CSCI/ECEN 5673: Distributed Systems

Spring 2017

Homework 2 Solutions

Due Date: 02/28/2017

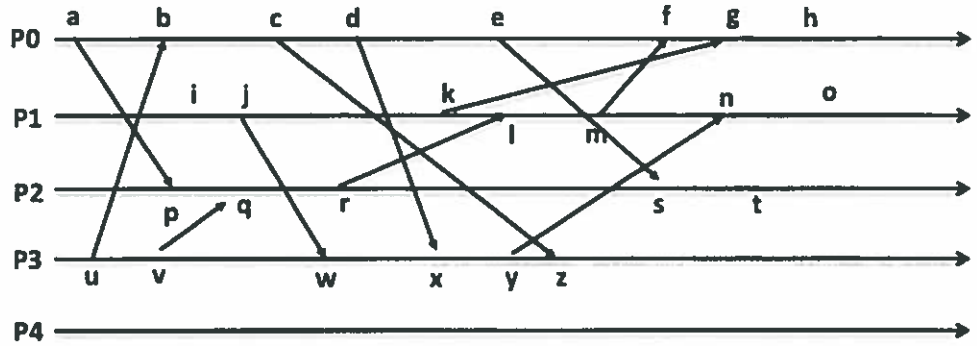
Please submit a hardcopy of your answers in class on Tuesday, February 28, 2017. BBA students may submit a PDF copy of their answers via the submission link provided on Moodle. Write your answers in the space provided. Please DO NOT use any extra space. The space provided is sufficient for answering the questions.

Topics covered: Message ordering, Consensus.
Lecture Sets: three and four.

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:

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) **[3 Points]** Identify two messages between different pairs of processes, i.e. m1 between Pi, Pj and m2 between Pn, Pm, such that Pi ≠ Pj ≠ Pn ≠ Pm and m1 and m2 are causally related.

Messages <vq> and <mf>

b) **[3 Points]** Is there a violation of FIFO ordering between messages in this figure? If yes, provide one example of the corresponding messages.

Yes. Messages <cz> and <dx>

c) **[3 Points]** 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 violation of causal ordering in this figure

d) **[6 Points]** 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.

ap, ub, vq, jw, rl, cz, dx, kg, es, mf, yn

This ordering is not unique. For example, ap and ub can be switched in the the above ordering

2. **[8 Points]** Provide an example distributed application for each of the following scenarios (total four examples). Explain your answers.

a) Scenario one: No specific message delivery ordering is needed.

Image processing where different parts of an image are processed by different processes.

b) Scenario two: FIFO message delivery ordering is needed.

Stock market updates. It doesn't matter whether stock prices of two different companies are shown in different order to two different clients. However, successive updates of stock prices of a single company at different times must be shown in the temporal order.

c) Scenario three: Causal message delivery ordering is needed, but not total ordering.

Group chat system. We discussed this class. A member may see an answer to a question before seeing the question if causal ordering is not preserved.

d) Scenario four: Total message delivery ordering is needed.

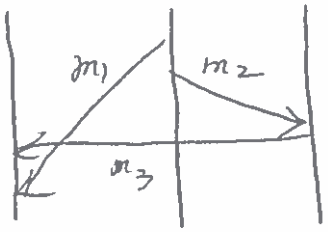
Airline reservation system. Discussed in class.

3. Consider the following definition of causal ordering among message receive events: if *send* (*m1*) → *send*(*m2*) and *m1* and *m2* are received by the same process, then *m1* must be received before *m2*.

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

   (a) **[5 Points]** If the underlying communication system provides only unicast message exchanges, does it preserve causal ordering among message receive events? If yes, provide supporting arguments, if no, provide a counter example.

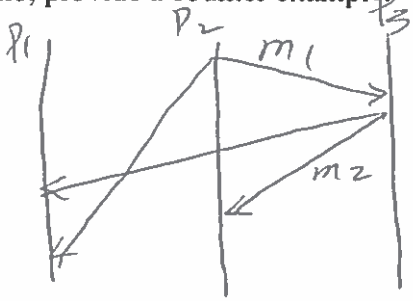
NO.



send(m1) → send(m3)
But m3 is received before m1

   (b) **[5 Points]** If the underlying communication system provides multicast, does it preserve causal ordering among message receive events? If yes, provide supporting arguments, if no, provide a counter example.

NO



send(m1) → send(m2)
but m2 is received before m1 at P1

4. **[5 Points]** 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 with only *f* rounds with *n* processes, $n > f$.

   Assume n = 3 processes, P0, P1 and P2; Initial values of P0 and P1 are 1 and initial value of P2 is 0. Further assume f = 1.

   In round 1: P0 sends its vector to P1 and then fails (before sending its vector to P2) After round 1: vector of P1 will be [1, 1, 0] while the vector of P2 will be [u, 1, 0]. Based on majority rule, P1 will decide consensus value as 1 but P2 will decide the consensus value as 0.

5. **[10 Points]** One limitation of the ISIS ABCAST protocol is that the sequencer node is a performance bottleneck. We can address this by having the role of sequencer rotate among all group members. Provide an algorithm for how you can rotate the sequencer role. Be specific in terms of what messages are used and the relevant content of those messages.

   No extra messages are needed to rotate the role of the sequencer.
   Suppose there are n group members, P_0, P_1, …, P_{n-1}.

   Protocol rule: Group member P_i is responsible for assigning a global sequence number m, if $i = m \bmod n$.

   So, P_0 will assign global sequence numbers 0, n, 2n, … P_1 will assign global sequence numbers 1, n+1, 2n + 1, …, and in general P_i will assign global sequence numbers i, n+i, 2n+i, …

   P_i assigns its global sequence number m only after it has received the sequencing messages (message sent by the sequencer that includes global sequence number) for all global numbers, 0 … m-1.

6. **[10 Points]** 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.

   This is covered in the Psync paper. Perform topological sort of a layer (called wave in the paper) in the context graph when all the nodes of that layer are in the graph (called stable wave in the paper). To find out if all nodes of a layer (say layer L) are in the graph, check if there is a node for each group member (node representing a message send by the member) in layer L or a layer following L.

7. **[12 Points]** The Byzantine Generals algorithm helps reach a consensus when less than one third of the processes undergo byzantine failure. However, it does not suggest how to diagnose such failures. Consider a system of five processes, of which at most one process can be faulty. Examine if the faulty process can be identified without any ambiguity. Investigate all relevant cases.

   The faulty process cannot be identified without ambiguity. Consider two cases:

   Case 1: A lieutenant is faulty (say lieutenant 1):

   General sends a value (say 1) to all lieutenant. Lieutenant sends 0 to all other lieutenants while all other lieutenants send 1. Lieutenant 1 will be suspected to have failed. However, the same scenario can occur is the General is faulty. The general sends 0 to lieutenant 1 and 1 to all other lieutenants.

   The non-faulty lieutenants cannot distinguish between these two scenarios.

8. Consider a distributed system comprised on n processes that is timed asynchronous, has reliable message delivery, and up to f processes may fail (fail stop failure).

   a) **[4 Points]** Does the consensus algorithm we discussed for synchronous distributed system for scenario 3 work in this new scenario? Explain your answer.

   No. The reason is that after f rounds, it is not guaranteed that the vectors of all non-faulty processes will be identical. For example, it is possible that the messages sent by a process P0 are not received by any other process in all rounds.

b) **[10 Points]** Devise a consensus algorithm for this new scenario. Explain the limitations your algorithm.

Because FLP result still applies, no algorithm can solve consensus. This is similar to the asynchronous case. Note that consensus needs to be among all non-faulty processes. Messages from a non-faulty process may be indefinitely delayed to prevent consensus.

It is possible to reach an agreement among a set of processes, but not necessarily among the set of non-faulty processes.

9. **[16 Points]** State the safety and liveness properties for the following applications.

a) Traffic light at a four-way intersection controlled by two processes. One process controls N-S direction lights and the other process controls the E-W direction lights.

Safety: N-S and E-W direction lights will never both be green at the same time.
Liveness: Lights in each direction will turn green eventually

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

Safety: A seat will never be booked by more than one client.
Liveness: each seat will eventually be booked assuming that there are sufficient number of clients.

c) An ATM that dispenses cash to the bank clients.

Safety: The ATM will never dispense incorrect cash.
Liveness: The ATM will eventually dispense cash.

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

Safety: A plane will never be allowed to land or takeoff when another plane is landing or taking off.
Liveness: A plane will eventually get permission to land or takeoff.