the maximum detection time?

Solution: 200ms. 1190ms.

The timeout should be the travel time of a complete ping-ack process in the worst case.

The maximum detection time happens when the pinged process fails immediately after it sends the last ack. It is as early as 10ms after the last ping message is sent. The detection time is $1000 - 10 + 200 = 1190$ms.

2. (a) (6 points) Consider a hierarchical NTP synchronization between four processes, A, B, C, and D as shown in Figure 1a. Each arrow is labeled with the *round-trip delay* betwen two processes. What is the bound on the clock skew between every pair of processes?

Solution: Given the accuracy of NTP prediction (with no drift): $o = o_i \pm \frac{d_i}{2}$ the bound on the clock skew between every pair of the nodes will be (in ms):

	A	B	C	D
A	-	25	25.5	26
B	-	-	0.5	1
C	-	-	-	1.5
D	-	-	-	-

Note $d_i = RTT$. Also note that we are summing up the skew between each level because we are measuring the bounds (i.e. worst case scenario).

(b) (4 points) Consider a maximum drift rate of 0.01%. How frequent should *each* synchronization be to ensure that no two processes have a skew larger than 50 ms?

Solution: A non-zero drift will modify the bound to:

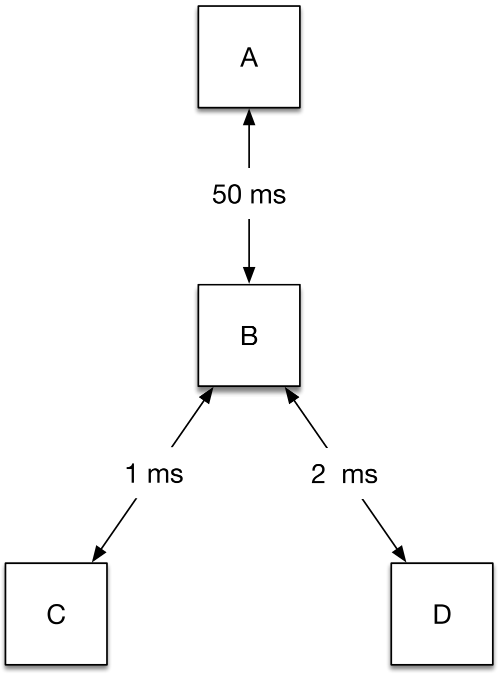
$$o = o_i \pm \big(\frac{d_i}{2} + \alpha\Delta T\big)$$

where α is the maximum drift rate and ΔT is the delay between each sync. For each pair, the min sync frequency will be:

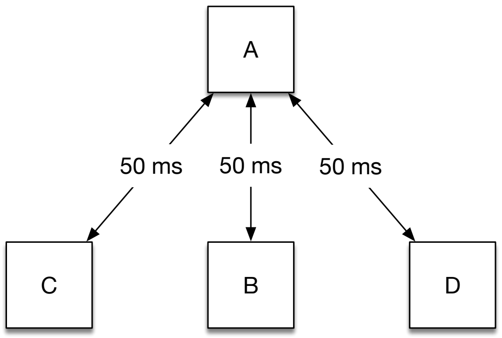
$$\frac{d_i}{2} + \alpha\Delta T \leq 50 \Rightarrow \alpha\Delta T \leq 50 - \frac{d_i}{2} \qquad (1)$$
$$\Rightarrow \Delta T \leq \frac{1}{\alpha}\big(50 - \frac{d_i}{2}\big)$$

Therefore, the maximum sync time between nodes should be:

$$\text{A} - \text{B} \leq \frac{1}{0.0001} * (50 - 25) = 250000ms$$
$$\text{B} - \text{C} \leq \frac{1}{0.0001} * (50 - 0.5) = 495000ms$$
$$\text{B} - \text{D} \leq \frac{1}{0.0001} * (50 - 1) = 490000ms$$



(a) NTP configuration for part a.



(b) NTP configuration for part c.

Figure 1: NTP configuration for Question 2

(c) (2 points) Suppose that there was a flat hierarchy instead where B,C, and D all synchronized with A, with an RTT of 50ms, as shown in Figure 1b. What would be the clock skew between every pair of processes?

Solution: Similar to part(a), the maximum clock skew between every pair of processes will be equal to RTT therefore it will be $25ms$ between $\textcircled{A}$ and every other node and $50ms$ for all the other pairs.

3. (a) (5 points) Looking at fig. 2, write down the Lamport timestamp of each event.

Solution: Check fig. 3

(b) (5 points) Using fig. 2, write down the vector timestamp for each event

Solution: Check fig. 4

(c) (5 points) List all the concurrent events

Solution: Concurrent Events are: $A \parallel \{E, F, G, J, K, M, N, O\}$
$B \parallel \{F, G, J, K, M, N, O\}$
$C \parallel \{F, G, J, K\}$
$D \parallel \{F, G, J, K\}$
$E \parallel \{A, I, M\}$
$F \parallel \{A, B, C, D, J, K, L, M, N, O, P\}$
$G \parallel \{A, B, C, D, J, K, L, P\}$

$H \parallel \{J, K, L, P\}$
$J \parallel \{A, B, C, D, F, G, H, P\}$
$K \parallel \{A, B, C, D, F, G, H, P\}$
$L \parallel \{F, G, H, P\}$
$M \parallel \{A, B, E, F\}$
$N \parallel \{A, B, F\}$
$O \parallel \{A, B, F\}$
$P \parallel \{F, G, H, J, K, L\}$

4. (a) (4 points) Looking at fig. 2, suppose that P1 initiates the Chandy-Lamport snapshot algorithm at time 9. Write down *all* possible consistent cuts that the resulting snapshot could capture. (You can describe each cut by its frontier events.)

Solution: Since P1 initiates the snapshot at time 9, it writes down B as the latest event.

P2 must receive its marker from P1 before H due to FIFO channels. It may receive its *first* marker from P1 (or another process) before or after G, so the frontier would include F or G.

Likewise, P1's marker message to P3 must arrive before L, but P3 could receive its first marker before J, between J and K, or between K and L.

Finally P4 must receive its first marker before P but after O.

So the set of possible frontiers is $B \times \{F, G\} \times \{I, J, K\} \times O$.

(b) (4 points) Pick an example of a distributed system; feel free to use some of the ones previously mentioned in class. Give one example of each of the following in that system:

- A stable property
- An unstable property
- A liveness property
- A safety property

You may use the same system for all properties or several different ones.

Solution: We offer a few examples here; obviously there are many possible correct answers.

Stable property: this video has been viewed at least 100 times; this video has finished uploading; all correct processes have delivered message m.

Unstable property: this video has been viewed *exactly* 100 times; this account has 100 friends; some process has not yet delivered m

Liveness property: all of this account's friends will eventually see a post; a lock will be released

Safety property: a restricted video will only be shown to people who are allowed; a video will never have a negative number of views; a message will be delivered at most once

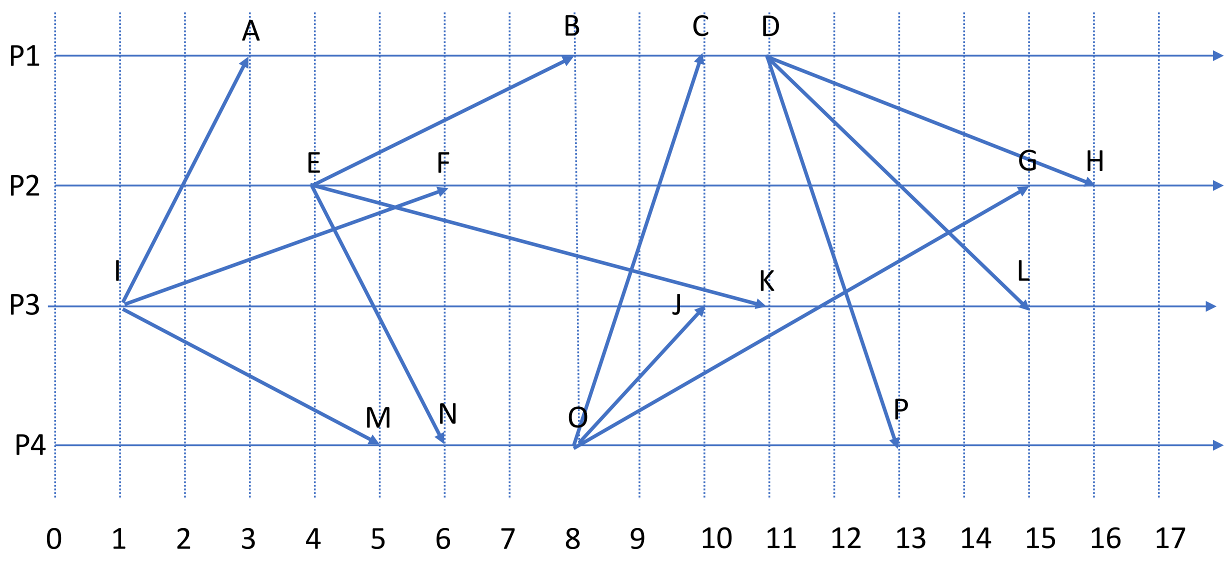


Figure 2: Timeline for questions 3 and 4.

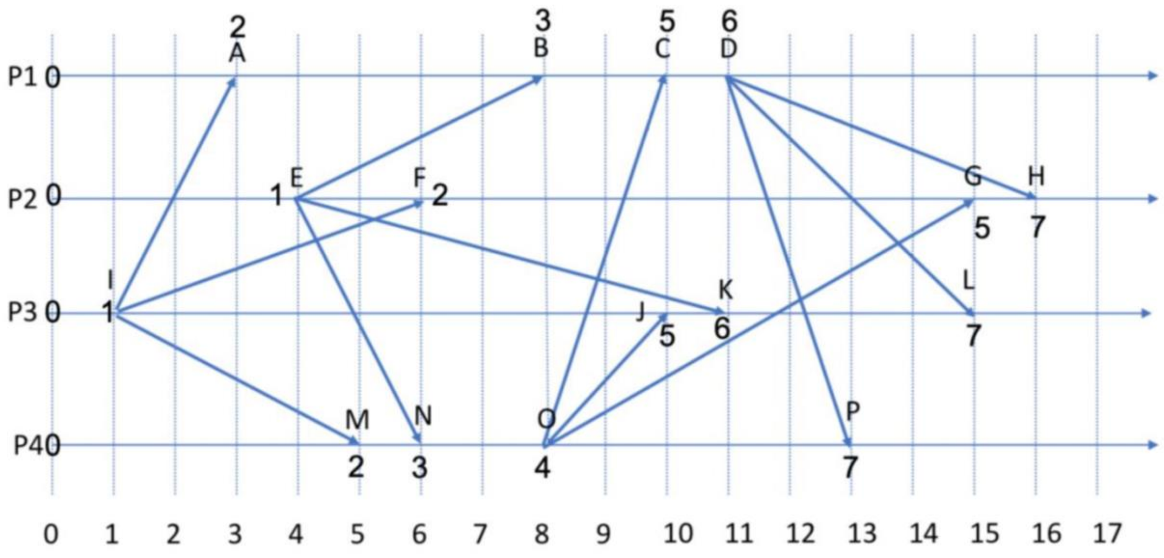


Figure 3: Lamport Timestamp for questions 3.

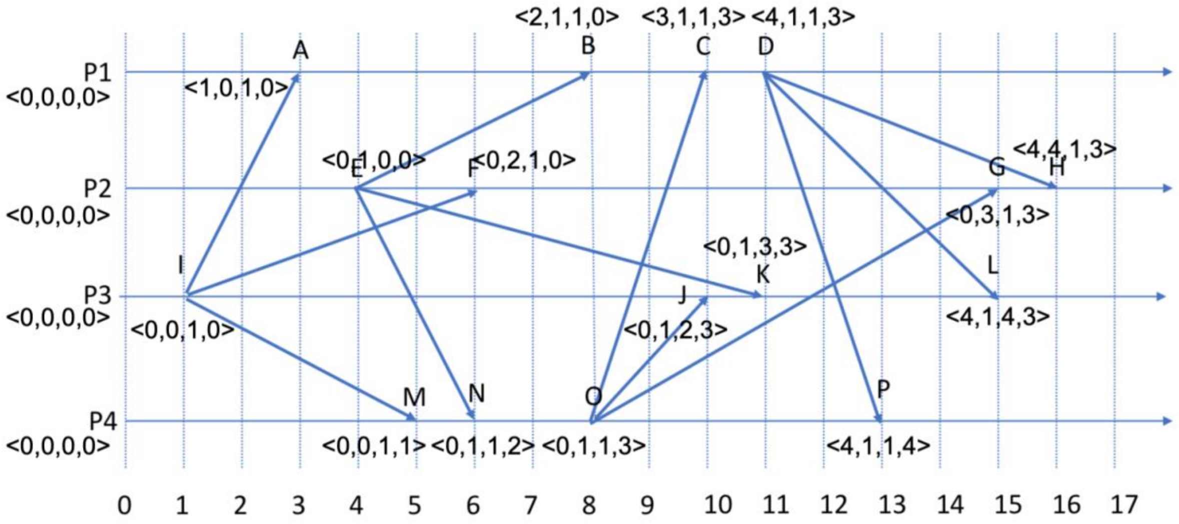


Figure 4: Vector Clock for questions 3.