CSCI/ECEN 5673: Distributed Systems
Spring 2022

Homework 2

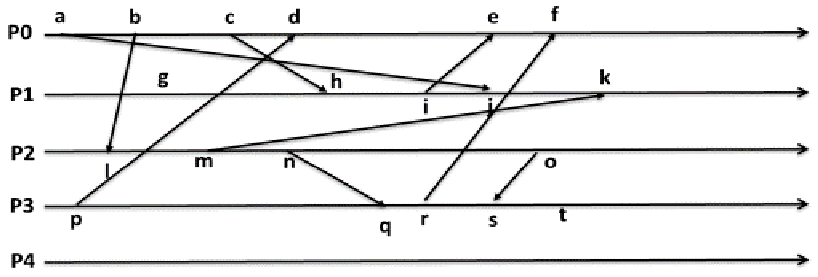
Due Date: 03/01/2022

Please submit a PDF copy of your answers via the submission link on Canvas by March 01, 2022. Write your answers in the space provided. Please DO NOT use any extra space. The space provided is sufficient for answering the questions.

Honor Code Pledge: On my honor, as a University of Colorado at Boulder student, I have neither given nor received unauthorized assistance on this work.

Name: Sai Akhil Teja M

1. Consider the following figure from Homework 1 that shows five processes ***(P0, P1, P2, P3, P4)*** with events ***a, b, c,*** ... and messages communicating between them.



a) **[15 Points]** Identify two messages between *different* pairs of processes, i.e. m1 between Pi, Pj and m2 between Pj, Pk, such that Pi $\neq$ Pj $\neq$ Pk and m1 and m2 are causally related.
Message1: message from event b on P0 to event l on P2
Message 2: message from event m on P2 to event k on P1
The above two messages are causally related

b) Is there a violation of FIFO ordering between messages in this figure? If yes, provide one example of the corresponding messages.

Yes, there is a FIFO violation. The message (let suppose m1) sent from event a on P0 is received later (event j) on P1 at event j than the message (let suppose m2) sent from event P0(event c) on P1 at event h. m1 is received later than m2 which is a FIFO property violation.

c) Is there a violation of causal ordering between messages in this figure? If yes, provide one example of the corresponding messages. Your example should not include the messages where there is a violation of FIFO ordering.
No, there is no violation of causal ordering in the above figure.

d) Provide a total ordering of messages that preserves causal ordering in this figure. Is your total ordering unique? If not, provide a different total ordering of messages that preserves causal ordering.

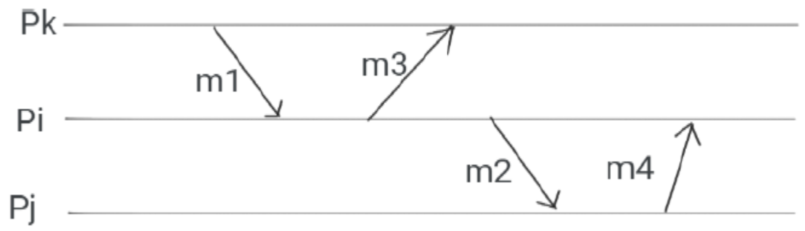
Let (a, b) denote a message whose send event is a and receive event is b in the figure. One possible total ordering can be (a, j), (b, l), (c, h), (m, k), (n, q), (o, s), (i, e), (p, d), (r, f). Total order is not unique and can have more than one total orders. A second possible total ordering is (a, j), (b, l), (c, h), (m, k), (n, q), (p, d), (i, e), (o, s), (r, f)

2. Suppose an underlying communication system provides a reliable, FIFO message communication.

   a) If the underlying communication system provides only unicast message exchanges, does it preserve causal ordering among messages? If yes, provide supporting arguments, if no, provide a counter example.

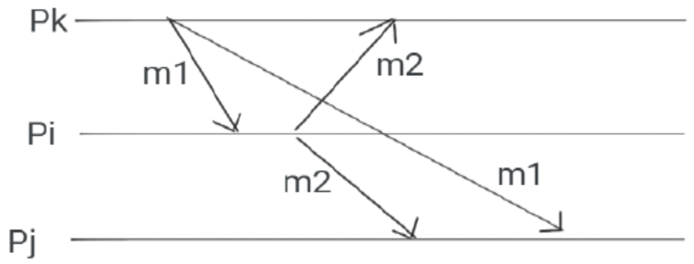
      If the underlying communication system provides only unicast message exchanges, it preserves causal ordering among messages.

      Let's say there are P_a P_b and P_c processes. P_b has received the message m1 from another process P_a and later sends a message m2 to P_c. Here m1, m2 are causally related. Had the message m2 been received at P_c before message m1, there would be a violation of causal ordering. However, this won't happen since P_c never receives m1 as the communications system is only providing unicast message exchange.



   b) If the underlying communication system provides multicast, does it preserve causal ordering among messages? If yes, provide supporting arguments, if no, provide a counter example.

      No, the causal ordering may not be preserved in such a case. Let's say there are P_a P_b and P_c processes. P_b has received a message m1 from another process P_a and has later sends a message m2 to P_c. Here m1, m2 are causally related. Let's say P_c receives m2 first and m1 next. In this case, m2 shall be delivered and not buffered though it provides a reliable FIFO because m1 and m2 are sent by different processes and are not related by FIFO. This is one example of causal ordering violation.



3. Psync provides causal delivery of messages exchanged with in a group of processes. In class it was mentioned that you can construct a total order of message delivery from the context graph by using topological sort. Explain how this can be done. Your algorithm should not use any extra messages.

Context graph properties:

• Eventually context graph of any node will have the messages in a multicast (As it is Synchrony model, after a certain time messages will be delivered). At this time buffer will not contain any messages (we wait till such time). Graph is connected i.e. we can access all the messages starting from the root .

• If messages a, b are related by FIFO, then we will have an edge from a to b.

• If messages a, b are causally related, then we will have an edge from a to b.

• Messages which are related by neither FIFO nor causal relation will not have any hierarchical relation.

Properties of topological sort:

• If there is an edge from A to B, A shall occur before B.
• If A, B are not hierarchically related, their relative order will depend on the algorithm used but there will be a definite order i.e., either A will be before B or B will be before A always.

Below algorithm is similar to DFS and can be used for generating topological sort for a context graph.

Pseudo-code:

```
• 1: Define a bool array named as visited[] and a stack;
• 2: Initialize all the vertices as not visited i.e. visited[] is 'false'.
• 3: Call the recursive function topologicalSortUtil() from context graph's root.
• 4: def topologicalSortUtil(int v, bool visited[], stack<int> &Stack):
    o 1: Mark the current node as visited.
    o 2: Recur for all the vertices adjacent to this vertex which are still unvisited
    o 3: Push current vertex to stack which stores result.
• 5: Output the contents of the stack, that are topologically sorted.
```

Note: Root node of the context graph can be ignored from the topological sort as it does not denote any message.

Because of the above mentioned properties of the topological sort if there m1,m2 have FIFO or causal relation, we will have an edge m1->m2. So m1 will be listed before m2 in topological sort. For the message nodes which are not hierarchically related, order will depend on the algorithm used (more specifically it depends on how we choose from the adjacent vertices of the graph in step 4.2) but we will have a definite order.

Hence, topological sort will provide us a total order.

4. Consider the consensus algorithm for synchronous distributed systems under scenario 3 that we discussed in class. Construct an example to show that the processes may not reach a consensus in *f* rounds with *n* processes, *n* > *f*.

   A) If any of the process sends its value to few processes but fails before sending to remaining when a round is in progress, remaining process cannot reach consensus as they look at different vectors. In the worst case, every process fails in each of the initial 'f' rounds, then f+1 round would be without a failure. So, f+1 rounds ensure that we shall have at least 1 round of exchange without a process failure.

   Example that consensus cannot be reached in f rounds:

   • Let's consider example with n=3, f=1

   • Let's say there are 3 processes P_a, P_b and P_c with initial vectors (0,u,u),(u,1,u),(u,u,1).

   • In the first round of exchange, P_a sends its value to both P_b&P_c, P_b sends to P_a&P_c. But P_c sends to P_b and fails before sending to P_a. Vectors of process would look like P_a - (0,1,u) P_b - (0,1,1) P_c – failed.

   Since P_b has a majority of 1s and P_a does not, processes cannot reach a consensus at this stage. Thus, they would need at least one more round of exchange.

5. State the safety and liveness properties for the following applications.

   a) A distributed repository of personnel data that provides personnel information to the authorized users.

      Safety properties: Unauthorized users can not access any personal data.

      Liveness properties: Authorized user will be eventually able to access their data.

   b) A highly available airline reservation system that allows clients to book seats.

      Safety properties: No two clients shall be able to book the exact same seat.

      Liveness properties: Client will be eventually able to book a seat.

   c) An air traffic control system that manages plane landings and takeoffs.

      Safety properties: No two landings or takeoffs, or a landing and a takeoff will be assigned to the same runway.

      Liveness properties: Plane requesting a landing, or a takeoff will eventually be assigned a runway.