Homework 3

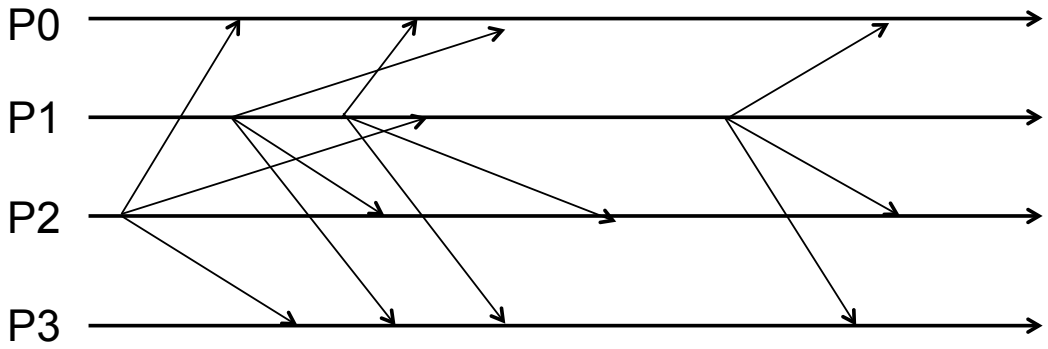
Due: November 20th, 2015 23:59:59pm

Please typewrite your answer to the homework problems, save your file in `.pdf` format, name your file as "your BU email id"-homework3.pdf (e.g., jdoe-homework3.pdf), and submit it to Blackboard before deadline. Late homework will not be accepted.

Problem 1. (10 pts)
Compare the following five consistency models: eventual consistency, linearizability, causal consistency, FIFO consistency, and sequential consistency. Among these:
(a) Which is the "fastest" (most available) models among these? (3 pts)
(b) Which are the two "most consistent" models among these? (3 pts)
(c) What is the difference between the two most consistent models among these? (4 pts)

Problem 2. (Chapter 7 Problem 17 in Tbook) (10 pts)
Consider a non-blocking primary-backup protocol used to guarantee sequential consistency in a distributed data store. Does such a data store always provide read-your-writes consistency?

Problem 3. (10 pts)
Given the following multicast timeline, assume that the processes use the FIFO Ordering multimcast algorithm, mark the timestamps at the point of each multicast send and each multicast receipt. Also mark multicast receipts that are buffered, along with the points at which they are delivered to the application.



Problem 4. (10 pts)
Repeat Problem 3 with the causal ordering multicast algorithm.

Problem 5. (10 pts)
Repeat Problem 3 with the total ordering multicast algorithm using a sequencer. Assuming that the sequencer receives the multicast instantaneously after the multicast is sent.

Problem 6. (15 pts, 3 pts each)
Consider a key-value store that uses quorum for consistency. The total number of replicas, N, for a key, is fixed. However, N may be different for different keys. Each read has to access at least R replicas, while each write has to write to at least W replicas. For each of the following design choices, say whether it (by itself) does or does not prevent write-write conflicts. For each case, briefly explain why or why not. (You can assume there is no caching.)
(a) R + W = N

(b) W = N
(c) R + W > N/2
(d) R + W > 3N/2
(e) R + W > 3N/2, W > 2N/3

Problem 7. (Chapter 18 Problem 13 in Cbook) (5 pts)
In a gossip system, a front end has the vector timestamp (3, 5, 7) representing the data it has received from members of a group of three replica managers. The three replica managers have vector timestamps (5, 2, 8), (4, 5, 6) and (4, 5, 8), respectively. Which replica manager(s) could immediately satisfy a query from the front end, and what would the resultant timestamp of the front end be? Which could incorporate an update from the front end immediately?

Problem 8. (20 pts, 5 pts each)
Two transactions T1 and T2 have instructions as follows (a and b are variables at the server):

T1: write(a, foo, T1); read(b, T1); read(a, T1);
T2: read(b, T2); write(b, bar, T2); write(a, baz, T2);

For each of the following interleavings, explain if it is serially equivalent and why:
(a) write(a, foo, T1); read(b, T1); read(b, T2); write(b, bar, T2); read(a, T1); write(a, baz, T2);
(b) read(b, T2); write(b, bar, T2); write(a, foo, T1); read(b, T1); write(a, baz, T2); read(a, T1);
(c) write(a, foo, T1); read(b, T2); write(b, bar, T2); read(b, T1); write(a, baz, T2); read(a, T1);
(d) read(b, T2); write(a, foo, T1); read(b, T1); read(a, T1); write(b, bar, T2); write(a, baz, T2);

Problem 9. (10 pts, 5 pts each)
Given the following real-time execution ordering:

W1(x, 3), R2(x), R3(x), W2(x, 2), R1(y), W2(y, 1), R3(y), R2(z), R3(z)

where W2(x, 3) means that transaction 2 sets the value of x to 3. The original values of x, y, z are 4, 5, 6, respectively and the transaction number serves as the transaction's timestamp. Assume a transaction commits immediately after its last operation, show how the following concurrency control schemes affect the ordering of the reads and writes in the execution history, and the value returned from each read operation.
(a) Strict 2PL locking
(b) Timestamp ordering
Assume that if a transaction aborts, it will be restarted as a new transaction with its reads and writes added to the end of the execution history. The new transaction will be assigned the next available transaction ID.