Northeastern University

CS 6650 Scalable Dist Systems

Homework #3 [100 points]

Name: Nisarg Patel
Email: 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 (T_{round}= 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 = $t + T_{round}/2$
= 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.

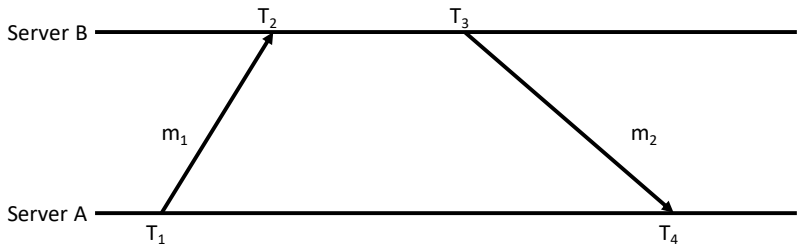
We can consider minimum transmission time(*min*) to be 0 *ms* since it is unknown at this point. We have:
Accuracy = ± ($T_{round}/2 - min$)
= ±10 *ms*.
The accuracy of the setting is ±10 *ms*.

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 = ± ($T_{round}/2 - min$)
= ±(10 – 8) *ms*.
= ±2 *ms*.
Hence the accuracy of the setting based on this new information is ±2 *ms*.

---

2. 14.7 [10 points]

Ans:



From the description, we have:
T_1 = 16:34:13.430
T_2 = 16:34:23.480
T_3 = 16:34:25.700
T_4 = 16:34:15.725

Offset can be estimated by:
$o_i = (T_2 - T_1 + T_3 - T_4)/2$
= (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,
$d_i = T_2 - T_1 + T_4 - T_3$
= 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 = ± (d_i/2) = ± (75/2) *ms* = **± 37.5 *ms*** = 38 *ms* (rounding to 1 *ms* precision).

---

3. A system of four processes, (P_1, P_2, P_3, P_4), performs the following events: [25 points]

   a. P_1 sends a message to P_3 (to event e).
   b. P_1 receives a message from P_3 (from event g).
   c. P_2 executes a local event.
   d. P_2 receives a message from P_3 (from event f).
   e. P_3 receives a message from P_1 (from event a).
   f. P_3 sends a message to P_2 (to event d).
   g. P_3 sends a message to P_1 (to event b).
   h. P_4 executes a local event.

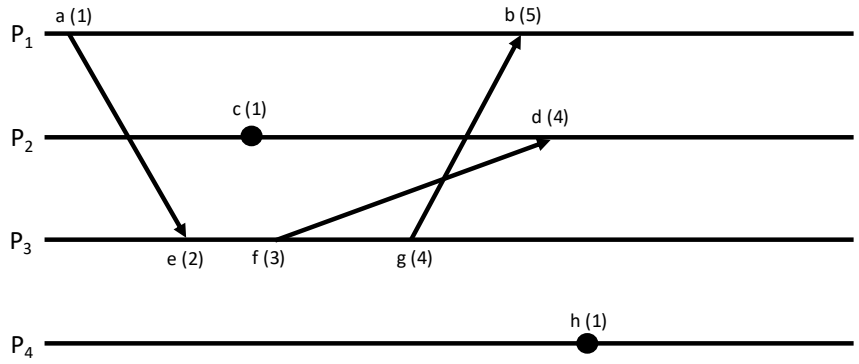
   When taking place on the same processor, the events occur in the order listed.
   Assign Lamport timestamps to each event. Assume that the clock on each processor is initialized to 0 and incremented before each event. For example, event *a* will be assigned a timestamp of 1.

   a. 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 (P_1, P_2, P_3, P_4). For example, event *a* will be assigned a timestamp of (1, 0, 0, 0).
   b. Which events are concurrent with event *d*?

Ans:

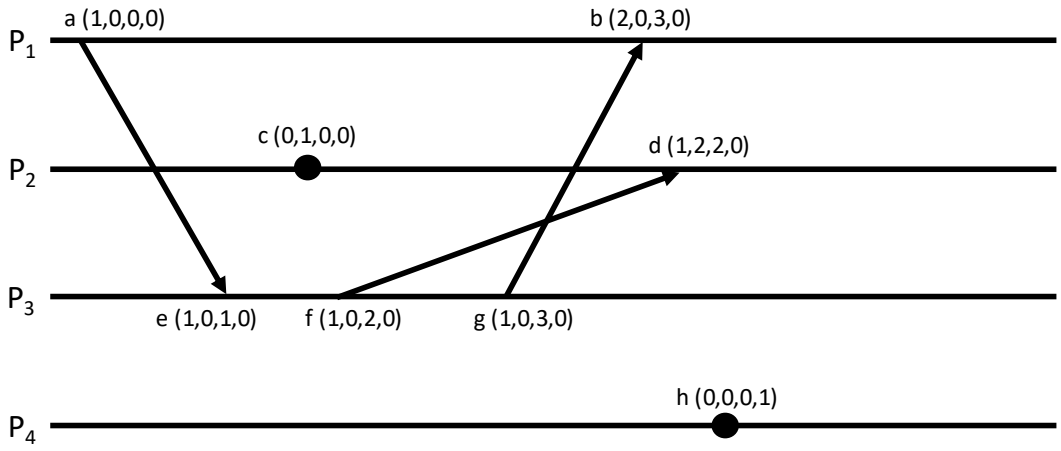
(a)

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*.

---

4. 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:

(i) We are having,

T_{server} = 6:21:10.700
T_0 = 6:22:15.100
T_1 = 6:22:15.250

Thus, we have, round trip time:

$T_{round} = T_1 - T_0$
= 6:22:15.250 - 6:22:15.100
= 150 *ms*

Thus, we set the time to:

$Time = T_{server} + T_{round}/2$

$= 6{:}21{:}10.700 + 75\ ms$

$\boldsymbol{Time = 6{:}21{:}10.775}$

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

So, we have the error(accuracy) as:

$Error = \pm\ (T_{round}/2 - min)$

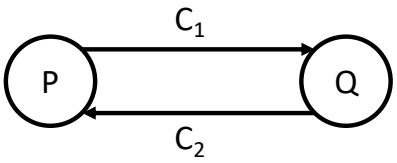
$= \pm(75 - 124)\ ms.$

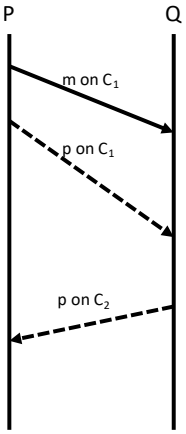
$Error$ **= ±49 *ms***.

---

5. 14.14 [15 points]

Ans:





Let C_1 be the outgoing channel from P to Q, and let C_2 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 C_1.

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 C_1 (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 C_1 and records its state, S(Q) = 102 (marker receiving rule).

7) Q records the state of channel C_1 as empty, $S(C_1) = \{\}$. It does not have any other incoming channel. (marker receiving rule).

8) Q sends marker message *p*, on channel C_2.

9) P receives marker message *p* on C_2, and since it has not received any message on C_2 since the initiation of the algorithm, P records the state, $S(C_2) = \{\}$.

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

Global state(S) = {S(P) = 101, S(Q) = 102, $S(C_1) = \{\}$, $S(C_2) = \{\}$}.

---

6. 15.9 [10 points]

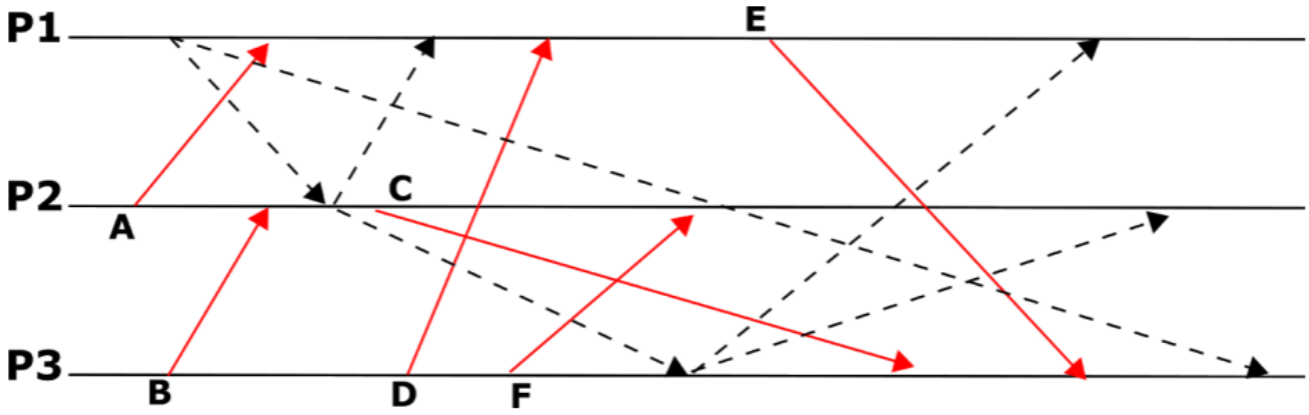
Ans:

Based on the temporary network partitions, it will be difficult to elect a single leader from the whole system to act as coordinator. But since the communication with the processes is slow, there will eventually be groups of processes that can communicate with each other faster than the processes in other groups. If such a subgroup is formed, we can consider it as a separate system and run the election(bully) algorithm on members of that group only. Thus, we would have a coordinator for that subgroup. We can perform this algorithm in all of the subgroups, thus each subgroup having coordinator among itself. After the network gets restored, the subgroups can be combined, and election algorithm can run again.
Thus, every time a change in group is detected, run the bully algorithm, for that group.
The problem now changes to detecting the fast-communicating groups (can also be a single node). For this, considering an original group of all the nodes, if any one of the nodes, detects slow communication using timeout, it can start the grouping algorithm by sending message to all members of the group. It then forms a group with those nodes who responded within a time T and sends message to other nodes to start grouping algorithm. Each node can also detect network restore if it gets message from out of the group node within time T', thus starting group algorithm itself. After this algorithm, each subgroup can call Bully Algorithm as explained above.

Note: This is just a proposed way to solve such an issue(Grouping and Subgroup Election) which could also be highly inefficient.

---

7. 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]



Ans: For the diagram given, the following steps take place:
1) P_1 initiates the snapshot algorithm, records its state as $S(P_1) = \{\}$, and sends marker message on channels C_{12}, C_{13}.
2) P_2 receives marker message at C_{12}, records state $S(P_2) = \{$sent A, received B$\}$, records $S(C_{12}) = \{\}$, sends marker message on C_{21}, C_{23}.
3) P_1 receives marker message at C_{21}, records $S(C_{21}) = \{A\}$.

4) P_3 receives marker message at C_{23}, records $S(P_3)$ = {sent B,D,F}, records $S(C_{23})$ = {}, sends marker message on C_{31}, C_{32}.
5) P_1 receives marker message at C_{31}, records $S(C_{31})$ = {D}
6) P_2 receives marker message at C_{32}, records $S(C_{32})$ = {F}
7) P_3 receives marker message at C_{13}, records $S(C_{13})$ = {E}

All the states are recorded, so the algorithm ends with:

$S(P_1)$ = {}	$S(P_2)$ = {sent A, received B}	$S(P_3)$ = {sent B,D,F}
$S(C_{21})$ = {A}	$S(C_{12})$ = {}	$S(C_{13})$ = {E}
$S(C_{31})$ = {D}	$S(C_{32})$ = {F}	$S(C_{23})$ = {}

Note: This diagram does not follow assumptions of Chandy-Lamport Algorithm, specifically, the FIFO ordering of message reception in all channels. Here C_{13} sends marker message first, but E is received first through the channel. This is leading to an inconsistent cut (recorded as E is in channel C_{13} but is never sent).
