CS 425/ECE 428 Midterm 1

March 8, 2021

- This is a *closed-book* exam. You are **not** allowed consult your textbook, lecture notes and slides, or any materials on your electronic devices. You can refer to one double-sided physical formula/cheat sheet (normal sized) that can be either hand-written or printed.
- This work should be entirely your own, and you are **not** allowed to collaborate or consult with anyone.
- This exam has 7 pages and 5 questions, worth a total of 70 points.
- Please write your answers in a separate document. Your answers can be either typed or *clearly* handwritten and scanned.
- You **cannot** use an online text editor (e.g. Google Docs) for typing your answers. You are only allowed to use offline text editors, such as Microsoft Word, textEdit, vim, notepad, etc to type your answers. Note that *the offline text editor is the only additional software you are allowed* to use during this exam.
- You **cannot** use an iPad (or any other tablet/device) to digitally ink your answers.
- Scratch paper must be loose-leaf blank sheets (notebooks are not allowed).
- There is no way for us to provide technical clarifications on questions with CBTF proctoring. We have tested the questions thoroughly on our end to ensure they are clear and unambiguous. However, if you are confused about any question, clearly state your assumptions and proceed with answering the question based on those assumptions. If we find that the question was truly ambiguous, and your assumptions are reasonable given the wording of the question, we will give you full credit for a correct answer under those assumptions.
- Once you are finished, please upload your answers into Gradescope. You must upload your answers within 1 hour 50 mins of the start of your exam (unless you have an approved time extension).
- After you have submitted, please refrain from discussing the exam questions and solutions with anyone until solutions are posted.

1. Multiple Choice . . . . *36 points*

   For questions with checkboxes (○ [i] Foo ○ [ii] Bar), select all that apply and write down all the appropriate choice numbers in your answer document. For questions with multiple choices (A. Foo B. Bar), choose the best option and write just the corresponding letter in your answer document.

   (a) (1 point) Assuming the same period, which failure-detection protocol uses more bandwidth?

   A. Ping-Ack
   B. Heartbeat
   C. Both use the same bandwidth

   (b) (1 point) Assuming the same period and negligible processing delays, which failure-detection protocol has higher worst-case failure detection time in a synchronous system? (Timeout values are properly selected using known minimum and maximum network delays.)

   A. Ping-Ack
   B. Heartbeat
   C. Both have the same worst-case failure detection time

   (c) (3 points) A ping-ack failure detector is complete in

   ○ [i] A synchronous system, with properly selected timeout value
   ○ [ii] A synchronous system, regardless of the timeout value
   ○ [iii] An asynchronous system

   (d) (1 point) True or false: It is always possible to perfectly synchronize two clocks (achieving a relative skew of zero after synchronization) in any asynchronous system.

   A. True
   B. False

   (e) (1 point) True or false: It is always possible to perfectly synchronize two clocks (achieving a relative skew of zero after synchronization) in any synchronous system.

   A. True
   B. False

   (f) (3 points) Suppose that e_i and e_j are two events on two distinct processes, P_i and P_j. Let $L(e_i)$ and $L(e_j)$ be Lamport timestamps of e_i and e_j respectively, and $V(e_i)$ and $V(e_j)$ be vector timestamps of e_i and e_j respectively. Which of the following *could be* true?

   ○ [i] $L(e_i) = L(e_j)$
   ○ [ii] $V(e_i) = V(e_j)$
   ○ [iii] $L(e_i) = L(e_j)$ and $V(e_i) < V(e_j)$.

   (g) (4 points) If event A has a Lamport timestamp of 3 and B has a Lamport timestamp of 6, which of the following *could be* true?

   ○ [i] $A \rightarrow B$
   ○ [ii] A is concurrent with B
   ○ [iii] $B \rightarrow A$
   ○ [iv] A and B are *consecutive* events on the same process

   (h) (4 points) If event A has a vector timestamp (3,2,4) and event B has a vector timestamp (5,2,6), which of the following *could be* true?

○ [i] $A \rightarrow B$

○ [ii] A is concurrent with B

○ [iii] $B \rightarrow A$

○ [iv] A and B are events on the same process

(i) (3 points) Consider three events and their vector timestamps: $e_1 : (5, 4, 2)$, $e_2 : (6, 5, 5)$, $e_3 : (5, 5, 7)$.

○ [i] A consistent cut that contains e_2 must contain e_1.

○ [ii] A consistent cut that contains e_2 must contain e_3.

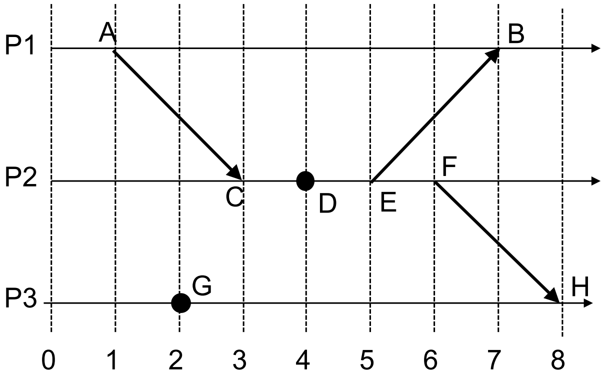
○ [iii] A consistent cut that contains e_3 must contain e_1.

(j) (1 point) True or false: a snapshot recorded by Chandy-Lamport algorithm always represents the state of all processes and channels at some point in (physical) time

A. True

B. False

(k) (3 points) Consider the following timeline with eight events. Dots represent local events at a process. Arrow beginnings represent send events, and arrow endings represent the corresponding receive events. The numbers indicate real time.



Which of the following are valid linearizations of the events in the timeline:

○ [i] A, B, C, D, E, G, F, H

○ [ii] A, C, D, E, F, B, G, H

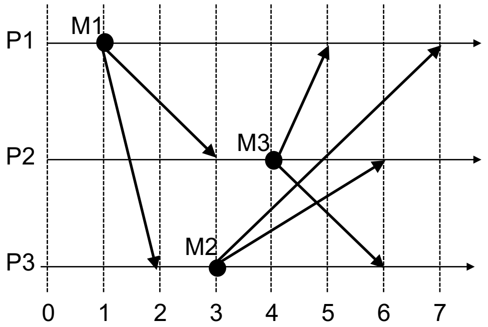
○ [iii] G, A, C, D, E, B, F, H

(l) (2 points) A global liveness property is false for state S_i and true for *all* states reachable from S_i (along all possible linearizations). If the state S_i is always reachable from the initial system state S_0 along all possible linearizations, which of the following is *certainly* true:

○ [i] the system satisfies liveness.

○ [ii] the property is stable.

(m) (3 points) The execution below shows three multicasts among a group of three processes: M1 from P1, M2 from P3, and M3 from P2.



Assume that the multicast messages are self-delivered at the sending process when the multicast is issued (indicated by arrow beginnings), and are delivered at other processes as soon as they arrive (indicated by the arrow endings). The execution satisfies which of the following orderings:

○ [i] FIFO-ordered

○ [ii] Total-ordered

○ [iii] Causal-ordered

(n) (3 points) A system of **two** processes uses basic multicast protocol (B-multicast) for group communication. (Recall that in B-multicast, message delivery happens immediately when the protocol receives the message, and self-delivery happens immediately when the multicast message is sent.) If each unidirectional communication channel between the processes follows FIFO order and no messages are dropped, then the resulting multicast is always:

○ [i] FIFO-ordered

○ [ii] Total-ordered

○ [iii] Causal-ordered

(o) (3 points) A system of **three** processes uses basic multicast protocol (B-multicast) for group communication. (Recall that in B-multicast, message delivery happens immediately when the protocol receives the message, and self-delivery happens immediately when the multicast message is sent.) If each unidirectional communication channel between the processes follows FIFO order and no messages are dropped, then the resulting multicast is always:

○ [i] FIFO-ordered

○ [ii] Total-ordered

○ [iii] Causal-ordered

2. Clock Synchronization . . . . *6 points*

   (a) (3 points) A client attempts to synchronize its clock using Cristian's algorithm. It sends requests to three different servers simultaneously, and records the round-trip time, and timestamp returned by each server, as shown in the table below.

Server	Round-trip Time	Returned Timestamp (hour:min:sec)
A	88ms	19:35:23.648
B	110ms	19:35:23.312
C	64ms	19:35:23.475

   Assume that the minimum delay between the client and each server equals 20 ms, and the client is aware of it.

   (i) To minimize the worst-case skew, which server should the client synchronize with?

   (ii) What time should it set its software clock to upon receiving the corresponding server's response?

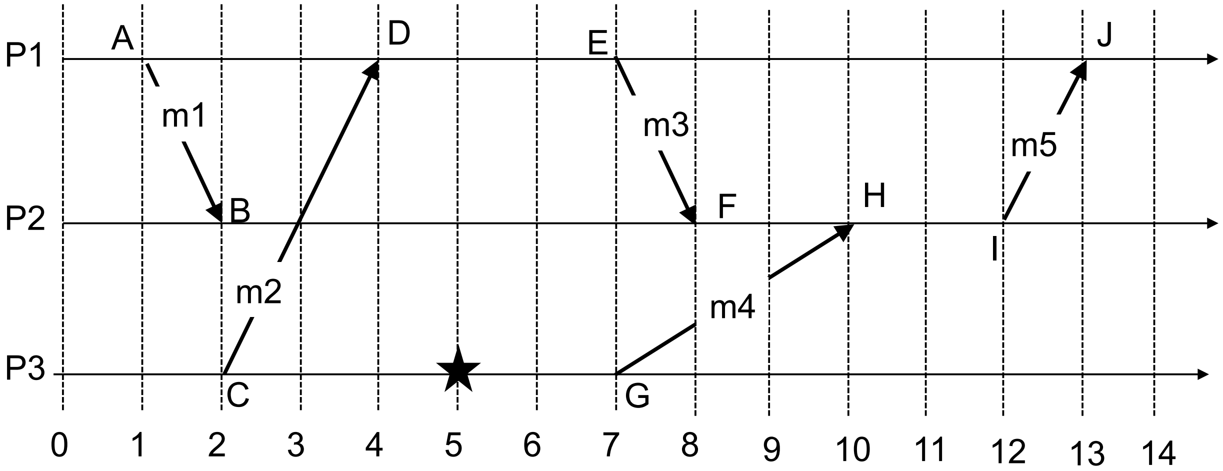
   (iii) What is the corresponding worst-case skew, as estimated by the client right after synchronization?

   (b) (3 points) Consider two NTP peers synchronizing their clocks. Server B receives server A's message M at local time 07:26:10, with the message bearing A's send timestamp 07:26:24. B sends a reply to it by message M' at local time 07:26:13. Server A receives the message M' at local time 07:26:37, with M' bearing B's timestamps when B received M and sent M'.

   (i) Estimate the offset for server A with respect to server B.

   (ii) What is the estimated synchronization bound for the computed offset right after synchronization?

3. Global Snapshots . . . . *12 points*



Consider the timeline above. The arrow beginnings and endings represent send and receive events for five messages in a system of three processes. The numbers below indicate real time. P3 starts the Chandy-Lamport snapshot algorithm at time 5 (marked by a star). Assume all requirements for Chandy-Lamport algorithm hold, i.e. each unidirectional communication channel between the processes follows FIFO order, no messages are dropped, and no process fails. Answer the following questions in reference to this (select all choices that apply).

(a) (4 points) Which of the following are possible frontier events after which P1 records its state?

○ [i] A
○ [ii] D
○ [iii] E
○ [iv] J

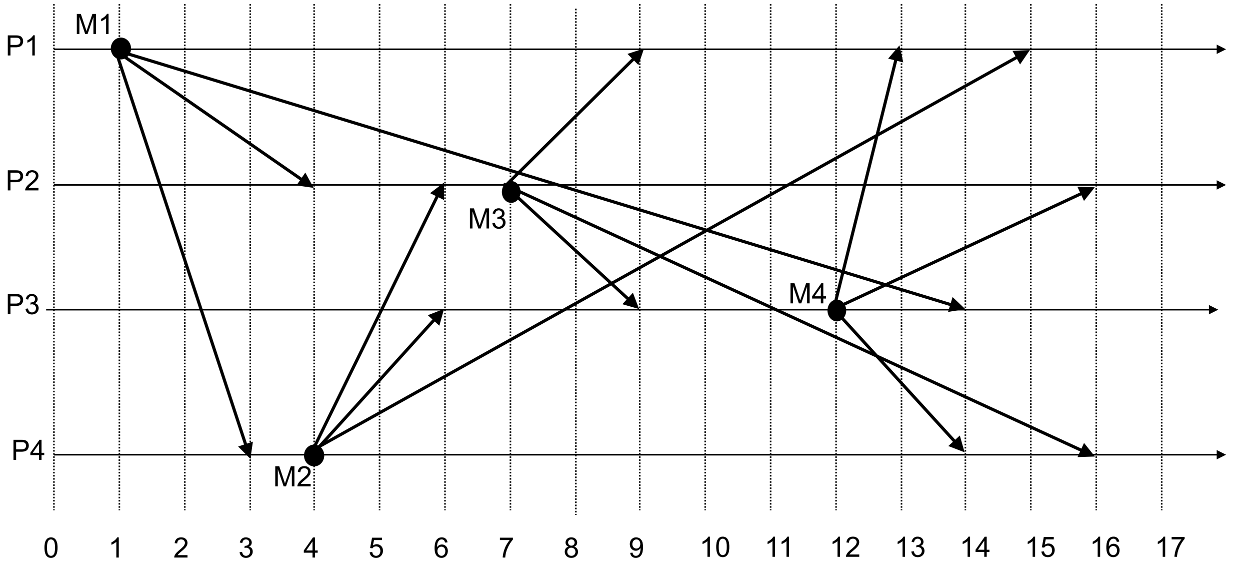
(b) (4 points) Which of the following are the possible frontier events after which P2 records its state?

○ [i] B
○ [ii] F
○ [iii] H
○ [iv] I

(c) (4 points) What could be the recorded state of the $P1 \rightarrow P2$ channel?

○ [i] Empty
○ [ii] $m1$
○ [iii] $m3$
○ [iv] $m1, m3$

4. Causal-ordered multicast .... *8 points*



The execution above shows fours multicasts among a group of four processes: M1 from P1, M2 from P4, M3 from P2, and M4 from P3. The numbers below represent real time.

We want these multicasts to be **causally ordered**. The corresponding protocol implementation immediately self-delivers a message at the sending process when the multicast is issued (indicated by arrow beginnings). The implementation behaves in the following manner when a message m is received by the protocol at another process (indicated by arrow endings):

- if all ordering requirements are satisfied, the message m is delivered immediately.
- if not, the message m is buffered till all other required (pending) messages are delivered. Once all pending messages are delivered (say, at real time unit t), this message m will be delivered immediately (at the same time unit t).

Fill in the table to show the (real) time at which each multicast is delivered at each of the four processes. Self-delivery times have been *correctly* pre-filled. The blank entries that you need to fill in have been labeled from (a) to (l) for ease of reporting in your answer document.

	M1	**M2**	**M3**	**M4**
P1	**1**	(a)	(b)	(c)
P2	(d)	(e)	**7**	(f)
P3	(g)	(h)	(i)	**12**
P4	(j)	**4**	(k)	(l)

5. Total-ordered multicast . . . . *8 points*

   (a) (5 points) In an execution of ISIS algorithm, a process has the following messages in its queue, with the corresponding priorities:

   - Message A: final priority 2
   - Message B: final priority 4
   - Message C: proposed priority 5
   - Message D: final priority 8
   - Message E: proposed priority 10

   In the final total ordering of the messages above, which of the following *could be* true (select all that apply by writing the appropriate choice numbers):

   ○ [i] C is delivered before B

   ○ [ii] C is delivered before D

   ○ [iii] D is delivered before C

   ○ [iv] E is delivered before C

   ○ [v] E is delivered before D

   (b) (3 points) Consider a modification of ISIS algorithm where, instead of waiting for all processes, the sender waits to receive proposed priorities only from a *majority* of processes (which may include the sender itself). It then picks the highest proposed priority among those received, and sends it to all processes as the final priority (including itself). As in the original ISIS algorithm, a message is marked as deliverable once its final priority has been received. Each process maintains priority queues as in the original ISIS algorithm – messages are sorted in the ascending order of their proposed or final priorities (whichever is applicable), and a message at the head of the queue is delivered if it has been marked as deliverable. Will this modified ISIS algorithm satisfy total-ordering? If yes, prove why. If not, describe a counter example (you may do so in words).