Multiple choice questions

1. [0.5 point] Select all methods (of socket class from Python's socket module) that block the interpreter by default. For incorrect selection, some point will be deducted.

   a. accept()
   b. listen()
   c. bind()
   d. recvfrom()

2. [0.5 point] Choose all correct statements about physical clocks. For incorrect selection, some point will be deducted.

   a. Time-of-day clock always moves forward
   b. Monotonic clocks may move backwards
   c. Better to use time-of-day clock to measure elapsed time on a single node
   d. Better to use monotonic clock to measure elapsed time on a single node

3. [0.5 point] Choose all correct statements for logical clocks. For incorrect selection, some point will be deducted.

   a. Given two Lamport timestamps such that L(a) < L(b), then a → b is always TRUE
   b. Given two Vector timestamps such that V(a) < V(b), then a → b is always TRUE
   c. Lamport clocks implement partial order
   d. Given two different vector timestamps we can surely say whether they are causal or concurrent

4. [0.5 point] Choose all correct statements about multicast ordering. For incorrect selection, some point will be deducted.

   a. Causal ordering ensures FIFO ordering
   b. FIFO ordering ensures Causal ordering
   c. Total ordering ensures Causal ordering
   d. FIFO ordering ensures Total ordering

Open questions

5. **[1.5 points] Explain the FIFO, Causal, and Total ordered multicast; with up to three sentences for each of them.**

   a. FIFO multicast
   - If m1 and m2 are multicast messages by the same node, and m1 is multicasted before m2, i.e., multicast(m1) → multicast(m2), then m1 must be delivered before m2, at all receivers
   - Multicasts by different nodes can be delivered at different order

   b. Causal multicast
   - Multicast messages that are causally related must be delivered in the same causality order, at all receivers
     - If multicast(G, m1) → multicast(G, m2)
     - then every process that delivers m2 will have delivered m1
   - Concurrent multicast messages can be delivered in either order

   c. Total-ordered multicast
   - All multicast messages are delivered at all group members in the same order
   - Unlike FIFO and Causal, this does not pay attention to order of multicast sending

6. **[2 points] Describe how following algorithms can be implemented.**

   a. **[1 point] FIFO multicast. Explain the multicast sender and receiver operations.**

Key idea: put sequence number while sending messages

Each sender process

- embeds the sequence number (SN) into message
- increments the SN after sending the message
- may need a buffer to keep the copy of un-acknowledged messages

Each receiving process keeps track of SNs of all senders. When a message is received

- Reply with ACK message to inform about the reception
- If message SN is:
  - as expected (next sequence), accept
  - higher than expected, buffer in a queue
  - lower than expected, reject

TCP implements FIFO ordering for unicast!

   b. **[1 point] Total-ordered multicast using a Sequencer. Explain the multicast sender and receiver operations**

Sending multicast at process Pi

- Instead of multicasting the message to everyone in group, the sender unicasts the message M to a special node called Sequencer

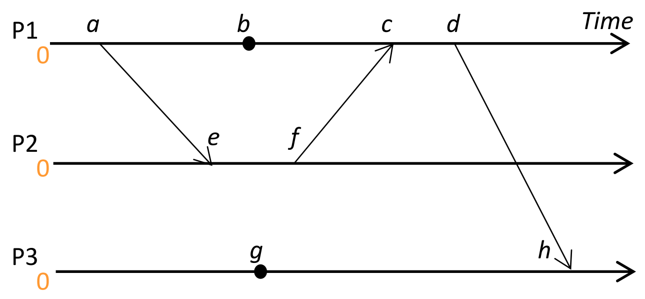
Sequencer

- A special node/process
- Maintains a global sequence number S (initially 0)
- When it receives a multicast message M, it sets S = S + 1, and multicasts <M, S> to all processes in group

Receive multicast at process Pi

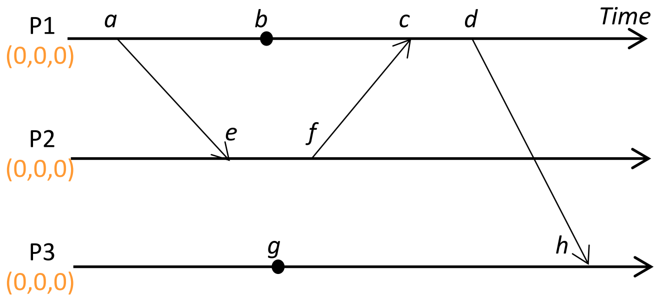
- Pi maintains a local received global sequence number Si (initially 0)
- If Pi receives a multicast M from Pj, it buffers it until it both
- Pi receives <M, S(M)> from sequencer, and
- Si + 1 = S(M)
- Then deliver it to application and set Si = Si + 1

7. [2.5 points] Problems on logical clock
   a. [1 point] Put correct Lamport timestamps for following diagram



Event	Timestamp
a	1
b	2
c	4
d	5
e	2
f	3
g	1
h	6

   b. [1.5 points] Put correct Vector timestamps for following diagram



Event	Timestamp
a	1.0.0
b	2.0.0
c	3.2.0
d	4.2.0
e	1.1.0
f	1.2.0
g	0.0.1
h	4.2.2

8. [2 points] Problems on Chord protocol (2 points total)
   a. [1 point] Given the chord overlay with m=5 and nodes [2, 16, 24, 26, 31], build the finger table for each of the nodes for s=m.

Node id	**Node's finger table (**list of the nodes in finger table**)**
2	16, 24
16	24, 2
24	26, 31, 2, 16
26	31, 2, 16
31	2, 16

   b. [1 point] Given the chord overlay and the finger tables (as above), find how the following lookup requests are resolved.
   Example: Lookup(A) at node B? *Answer:* node B → node C → node D. Node D returns the address of node E (which is responsible for A)

      i. Lookup(25) at node 2: 2→24, 24 returns addr of 25

      ii. Lookup(0) at node 16: 16→24→31, 31 returns addr of 2