**Northeastern University**

CS 6650 Scalable Dist Systems

**Homework #3** [100 points]

**Name: Nisarg Patel**

**Email:** [**patel.nisargs@northeastern.edu**](mailto:patel.nisargs@northeastern.edu)

***INSTRUCTIONS: Please provide clear explanations in your own sentences, directly answering the question, demonstrating your understanding of the question and its solution, in depth, with sufficient detail. Submit your solutions [PDF preferred]. Include your full name. Do not email the solutions.***

Study **Chapter 14 and 15 from** Coulouris Book Clocks and Time, Global States, Consensus

Answer the following questions using explanation and diagrams as needed. No implementation needed.

1. 14.4 [5 points]

**Ans:**

The most accurate timestamp returned by the server will be the one having the least round-trip time since width of the range of times is proportional to the round-trip time.

Since the third synchronization has the least round-trip time (*Tround*= 20 *ms*), we should use the timestamp received during this attempt to set the time (*t* = 10:54:28.342).

From this, we have:

*Time* = *t* + *Tround*/2

= 10:54:28.342 + 20 ms/2

= 10:54:28.342 + 00:00:00.010

**= 10:54:28.352**.

Thus, the client should set time to 10:54:28.352.

We can consider minimum transmission time(*min*) to be 0 *ms* since it is unknown at this point. We have:

Accuracy = ± (*Tround*/2 – *min*)

**= ±10 *ms***.

The accuracy of the setting is ±10 *ms*.

Now it is given that the least time between sending and receiving a message in the system = 8 *ms*.

Thus, we can consider *min* = 8 *ms*.

But since the range is still directly proportional to the round trip time, we will still choose the third synchronization and the client should still set the time to 10:54:28.352.

But the accuracy will be reduced to:

Accuracy = ± (*Tround*/2 – *min*)

= ±(10 – 8) *ms*.

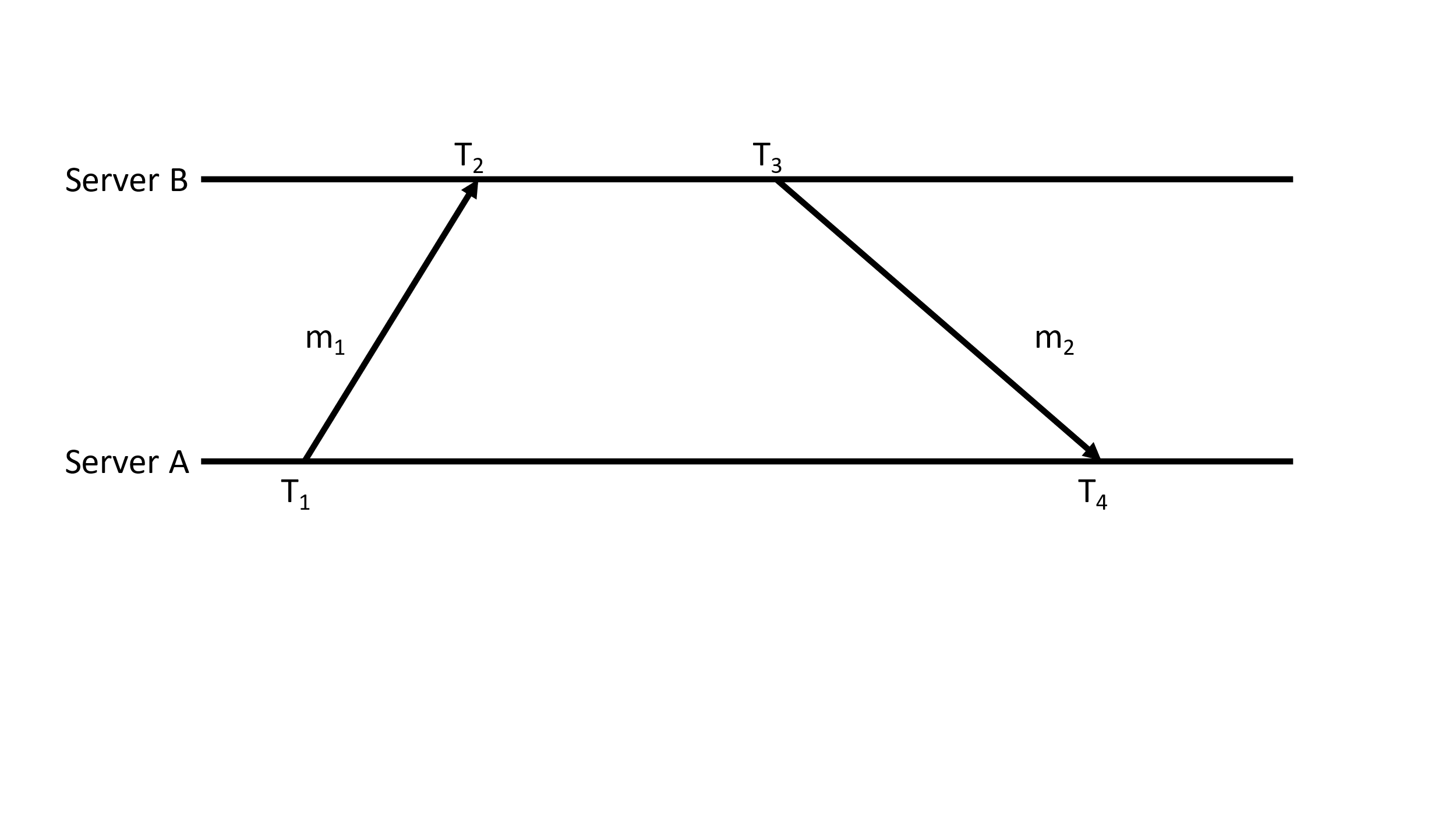
**= ±2 *ms***.

Hence the accuracy of the setting based on this new information is ±2 *ms*.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 14.7 [10 points]

**Ans:**



From the description, we have:  
*T1* = 16:34:13.430

*T2* = 16:34:23.480

*T3* = 16:34:25.700

*T4* = 16:34:15.725

Offset can be estimated by:

*oi* = (*T2* – *T1* + *T3* – *T4*)/2

= (16:34:23.480 - 16:34:13.430 + 16:34:25.700 - 16:34:15.725)

= (00:00:10.050 + 00:00:09.975)/2

= (20.025)/2 *sec*

**= 10.0125 *sec***

Hence the estimate of the offset is 10.013 *sec* (rounding to 1 *ms* precision).

We have delay,

*di* = *T2* – *T1* + *T4* – *T3*

= 16:34:23.480 - 16:34:13.430 + 16:34:15.725 - 16:34:25.700

= 00:00:10.050 - 00:00:09.975

**= 75 *ms***

Thus, the accuracy of the estimate = ± (*di*/2) = ± (75/2) *ms* = **± 37.5 *ms***=38 *ms* (rounding to 1 *ms* precision).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. A system of four processes, (P1, P2, P3, P4), performs the following events: [25 points]
   * 1. P1 sends a message to P3 (to event e).
     2. P1 receives a message from P3 (from event g).
     3. P2 executes a local event.
     4. P2 receives a message from P3 (from event f).
     5. P3 receives a message from P1 (from event a).
     6. P3 sends a message to P2 (to event d).
     7. P3 sends a message to P1 (to event b).
     8. P4 executes a local event.

When taking place on the same processor, the events occur in the order listed.

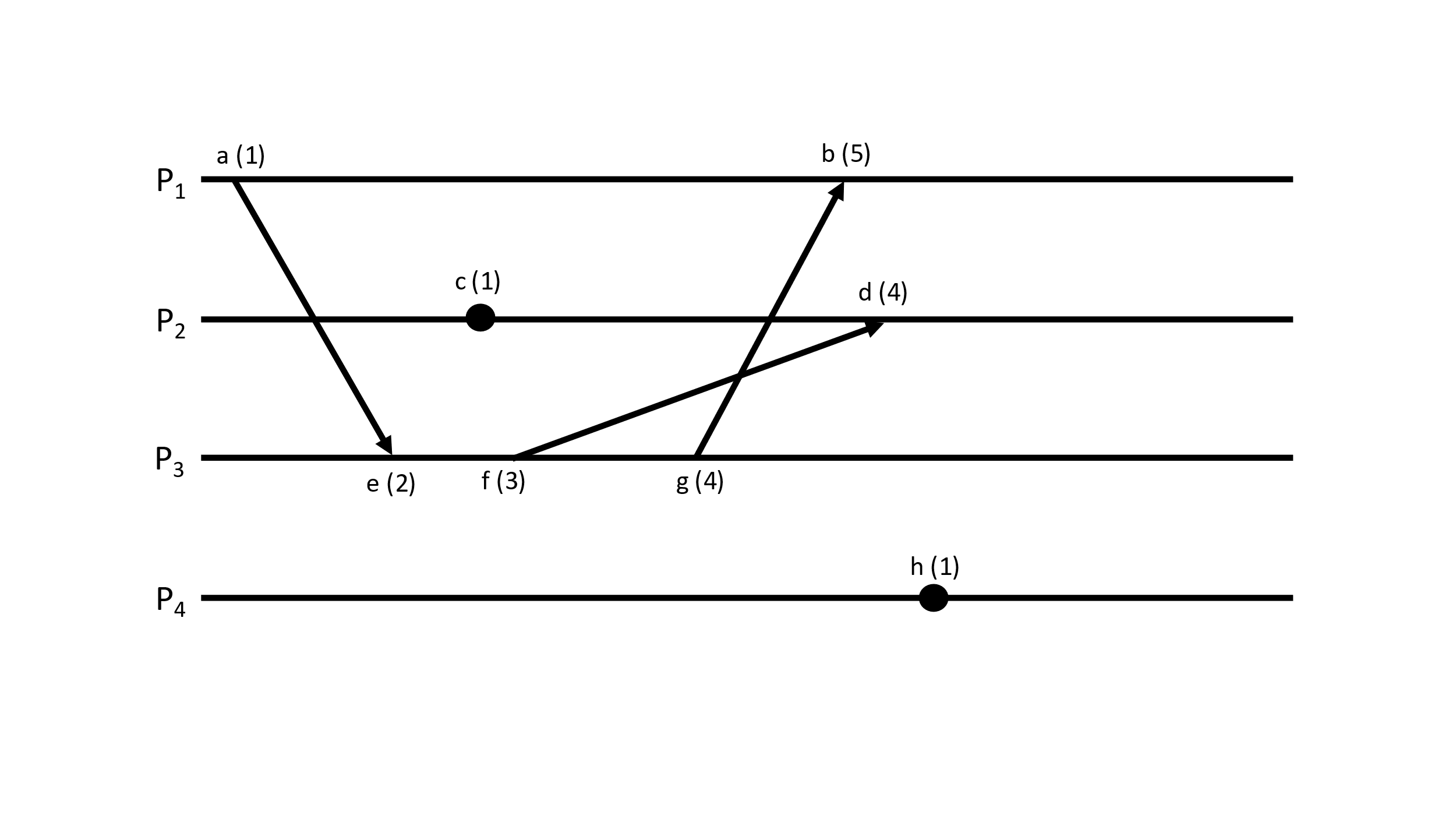
Assign Lamport timestamps to each event. Assume that the clock on each processor is initialized to 0 and incremented before each event. For example, event *a* will be assigned a timestamp of 1.

* 1. Assign vector timestamps to each event in question 2. Assume that the vector clock on each processor is initialized to (0,0,0,0) with the elements corresponding to (P1, P2, P3, P4). For example, event *a* will be assigned a timestamp of (1, 0, 0, 0).
  2. Which events are concurrent with event *d*?

**Ans:**

**(a)**

**Lamport Timestamps :**



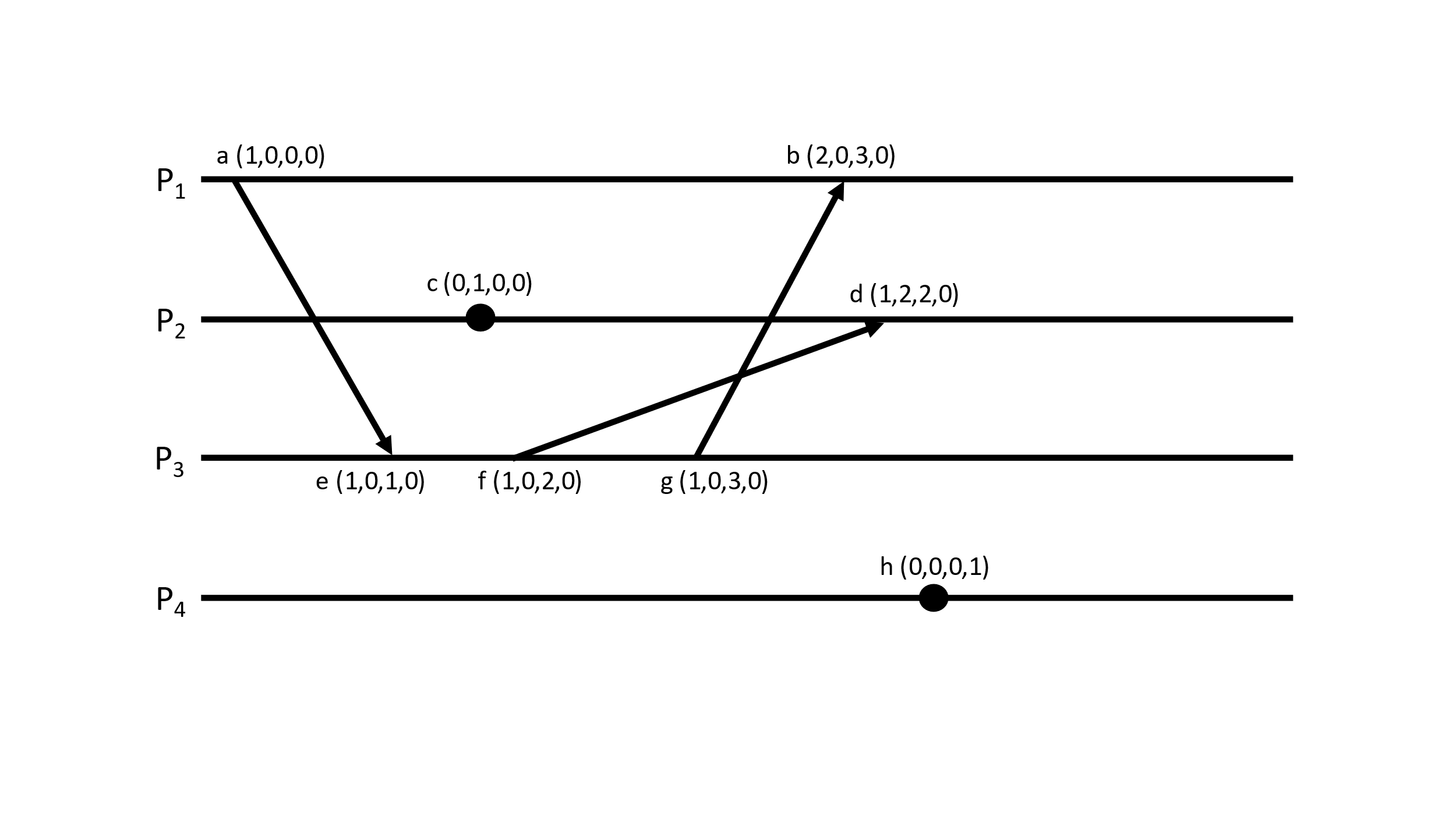
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

a. 1 b. 5 c. 1 d. 4

e. 2 f. 3 g. 4 h. 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Vector Timestamps :**



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

a. (1,0,0,0) b. (2,0,3,0) c. (0,1,0,0) d. (1,2,2,0)

e. (1,0,1,0) f. (1,0,2,0) g. (1,0,3,0) h. (0,0,0,1)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(b)**

Comparing vector time stamps from *d* (1,2,2,0), we have:

*a*<*d*, *c*<*d, e*<*d, f*<*d.*

But we can’t form a less than relation between *b*(2,0,3,0), *g*(1,0,3,0), and *h*(0,0,0,1).

Thus, we have *b* ||*d, g* || *d, h* || *d*.

Hence, events *b*, *g*, and *h* are concurrent with *d*.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. You are synchronizing your clock from a time server using Cristian's algorithm and observe the following times: [15 points]
   * + timestamp at client when the message leaves the client: 6:22:15.100
     + timestamp generated by the server: 6:21:10.700
     + timestamp at client when the message is received at client: 6:22:15.250
   * To what value do you set the client's clock?
   * If the best-case *round-trip* message transit time is 124 msec (0.124 sec), what is the error of the clock on the client?

**Ans:**

1. We are having,

*Tserver* = 6:21:10.700

*T0* = 6:22:15.100

*T1* = 6:22:15.250

Thus, we have, round trip time:

*T­round* = *T1 - T0*

= 6:22:15.250 - 6:22:15.100

= 150 *ms*

Thus, we set the time to:

*Time* = *Tserver* + *T­round*/2

= 6:21:10.700 + 75 *ms*

***Time* = 6:21:10.775**

1. We are given the best round trip message time (*min*) = 124 *ms*

So, we have the error(accuracy) as:

*Error* = ± (*Tround*/2 – *min*)

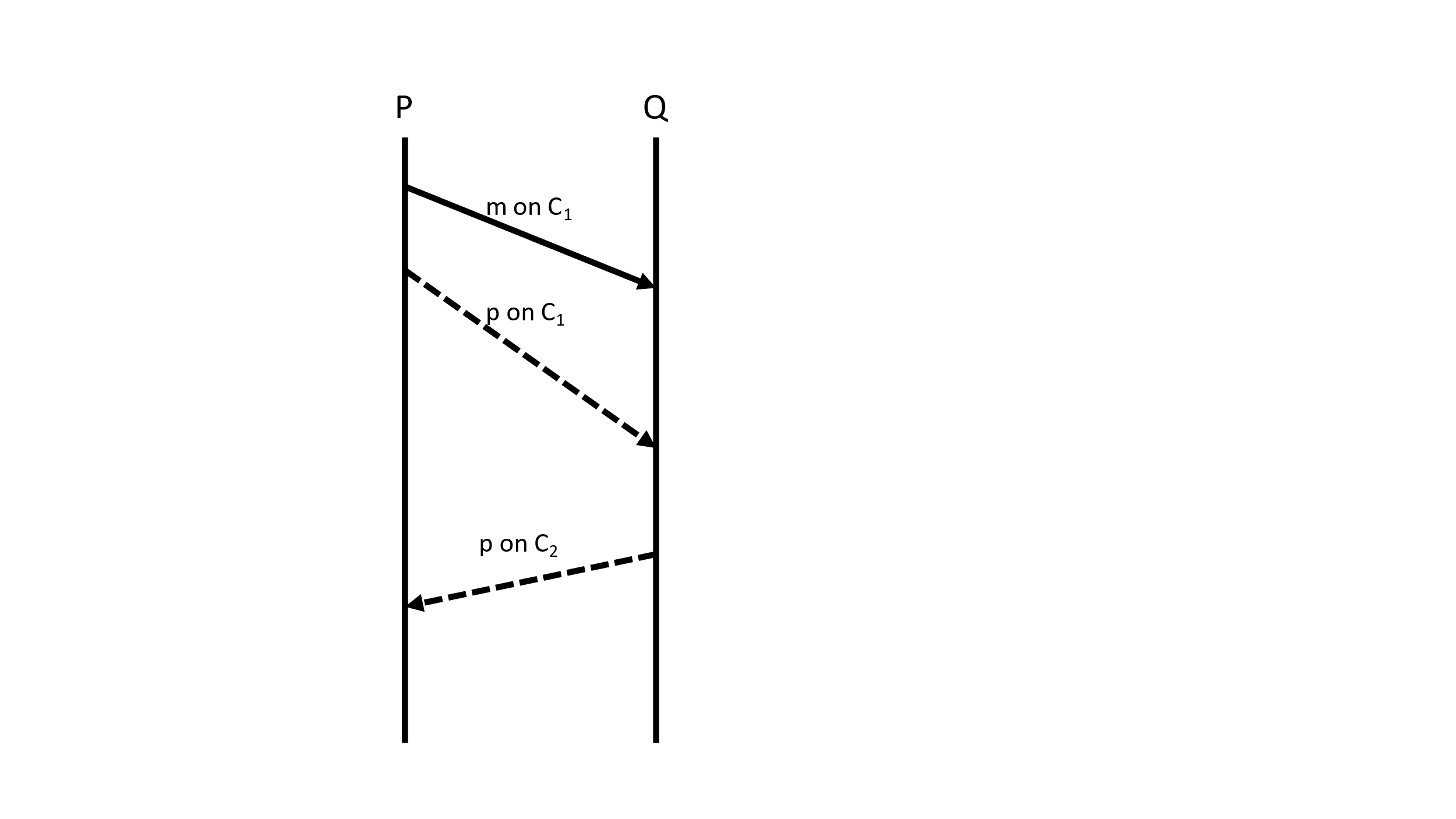
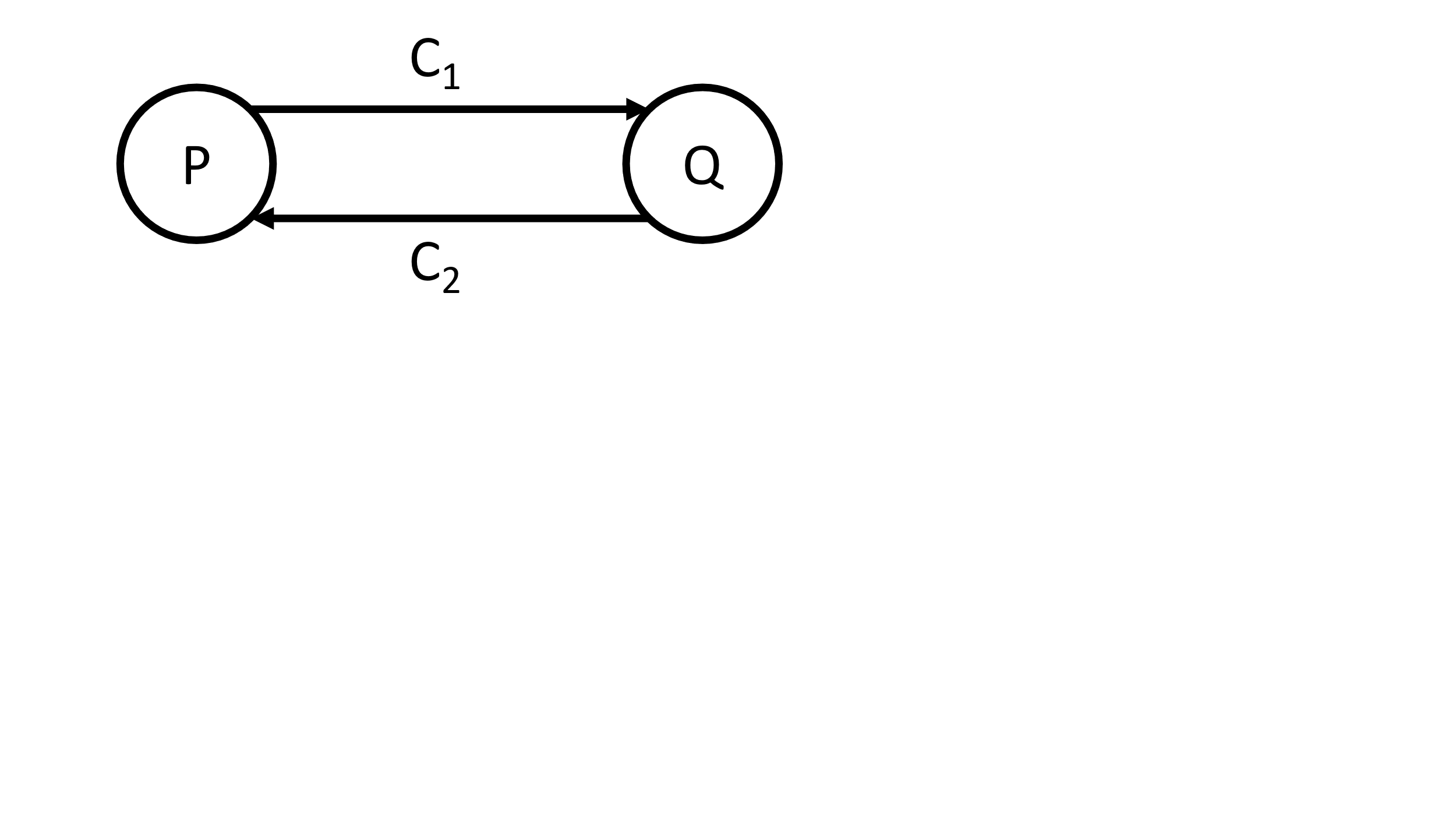
= ±(75 – 124) *ms*.

*Error* **= ±49 *ms***.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 14.14 [15 points]

**Ans:**



Let C1 be the outgoing channel from P to Q, and let C­2 be the outgoing channel from Q to P.

Let *m* be the message rotated, and let *p* be the snapshot message. In the diagram, *m* is shown as a solid line whereas *p* is shown as a dotted line.

Since P sends *m* first, the state of Q (S(Q)) is also 101 initially.

The operations of snapshot algorithm are as follows:  
1) P sends *m* on C1.

2) P initiates the snapshot algorithm.

3) P records it’s state S(P) = 101 (marker sending rule).

4) P sends marker message *p*, on channel C1 (marker sending rule).

5) Q receives message *m* first, since snapshot algorithm assumes FIFO ordering in the channel, and updates the state to 102.

6) Q receives marker message *p* on channel C1 and records its state, S(Q) = 102 (marker receiving rule).

7) Q records the state of channel C1 as empty, S(C1) = {}. It does not have any other incoming channel. (marker receiving rule).

8) Q sends marker message *p*, on channel C2.

9) P receives marker message *p* on C2, and since it has not received any message on C2­ since the initiation of the algorithm, P records the state, S(C2) = {}.

10) The algorithm terminates as all processes and channels have recorded states.

Global state(S) = {S(P) = 101, S(Q) = 102, S(C1) = {}, S(C2) = {}}.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 15.9 [10 points]

**Ans:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Using Chandy-Lamport algorithm, show when each process records its local state (you can annotate the figure) and list the channel states for each process captured in the snapshot. Black dotted lines are marker messages. Red lines are messages (A to F) [20 points]

Chart, line chart

Description automatically generated

**Ans:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_