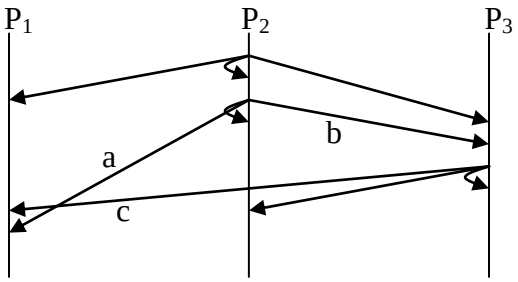
ECE428/CS425/CSE424: Distributed Systems Spring 2011

Homework 2

Each question is worth 10 points.

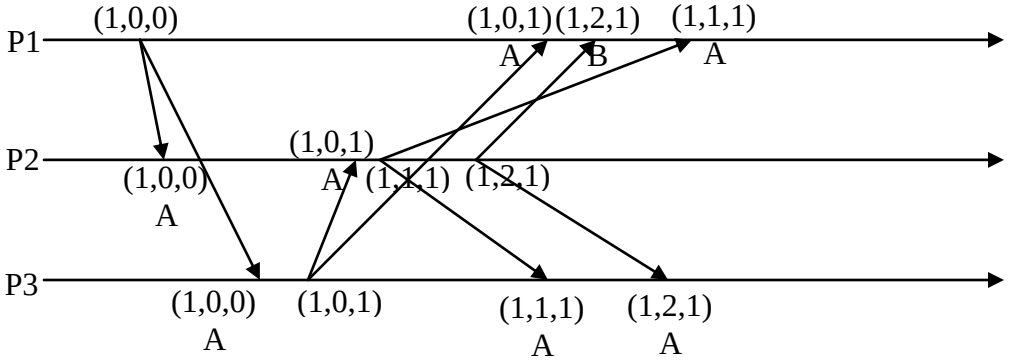
1. In the illustration below, three processes perform multicast.
   Which of the following ordering property is achieved in this example?
   (i) FIFO ordering, (ii) total ordering, (iii) causal ordering.

   (i) FIFO ordering is achieved
   (ii) Since message *b* arrives before message *c* is sent, message *a* should arrive at P1 before message *c* to achieve causal ordering. Hence, causal ordering is not achieved.
   (iii) The order of message *a* and *c* at P1 is different from the order at other processes. Hence, total ordering is not achieved.



2. Suppose that the timeline below corresponds to a system that implements the causal ordering algorithm for multicast discussed in class (Figure 12.16 in the textbook, 4th edition). Mark vector timestamp at each event in the figure, and indicate for each received message at each process whether it can be delivered immediately on receipt, or must be buffered until a later time.

   A means 'Accepted' and B means 'Buffered'.
   A vector timestamp (a,b,c) represents (a=P1's timestamp,b=P2's timestamp, c=P3's timestamp).



3. A broadcast channel is one where a message sent by a processor is received by all the processors in the system. In a reliable broadcast channel, all the processors receive the same value sent by the sender. Consider the Byzantine general's problem in a system where all the processors are on the same broadcast channel.
   (a) In the above system, is it possible to tolerate more than n/3 faulty processors?
   Yes. It is important to note that the assumption in this system is different from the original Byzantine general's problem. While in the original Byzantine general's problem, the communication of each pair cannot be heard by other processors, in this system, a message is heard by every processor.
   (b) Suggest an efficient algorithm for the Byzantine general's problem in this system.
   The commander broadcasts its message and all the non-faulty processors agree to the value sent by the commander. Agreement is reached with just one message. If a message is not received from the commander, non-faulty processor agree on a default value (e.g., 0).

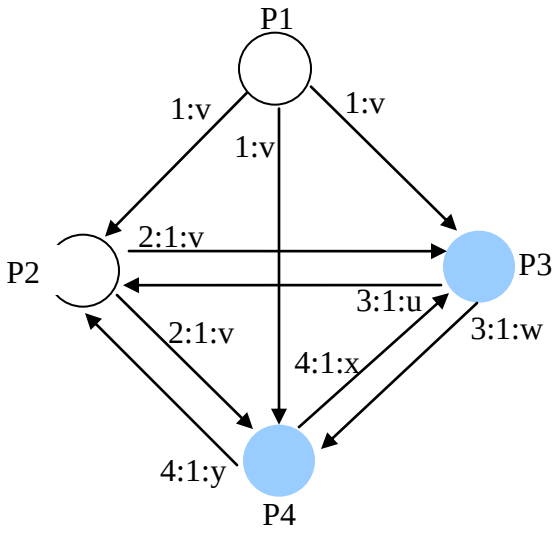
4. Exercise 12.5 from the textbook (4^{th} edition).
   We assume that the correctness requirement with faults is that at most one 'fault-free' processor should be in the critical section at any given time.
   1) A client that has not requested for a token, if detected to have failed, will not affect the system.
   2) The server performs the following steps for each client that has requested a token and has been enqueued at the server:
      a. The server checks if the client next in queue has failed or not.
      b. If the client is found to have failed, the server will drop the request and proceed to check the next client in the queue.
      c. If the client has not failed, the server will grant the token to the client.
   3) If a client in the critical section is detected to have failed, then the server ignores the token already granted and generates a new token to be granted to the next client in the request queue (if there are any, after testing for their failure).

   With a reliable failure detector, the correct clients can be granted a token even if a failed client previously owned the token. Hence, the system is fault-tolerant.

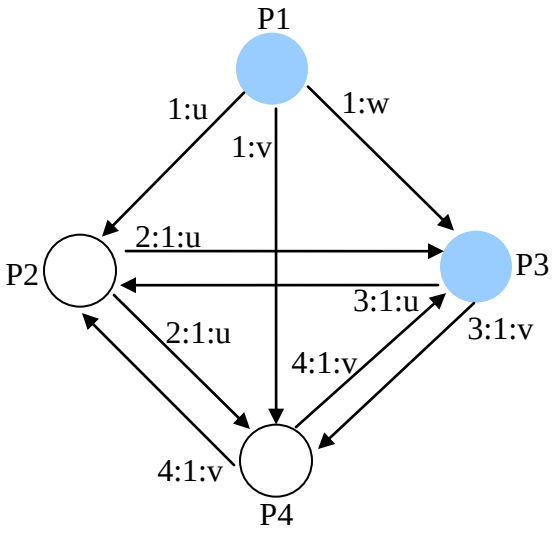
   If the server wrongly concludes that a fault-free client in the critical section has failed, then it will grant a token to another active client in its request queue. In this case, there will be two fault-free clients in the critical section. The algorithm will fail to achieve mutual exclusion in this case.

5. Consider the algorithm for the Byzantine general's problem with 4 nodes (Section 12.5.3 of the textbook, 4th edition). Illustrate how the algorithm may not achieve correct outcome if only two of the nodes are fault-free. You may show the scenario using a figure (such as figure 12.20 in the textbook).

Blue circle indicates a faulty process. White circle indicates a non-faulty process.



P1 sends v. P2 thinks that there is no majority.



P2 thinks that P1 sent u. P4 thinks that P1 sent v.