Problem Set 2: Time and Ordering, Snapshots

1. [5 pts] A clock is reading 12:33:57.0 (hr:min:sec) when it is discovered to be 3 seconds fast. Explain why it is undesirable to set it back to the right time at that point and show (numerically) how it should be adjusted so as to be correct after 6 seconds has elapsed.

2. [5 pts] A client attempts to synchronize with a time server. It records the round-trip times and timestamps returned by the server in the table below. Which of the times in the table below should it use to set its clock and why? What should it set its clock to? What is the estimated accuracy w.r.t to server's clock? If minimum time to send or receive a message from the server is know to be at least 8ms, does your answer change? Why?

Round-trip Time (ms)	Server Clock (hh:mm:ss)
24	12:43:33.456
28	12:43:35.235
22	12:43:36.124

3. [3 pts] If in problem 2 if it is required to sync to within 1ms of the server, is that feasible? Explain your answer.

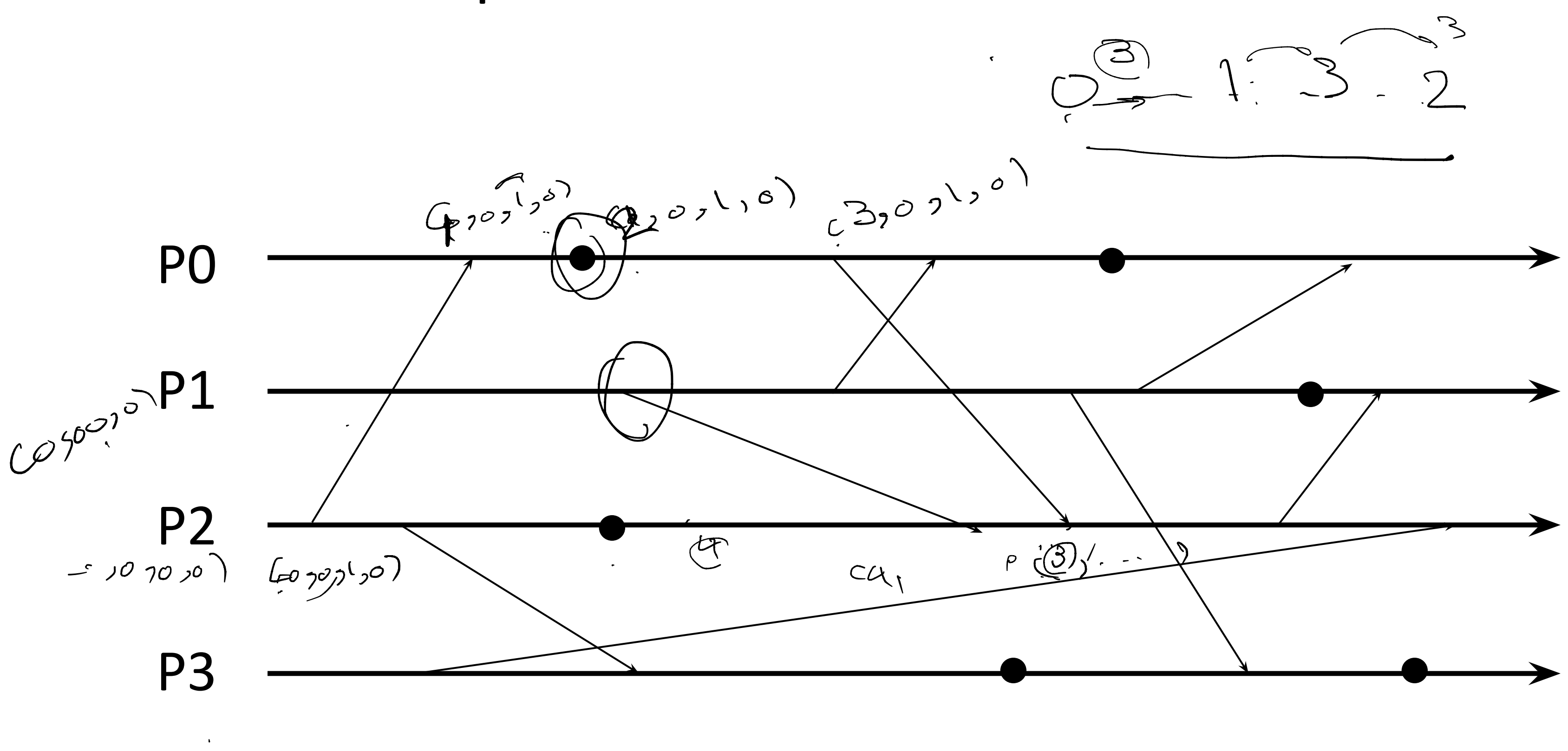
4. [5 pts] By considering a chain of zero or more messages connecting events e and e', and using induction, show that $e \rightarrow e'$ implies $L(e) <= L(e')$, where L(e) is the Lamport timestamp of event e.

5. [5 pts] Show that $V_j[i] <= V_i[i]$ where V_i is vector clock at process i.

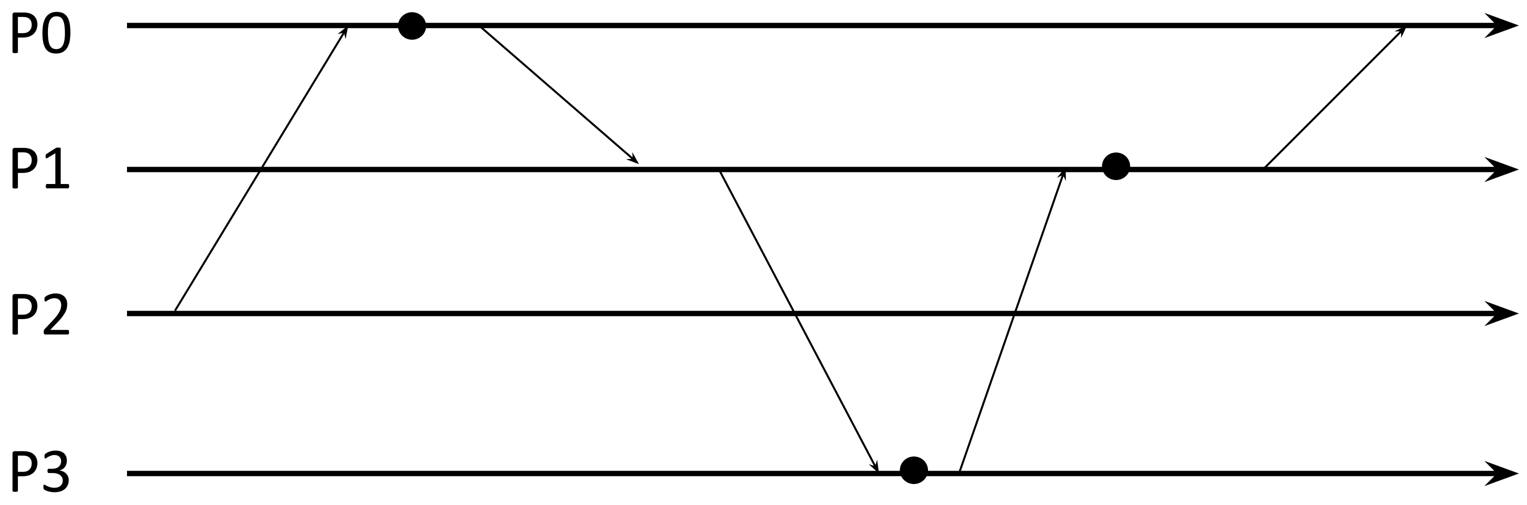
6. [5 pts] In a similar fashion to 4, show that $e \rightarrow e'$ implies $V(e) <= V(e')$

7. [5 pts] Using result from 5, show that if e and e' are concurrent that neither $V(e) <= V(e')$ nor $V(e') <= V(e)$. Thus show that if $V(e) < V(e')$ then $e \rightarrow e'$.

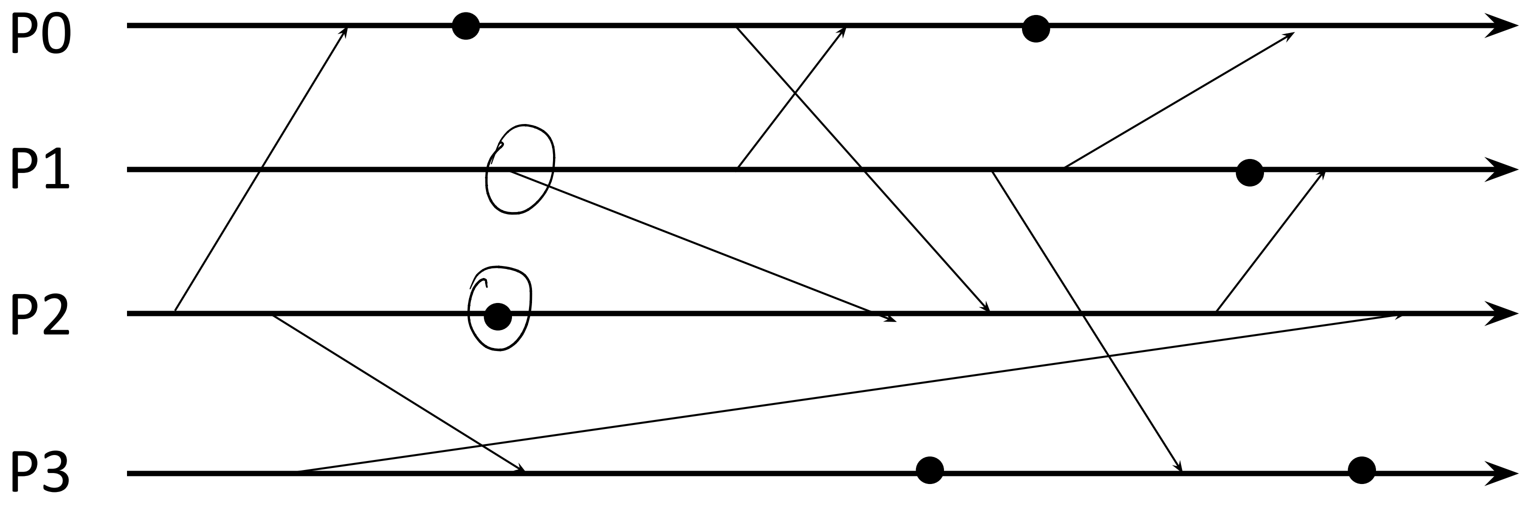
I. Mark one pair of events that are concurrent with each other.



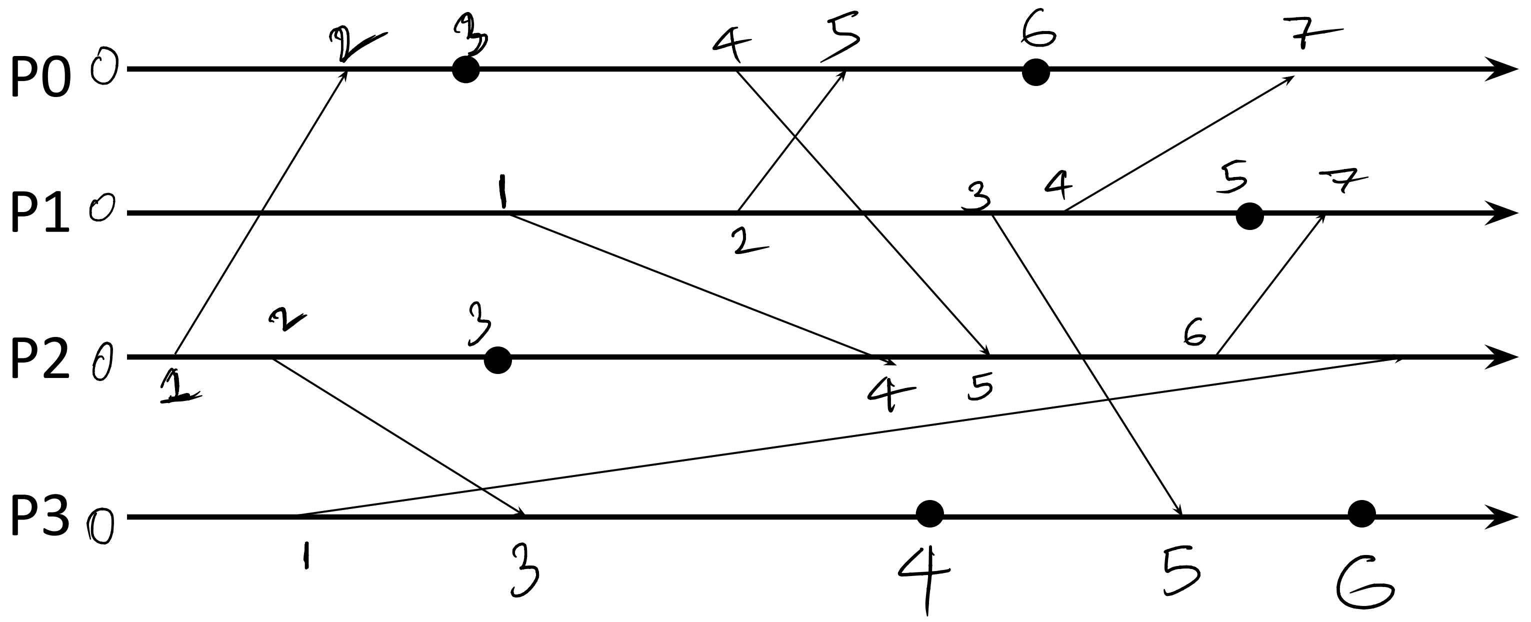
II. Mark one pair of events that are concurrent with each other.



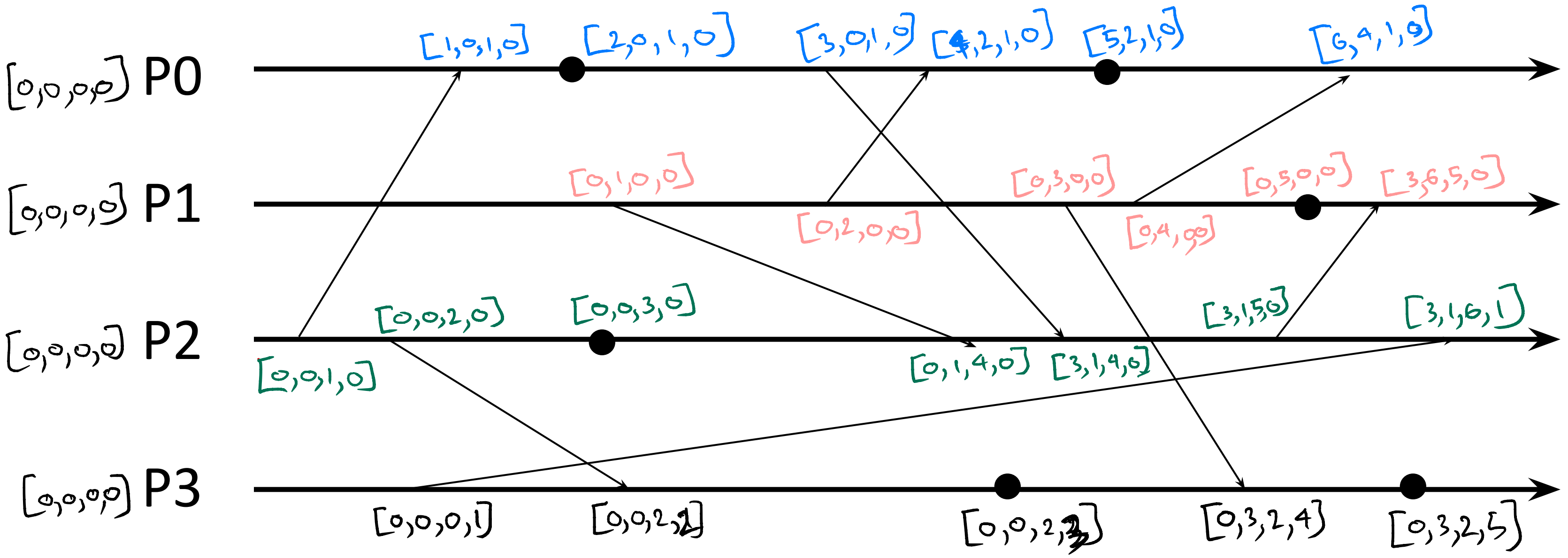
iII. Identify an event at process P2 that happens before an event at P1.



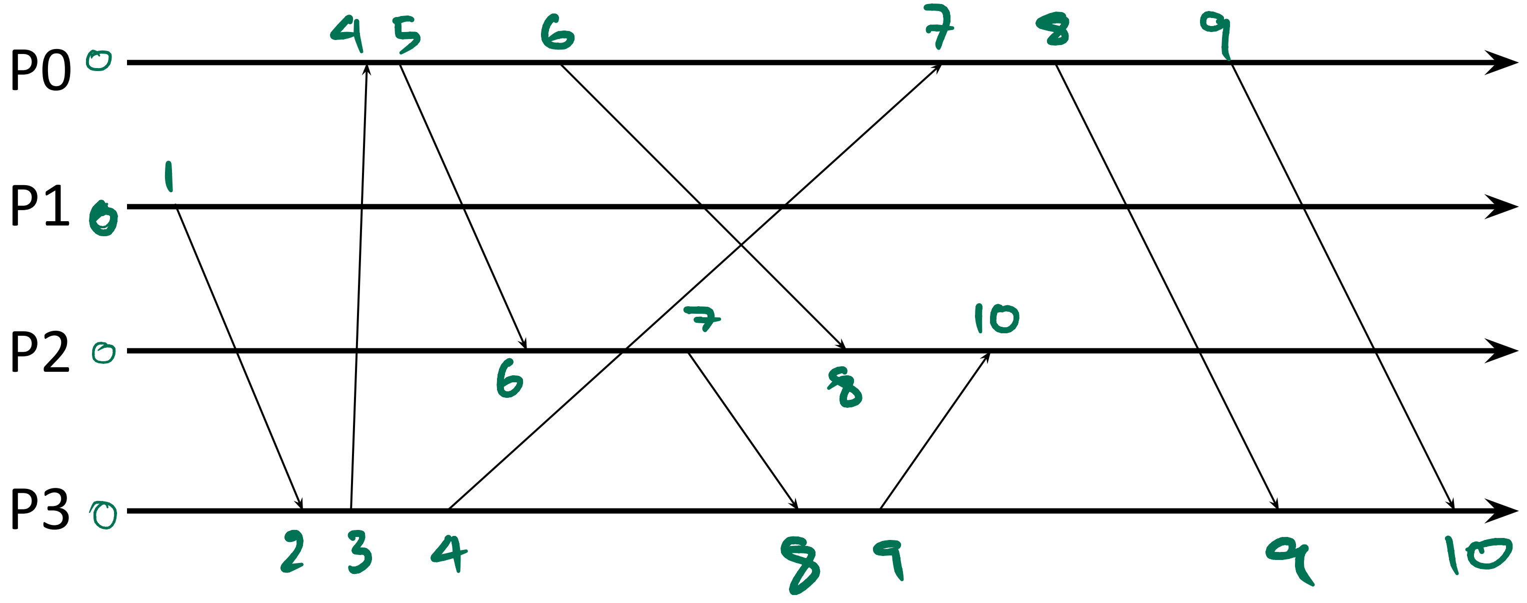
IV. Mark all Lamport timestamps on this figure for all events.
All processes start with zero timestamps.



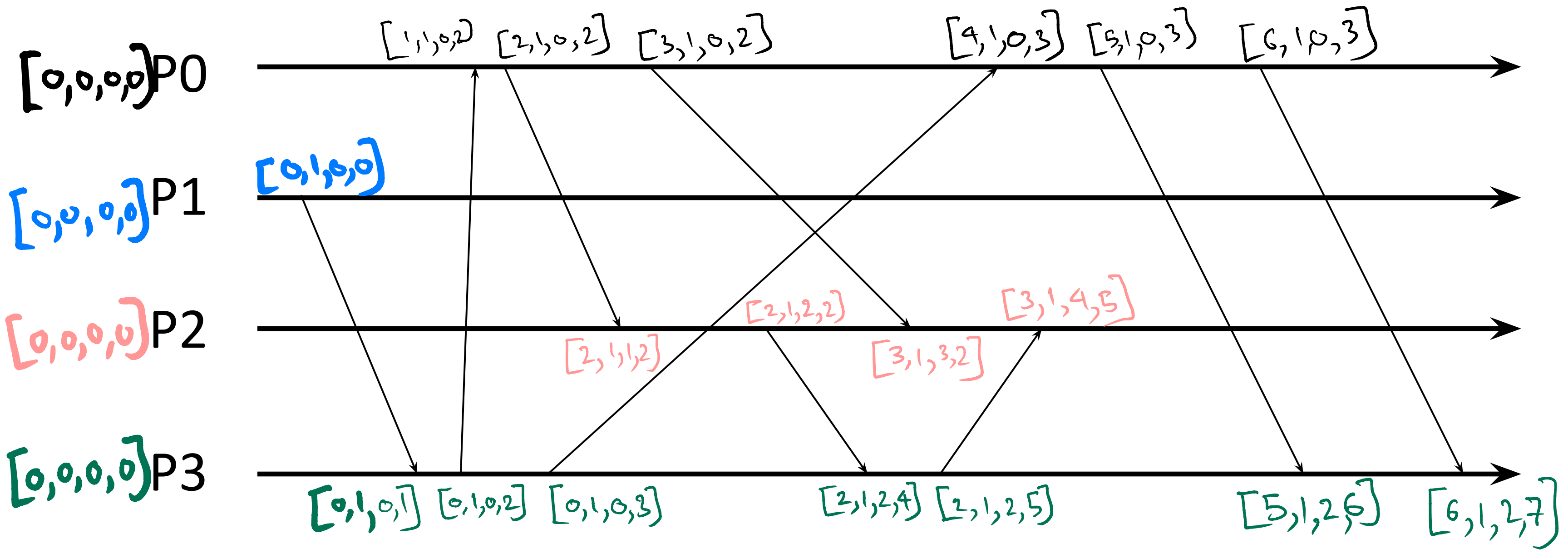
V. Mark all vector timestamps on this figure for all events.
All processes start with zero timestamps.



VI. Mark all Lamport timestamps on this figure for all events. All processes start with zero timestamps.

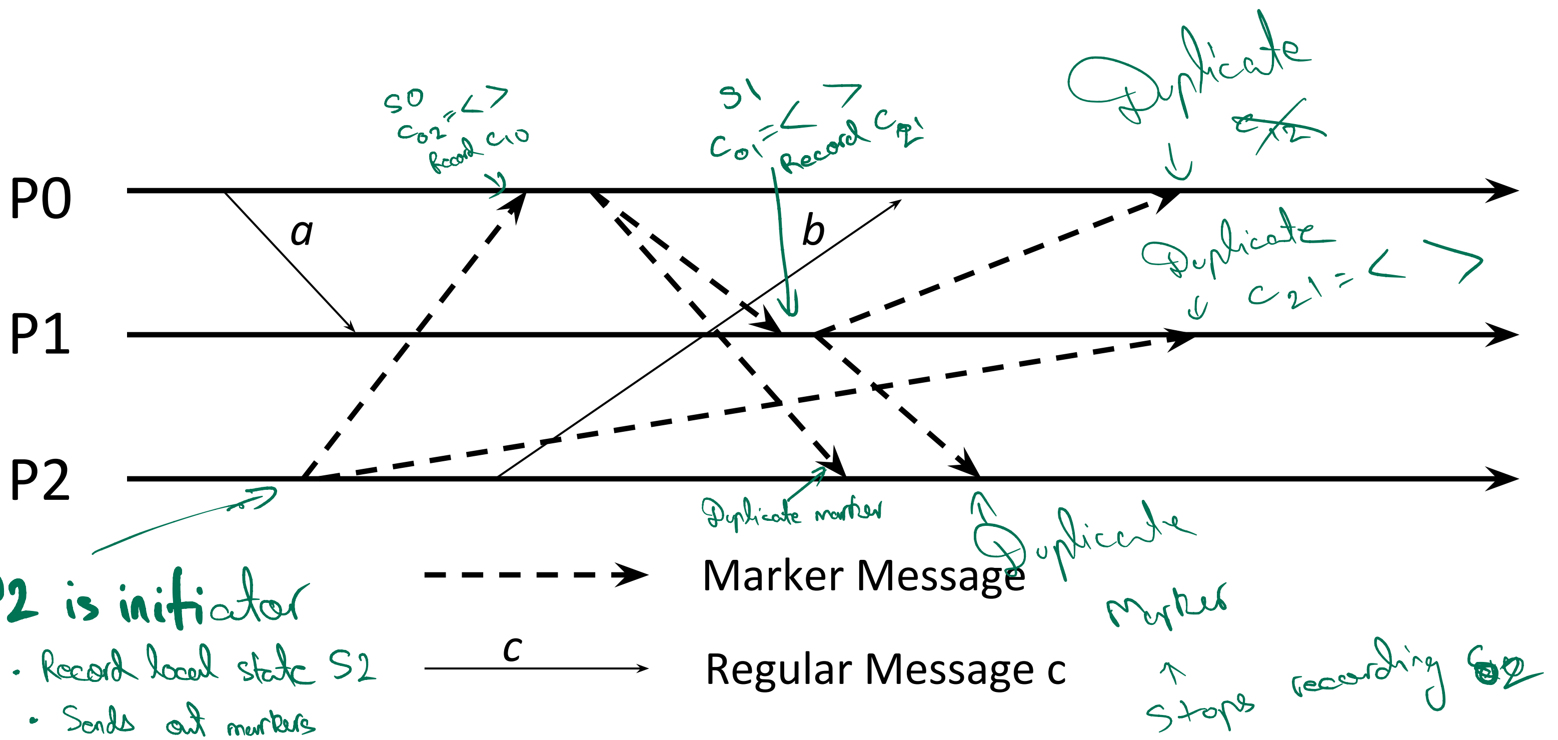


VII. Mark all vector timestamps on this figure for all events. All processes start with zero timestamps.



I. Chandy-Lamport Global Snapshots Algorithm

- Mark the entire global snapshot collected.



P2 is initiator

- Record local state S2
- Sends out markers
- Turn on recording on channels C_{02}, C_{12}

$S1 = C_{01} = < >$ $S2: C_{12} = < >$

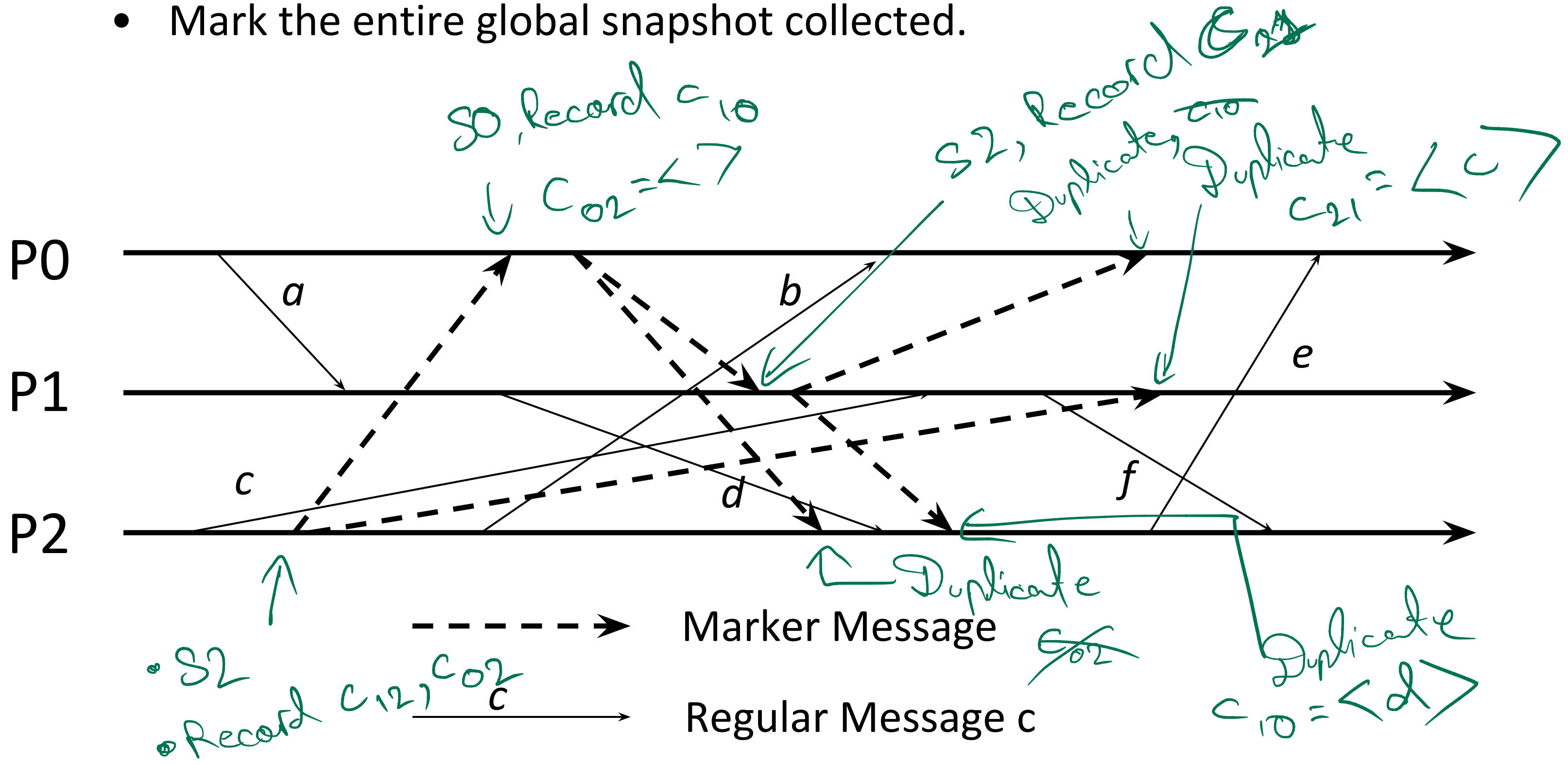
$S0 = C_{10} = < >$ $S1 = C_{21} = < >$

$S1 = C_{01} = < >$

$S_{=1} C_{21} = < >$

II. Chandy-Lamport Global Snapshots Algorithm

- Mark the entire global snapshot collected.

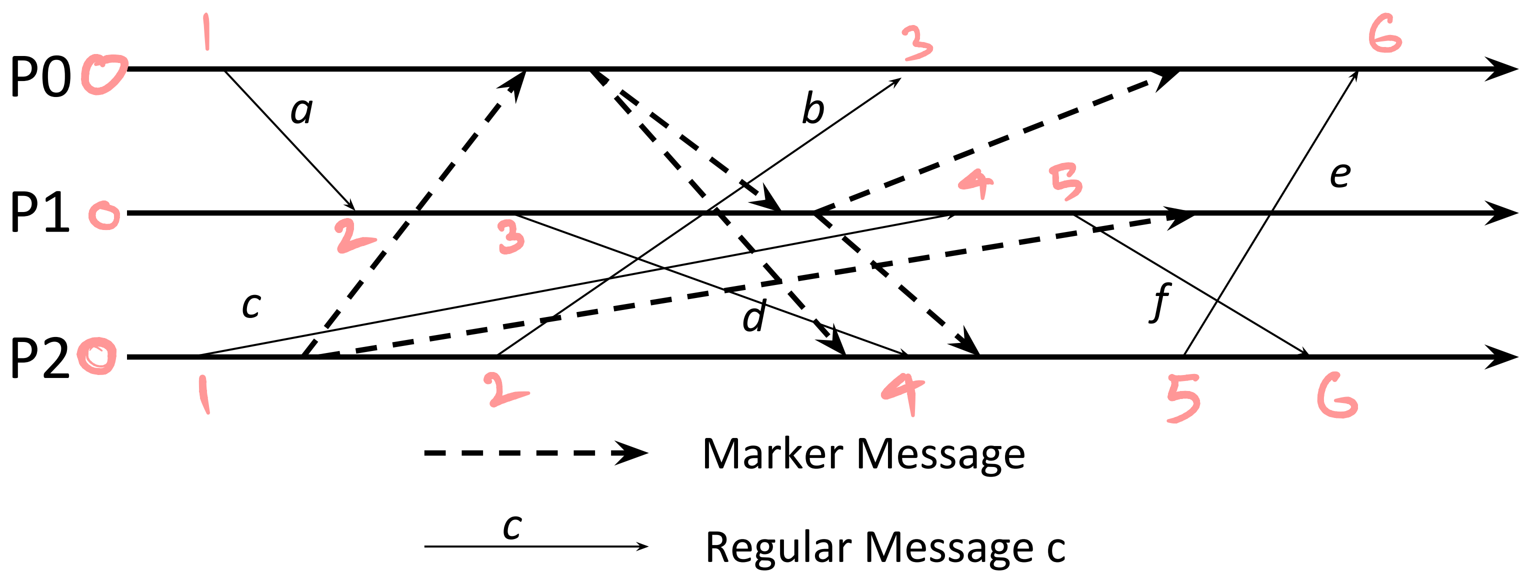


S2 = c_{02} = < > , c_{12} = < >

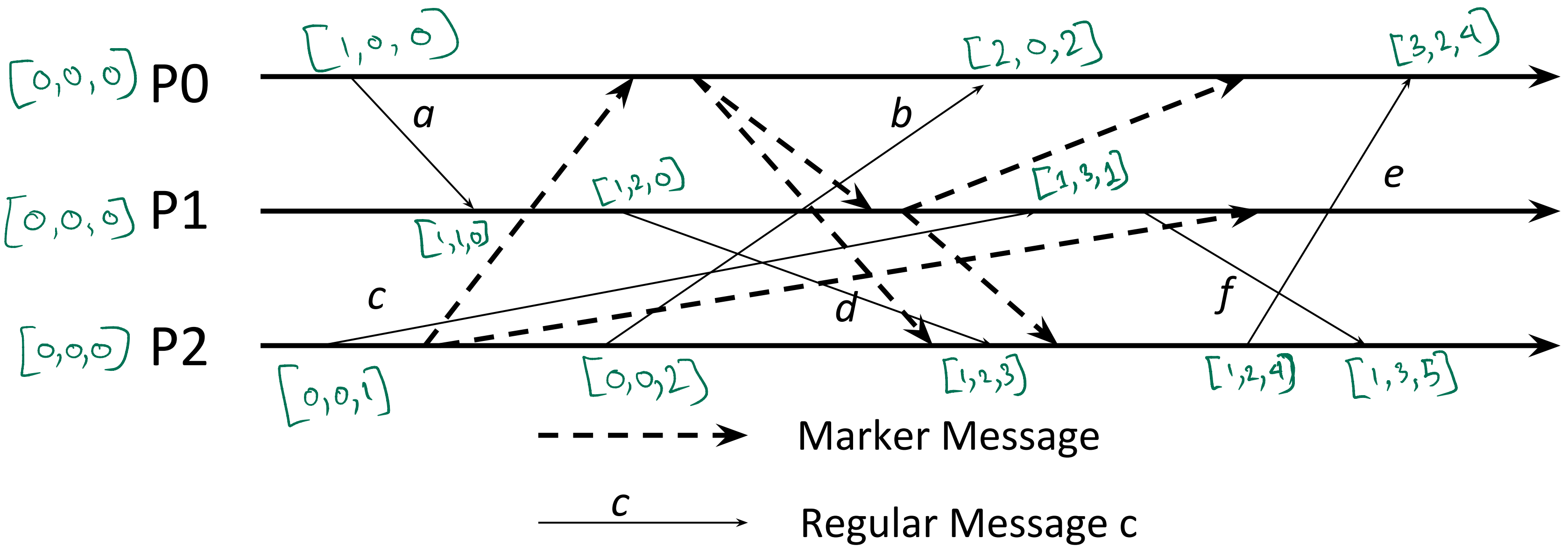
S0 = c_{10} = < d > , c_{12} = < >

S1 = c_{21} = < c >

III. Mark all Lamport timestamps for application messages on this figure for all events.
All Lamport timestamps start from zero.



IV. Mark all vector timestamps for application messages on this figure for all events.
All vector timetstamps start from zeroes.



Problem Set 2: Time and Ordering, Snapshots

1. A clock is reading 12:33:57.0 (hr:min:sec) when it is discovered to be 3 seconds fast. Explain why it is undesirable to set it back to the right time at that point and show (numerically) how it should be adjusted so as to be correct after 6 seconds has elapsed.
   It will be undesirable to set the clock back to the right time as we will have stored time events in the future in our log which will cause the system to malfunction. We will fix this by slowing down our clock. To fix the 3 second drift in 6 seconds, we will slow down our clock by a factor of ½(T_{error+}6seconds-12:33:57:0)+12:33:57:0.

2. A client attempts to synchronize with a time server. It records the round-trip times and timestamps returned by the server in the table below. Which of the times in the table below should it use to set its clock and why? What should it set its clock to? What is the estimated accuracy w.r.t to server's clock? If minimum time to send or receive a message from the server is known to be at least 8ms, does your answer change? Why?

Round-trip Time (ms)	Server Clock (hh:mm:ss)
24	12:43:33.456
28	12:43:35.235
22	12:43:36.124

   The client should use the time corresponding to the minimum round trip time of 22 seconds.

   The error for the case with no network latency will be (22-0-0)/2 = 11ms according to Christian's algorithm. The time set by the algorithm will be 12:43:36:124 + 0.011 = 12:43:36:135

   The Error will be at most (22-8-8)/2ms = 3ms with Christian's algorithm.
   The time set by the algorithm will be 12:43:36.124 + {(22 + 8 – 8)/2 ms}
   = 12:43:36.124 + 0.011 = 12:43:36.135
   While the overall time remains the same, with the server latency being 8 seconds, the error is bounded to 3ms while the error for a network with no latency is at most 11ms. This is because as the time duration for measuring intervals decreases (as we go towards no latency), measuring time gets harder.

3. If in problem 2 if it is required to sync to within 1ms of the server, is that feasible? Explain your answer.

To synchronize a clock within ± 1 ms it is necessary to obtain a round-trip time of no more than 18 ms if the network latency is 8 ms. (RTT - 8 – 8 = 1, RTT = 18ms) It is possible to obtain round-trip time of 18ms, but it may be improbable that such a time could be found. The file server risks failing to synchronize over a long period, when it could synchronize with a lower accuracy.

4. By considering a chain of zero or more messages connecting events *e* and *e′*, and using induction, show that *e*→*e′* implies *L(e)* <= *L(e')*, where L(e) is the Lamport timestamp of event e.
   Step 1: The event e_1 -> e_2 will be true if there is a direct connection between the two nodes and will imply L(e_1) <= L(e_2) (Will be true as the event e_1 will send its value (1 in this case) which will be incremented by e_2)
   Step 2: Let e_1 -> e_2 -> … -> e_{k-2} -> ek-1->e_k imply L(e_1) <= L(e_2) <= … <= L(e_{k-2}) <= L(e_{k-1}) <= L(e_k) be true for k causal events
   Step 3: To Prove: L(e_1) <= …. <= L(e_n) will be true for an event n = k+1 that is e_n->e_{n+1} if there is a causal relationship between them
   Let there be a chain of events connection e and e' such that e = e_1 and e' = e_n.
   As these are connected, e_1->e_n and from the Induction hypothesis, L(e_1) will be less than L(e_n). Hence L(e) will be less than L(e').

5. Show that V_j[i] <= V_i[i] where V_i is vector clock at process i.
   Lets us assume that V_j[i] is strictly greater than V_i[i]. If this is true then every connection from j to i will increase i's value in the vector array for V_i. This will also imply that in the beginning state, P_j will have detected an event e_i before P_i itself but this is not possible as there is no way for P_j to know the events that occur at Process i without process i without process i recording it. This will also not follow the property of vector clocks that state that "If ej occurred before pj recorded its state, then ei must have occurred before pi recorded its state". As this will not be possible, our assumption must be wrong and hence V_j[i] must always be less than V_i[i].

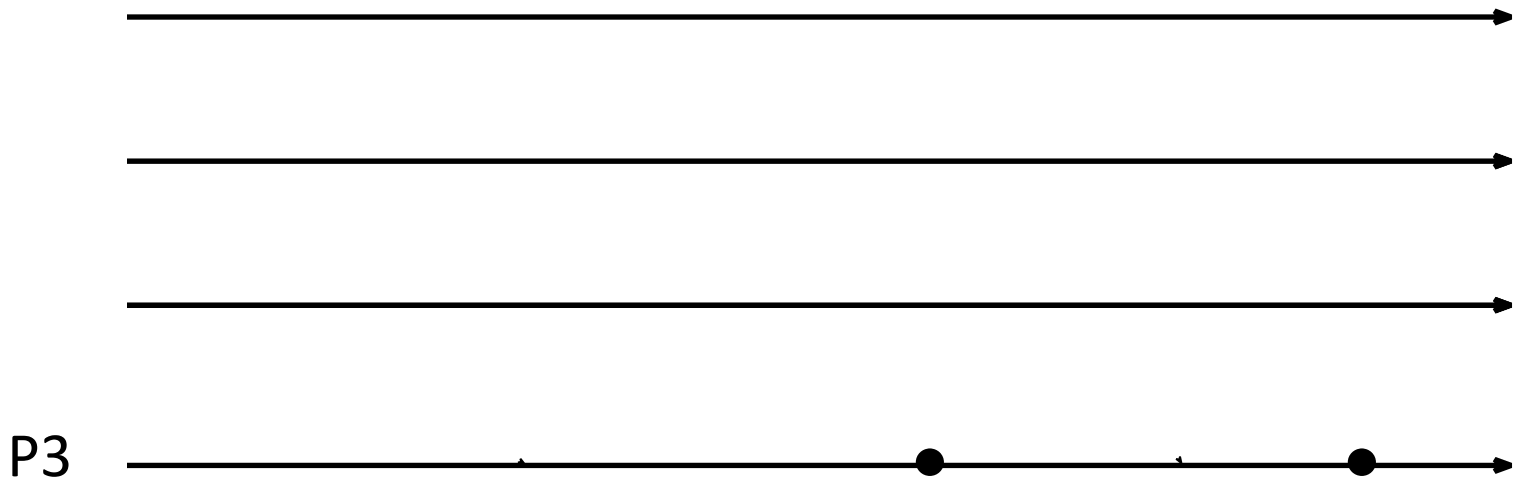
6. In a similar fashion to 4, show that *e*→*e′* implies *V(e)* <= *V(e')*
   Step 1: e_1->e_2 will imply V(e_1) <= V(e_2) as e_1 will be connected to e_2 and hence will at least have all the values of the vector smaller than itself from e_1 and will have all the values greater than e_1's vector array its existing vector array (e_2)
   Step 2: Let e_1->e_k imply V(e_1) <= V(e_k) be true for all k (Assuming there is a chain of events leading from e_1 to e_k)
   Step 3: To prove: V(e_1) <= … <=V(e_n) where n = k+1, e_1 = e and e_n=e'.
   Let the events e_1 be connected to e_n by a series of events such V(e_1) <= V(e_k) by the induction hypothesis. As e_k is connected directly to e_{k+1}, e_k->e_{k+1} and V(e_k) <= V(e_{k+1}) is true. This will imply that V(e_1)<= V(e_{k+1}) and as k+1 = n, e_1->e_n and V(e_1) <= V(e_n) (by the Induction Hypothesis).

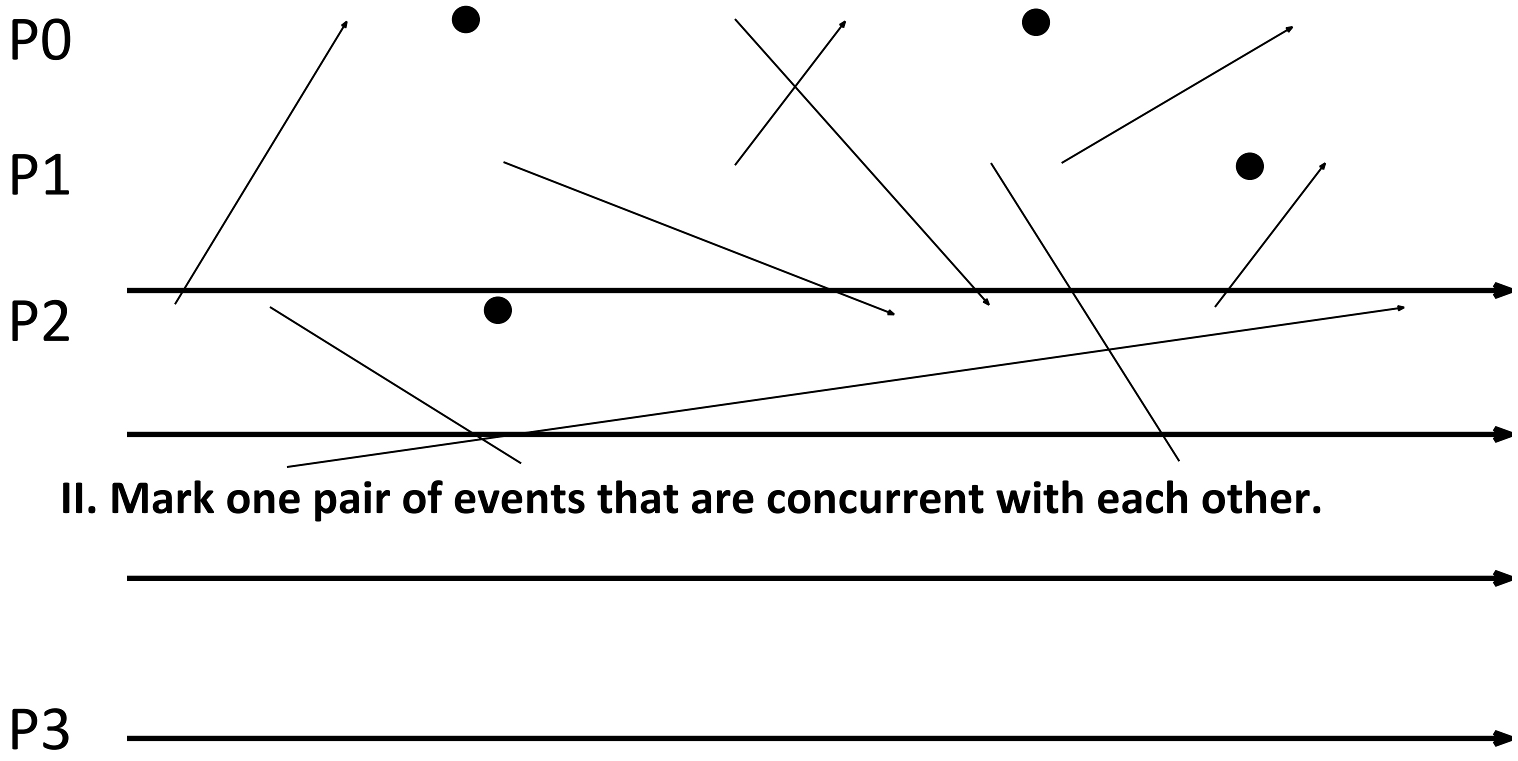
7. Using result from 5, show that if *e* and *e′* are concurrent that neither *V(e)*<= *V(e*') nor *V(e')*<= *V(e*). Thus show that if *V(e*) < *V(e*') then e→e'.

   Let e and e' be concurrent and let e occur at process i and e' occur at process j . Because these events are concurrent, none of the messages sent from the I'th process sent after event e (inclusive) will have sent its timestamp to process j by the time e' occurs at pj, and vice versa. We know that the source increments its value in the vector clock by 1 before sending a message and the receiving event increments its value by 1 only after receiving the event (and other entries if their timestamps are greater than the receiving events), it follows that V j [k] < V i[k] and Vi[j] < Vj[j] and therefore that neither V(e) <= V(e') nor V(e') <= V(e).

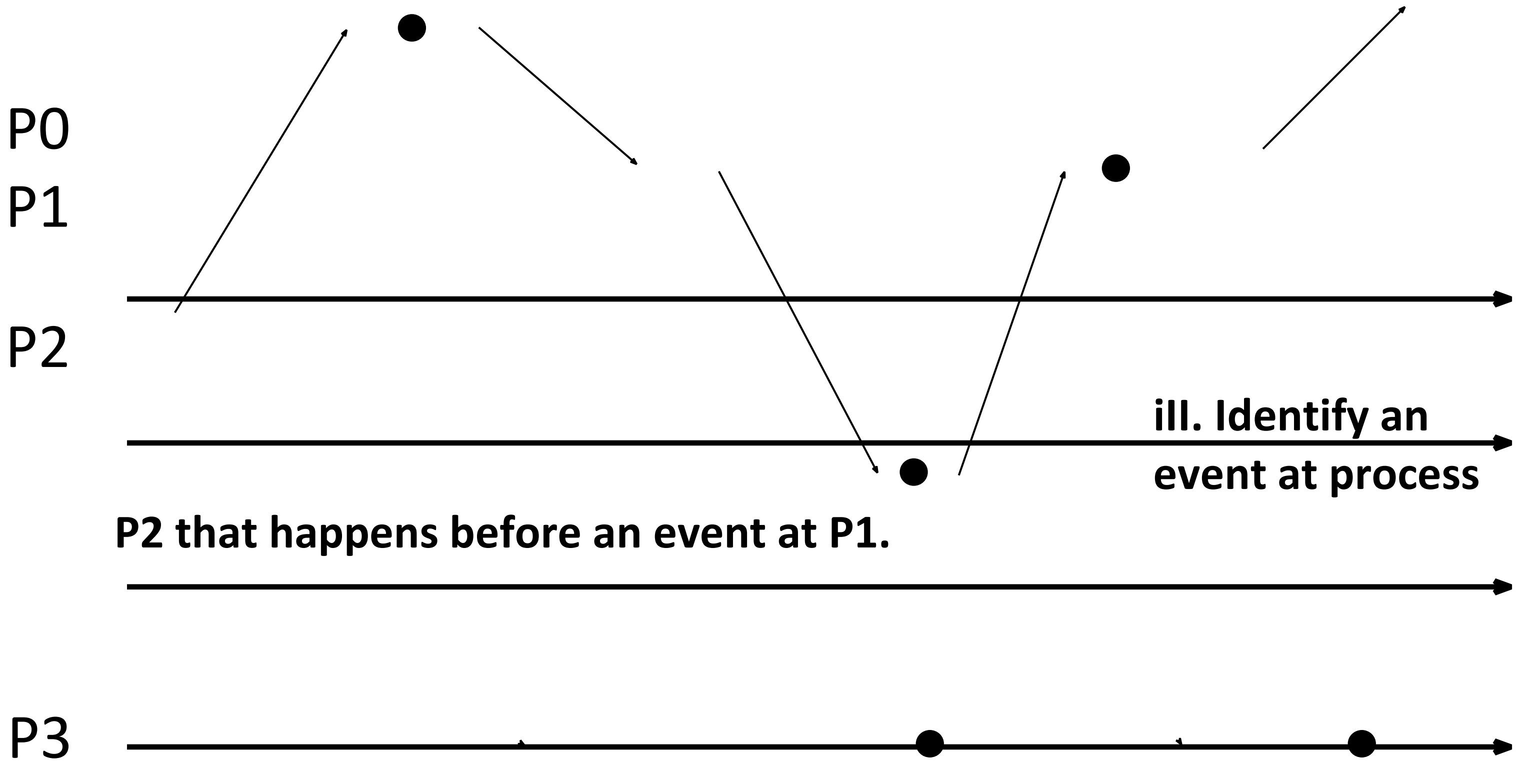
   This can only be possible when the two events V(e) < V(e') are causal. Hence it is proved that e -> e'.

I. Mark one pair of events that are concurrent with each other.

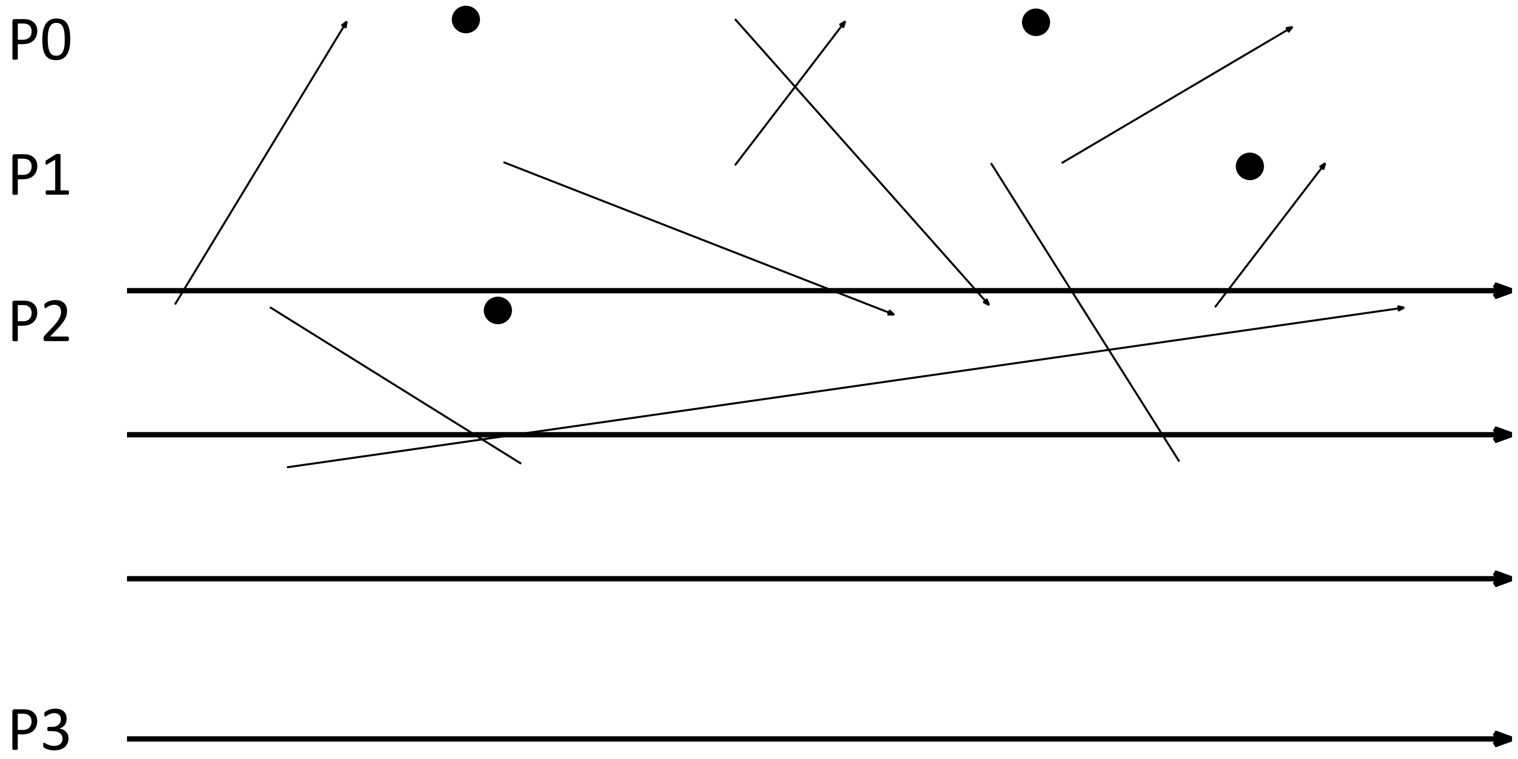




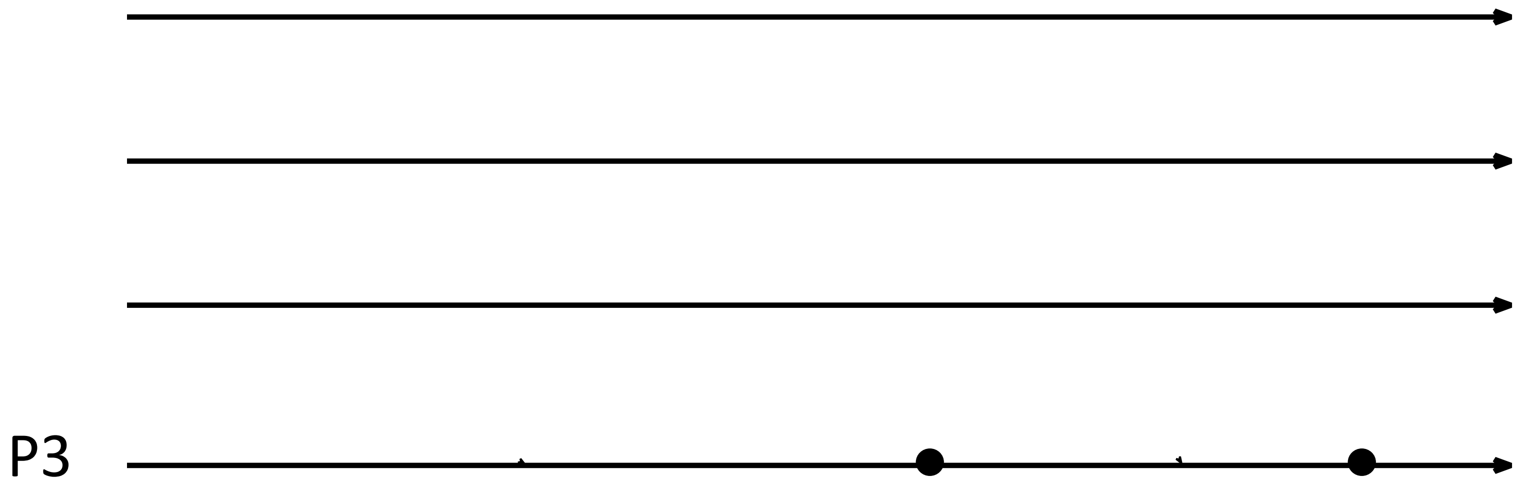
II. Mark one pair of events that are concurrent with each other.

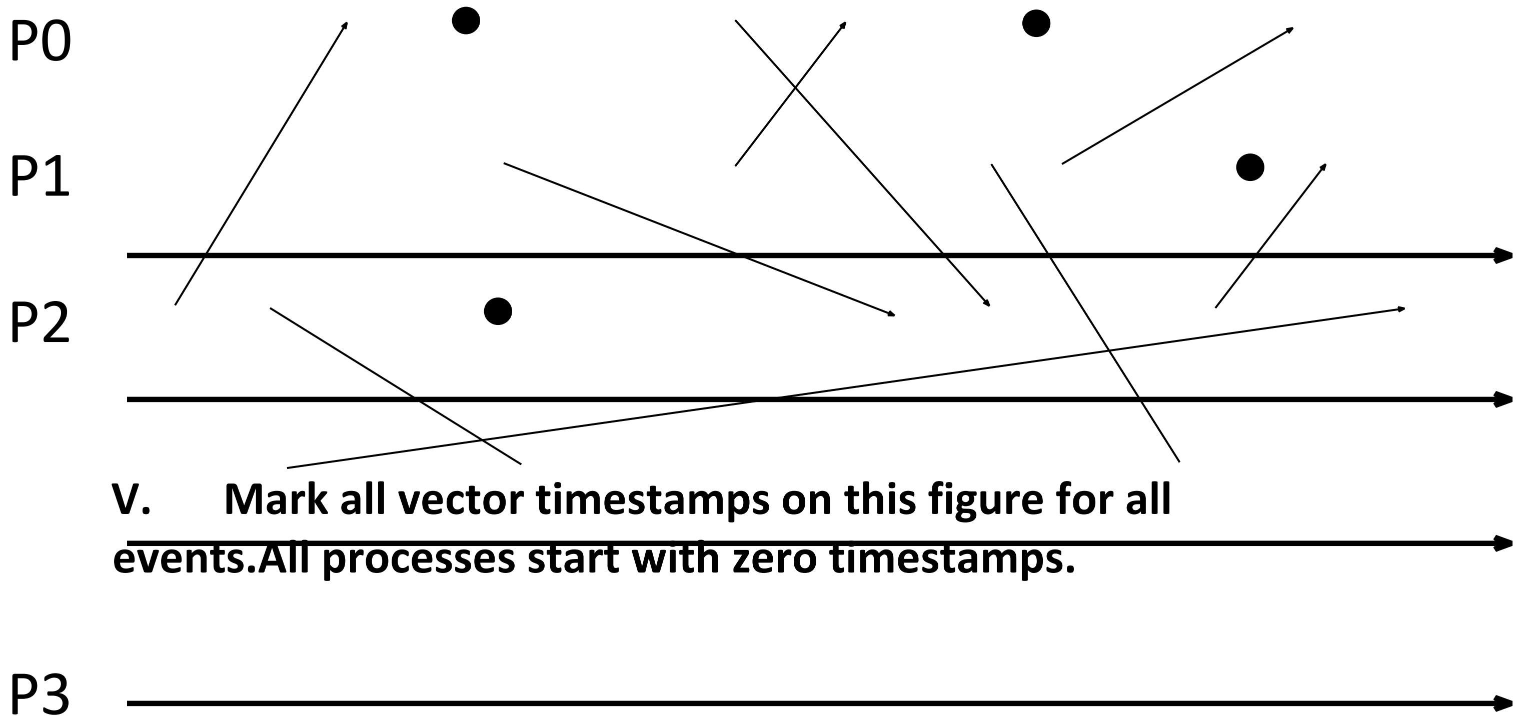


iII. Identify an event at process P2 that happens before an event at P1.

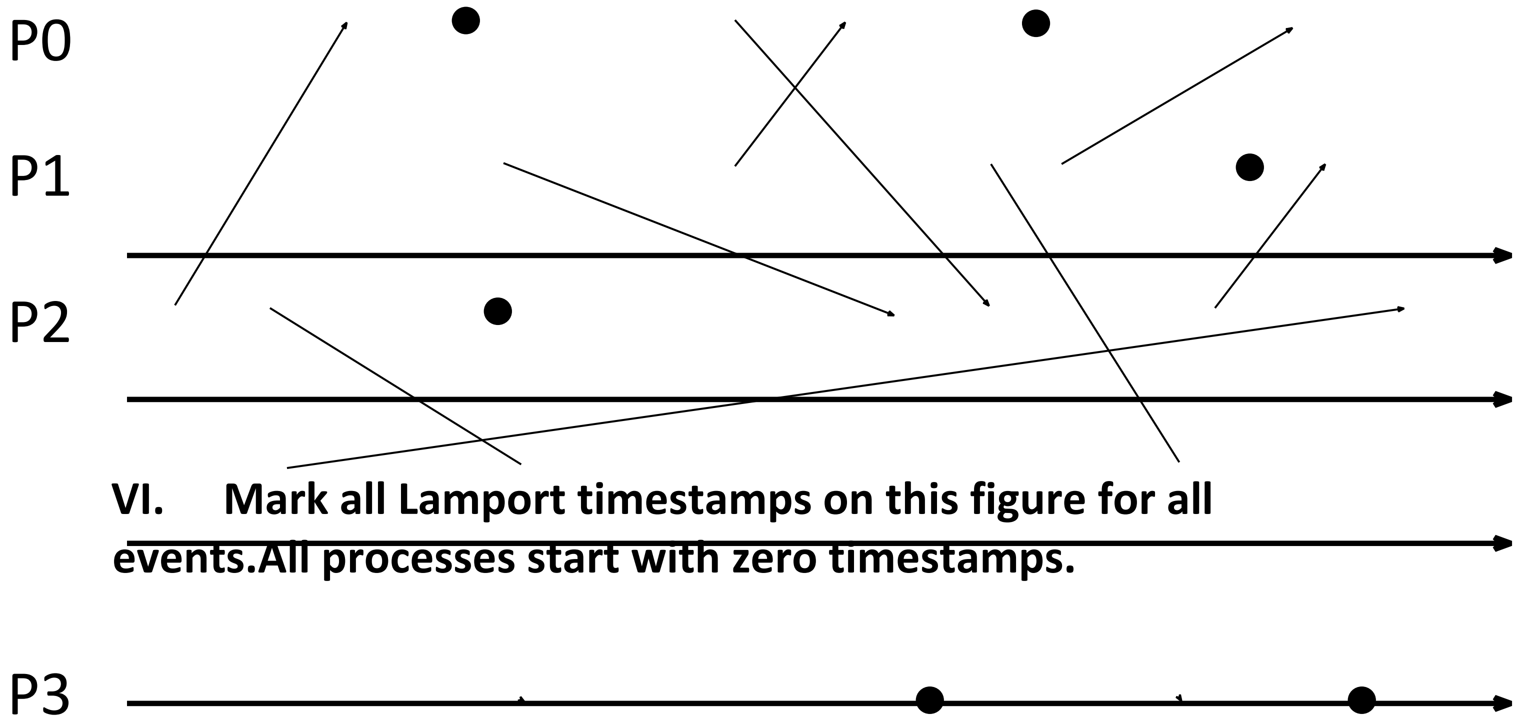

P0
P1
P2
P3

IV. **Mark all Lamport timestamps on this figure for all events.All processes start with zero timestamps.**

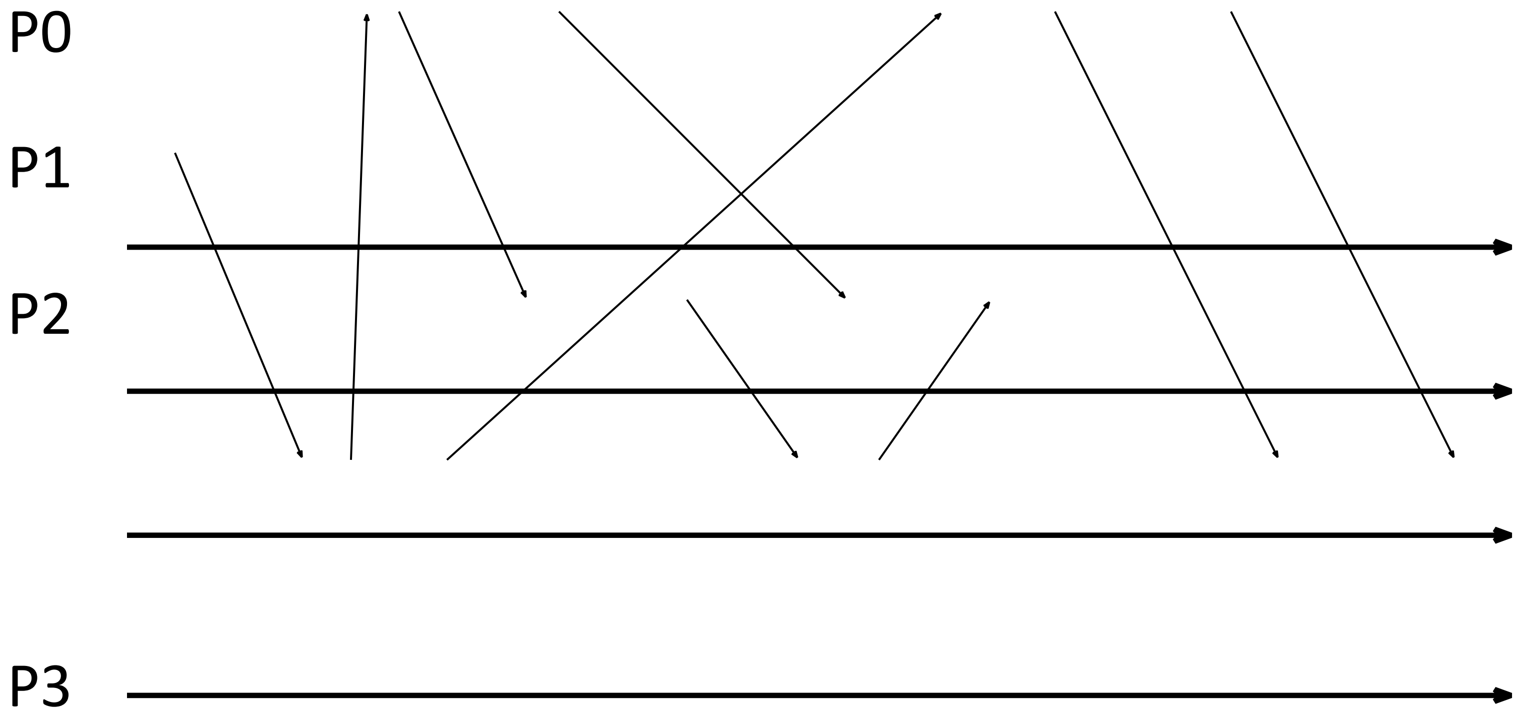




V. **Mark all vector timestamps on this figure for all events.All processes start with zero timestamps.**



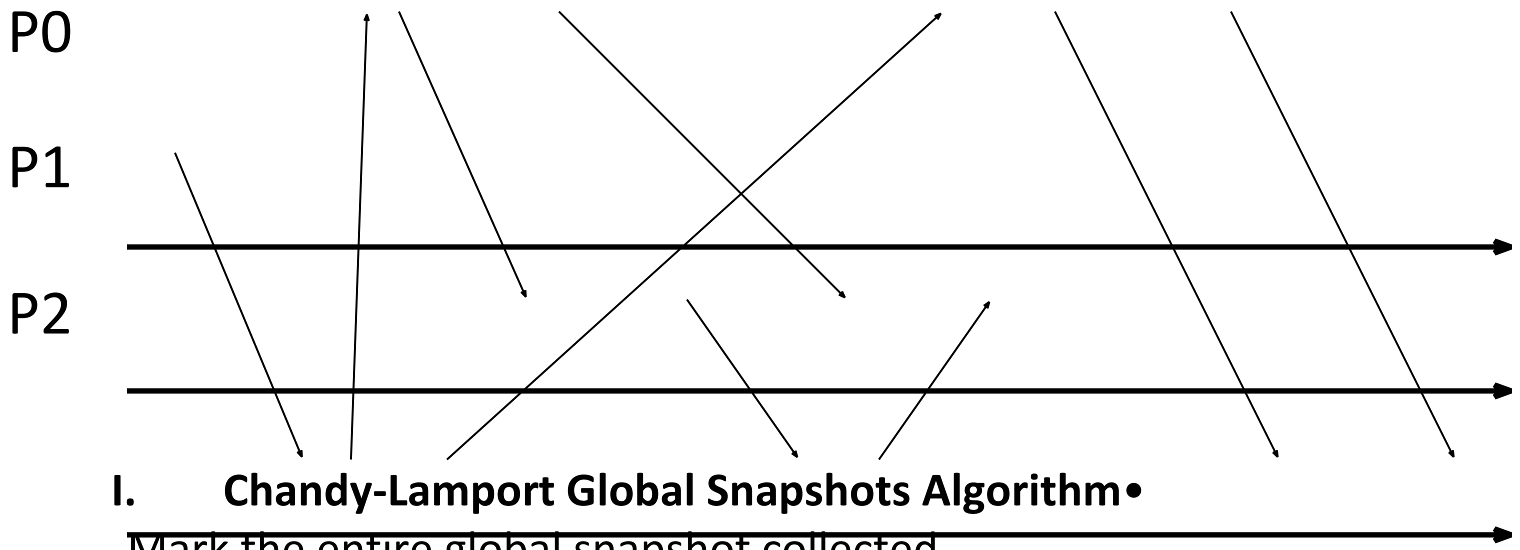
VI. Mark all Lamport timestamps on this figure for all events.All processes start with zero timestamps.

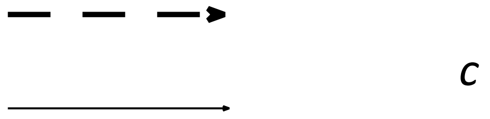

P0
P1
P2
P3

VII. Mark all vector timestamps on this figure for all events.All processes start with zero timestamps.

I. Chandy-Lamport Global Snapshots Algorithm•

Mark the entire global snapshot collected.

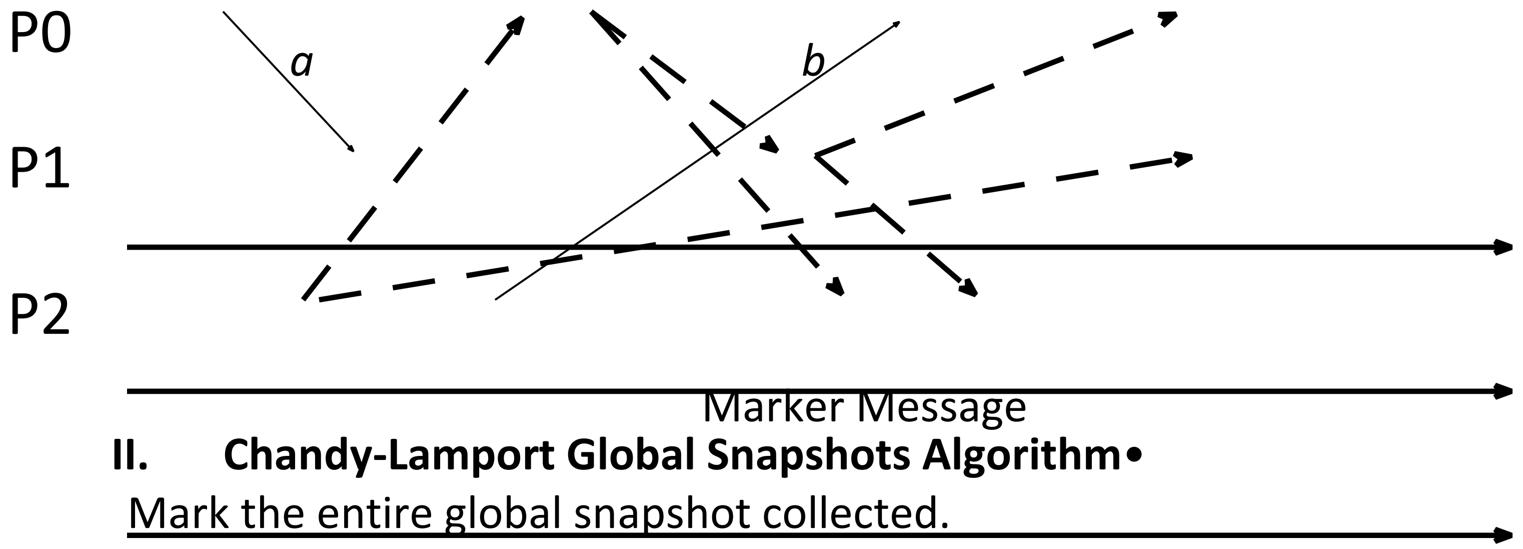


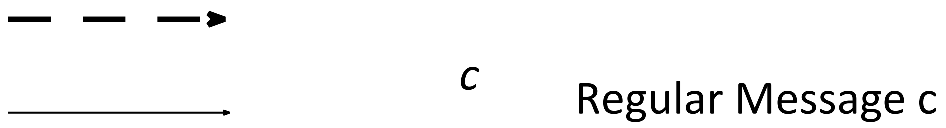


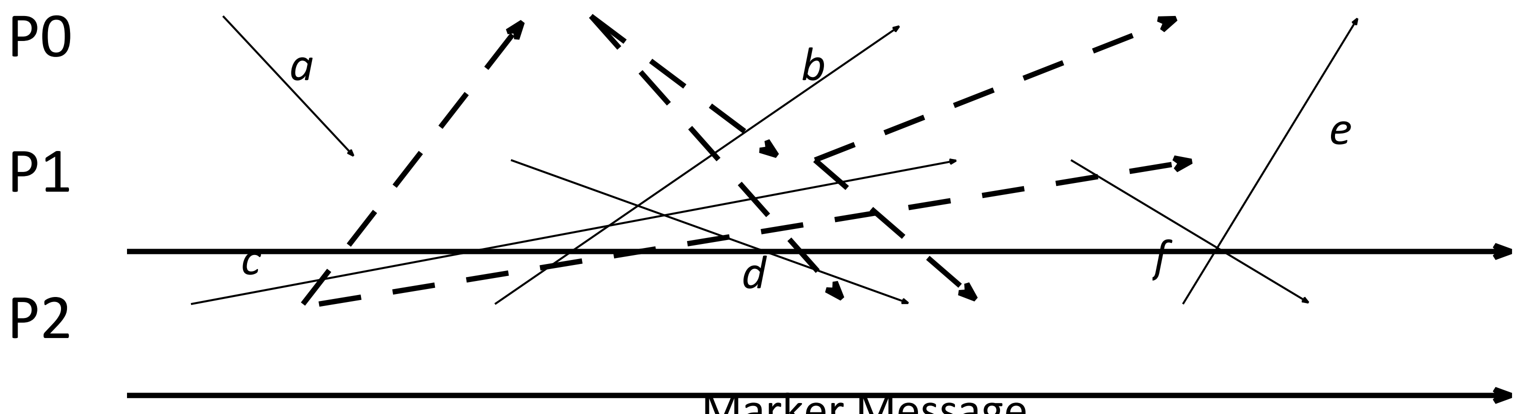
Regular Message c

II. Chandy-Lamport Global Snapshots Algorithm

• Mark the entire global snapshot collected.



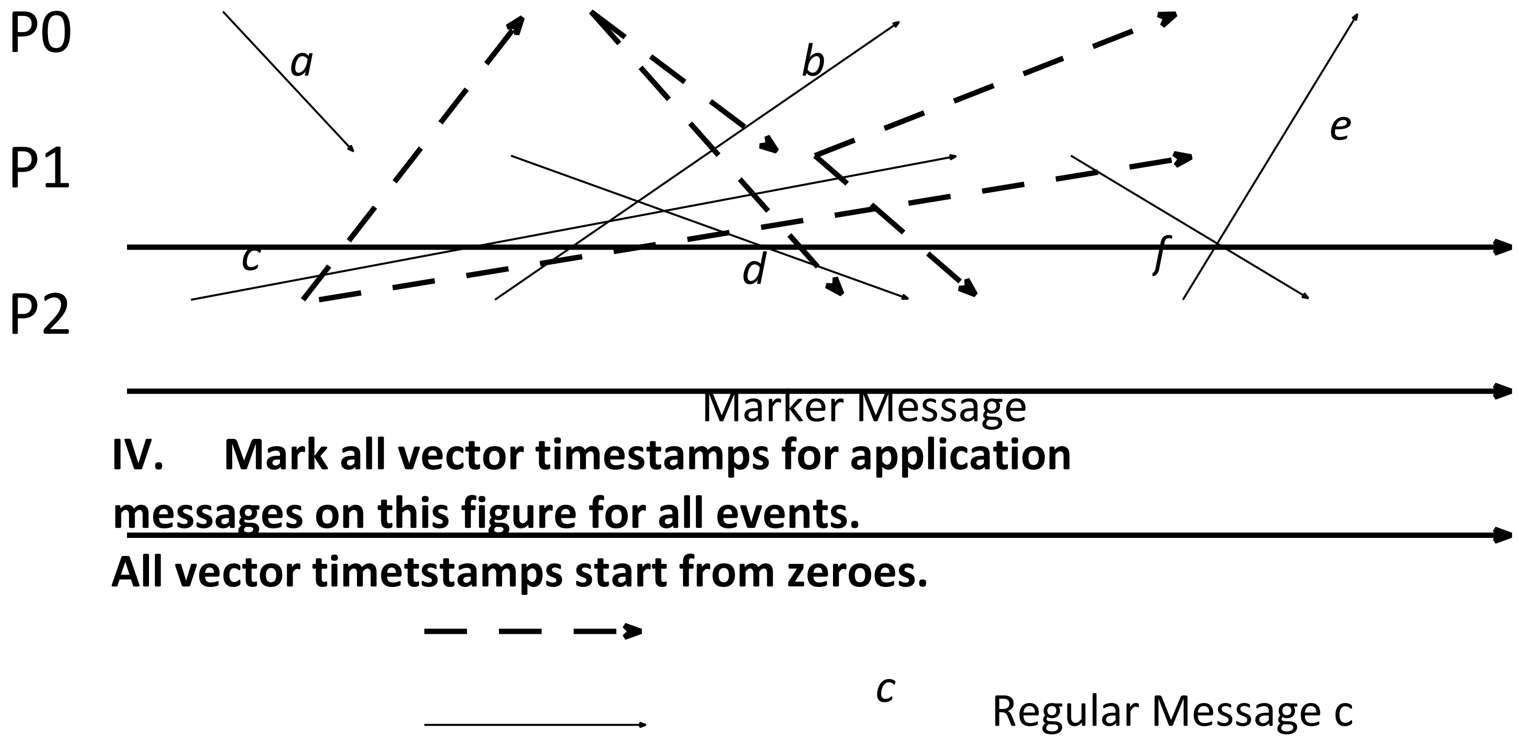




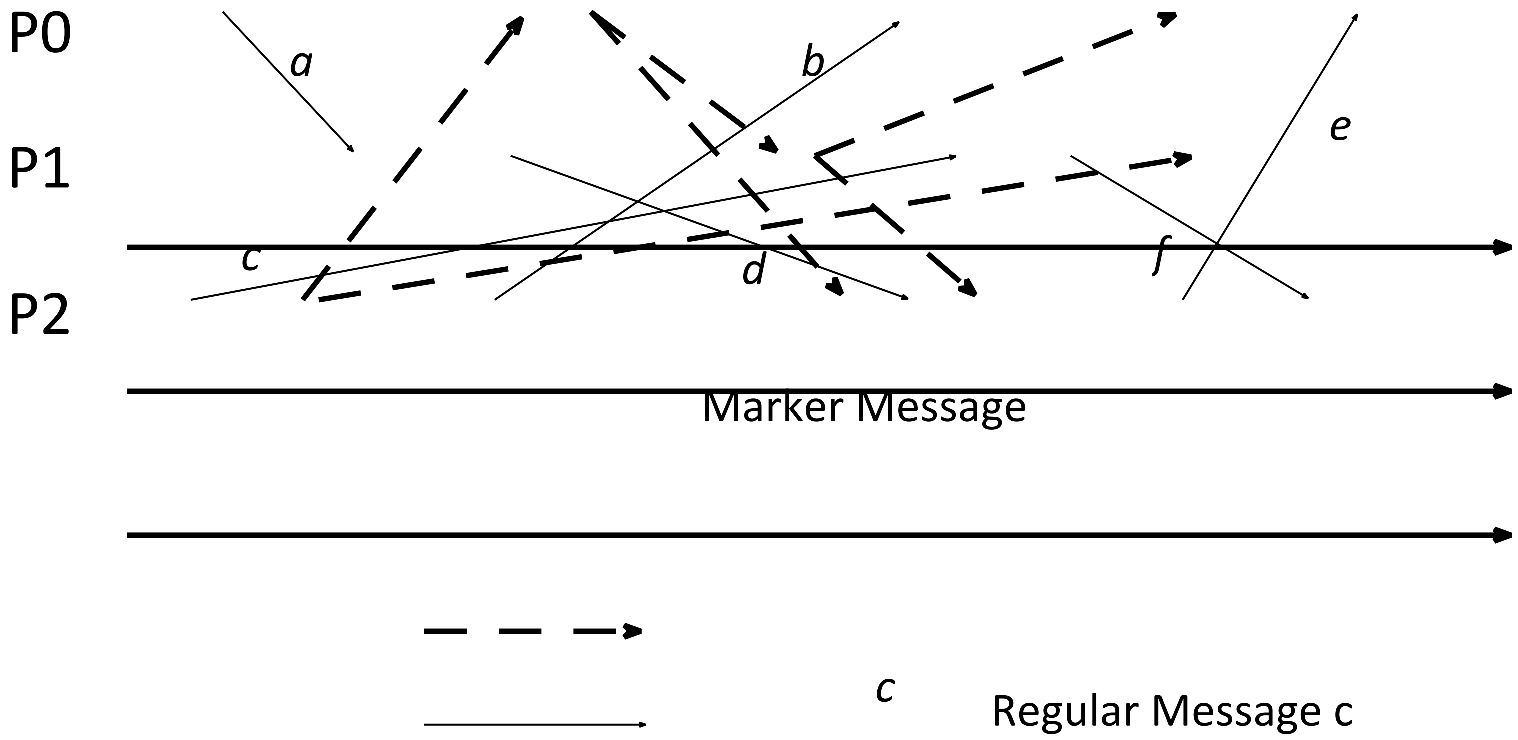
Marker Message

III. Mark all Lamport timestamps for application messages on this figure for all events.
All Lamport timestamps start from zero.

Regular Message c



IV. Mark all vector timestamps for application messages on this figure for all events.
All vector timetstamps start from zeroes.


P0
P1
P2
a
b
c
d
e
f
Marker Message
c
Regular Message c