# CSE 486/586 Distributed Systems
# Midterm
# Wednesday, 3/6/13

**DIRECTIONS**

- Time limit: 45 minutes (3:05pm - 3:50pm)
- There are 50 points.
- This is a closed-book, no calculator, closed-notes exam.
- The only exception is your cheat sheet (double-sided, letter-sized).
- You should turn in your cheat sheet as well as your answer sheets at the end of the exam.
- Each problem starts on a new page.
- Please use a pen, not a pencil. If you use a pencil, it won't be considered for regrading.
- Each problem also explains its grading criteria. "Atomic" means that you get either all the points or no point, i.e., there are no partial points.

| | |
|---|---|
| Name: | |
| UBITName: | |

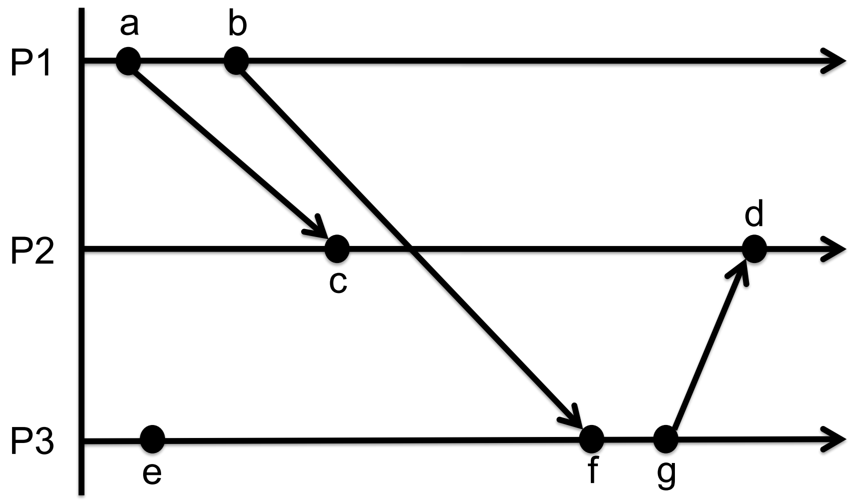
| Problem # | Score |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |

1. (a) What is the general definition of safety?
(**Grading**: atomic 3 points)
**Answer:**
A guarantee that something bad will never happen.

(b) What is an asynchronous system?
(**Grading**: atomic 3 points)
**Answer:**
In an asynchronous system, there is no bound on message propagation time and execution time.

(c) What is a synchronous system?
(**Grading**: atomic 3 points)
**Answer:**
In a synchronous system, there are known bounds on message propagation time and execution time.

2. Consider the following diagram for processes P1, P2, and P3.

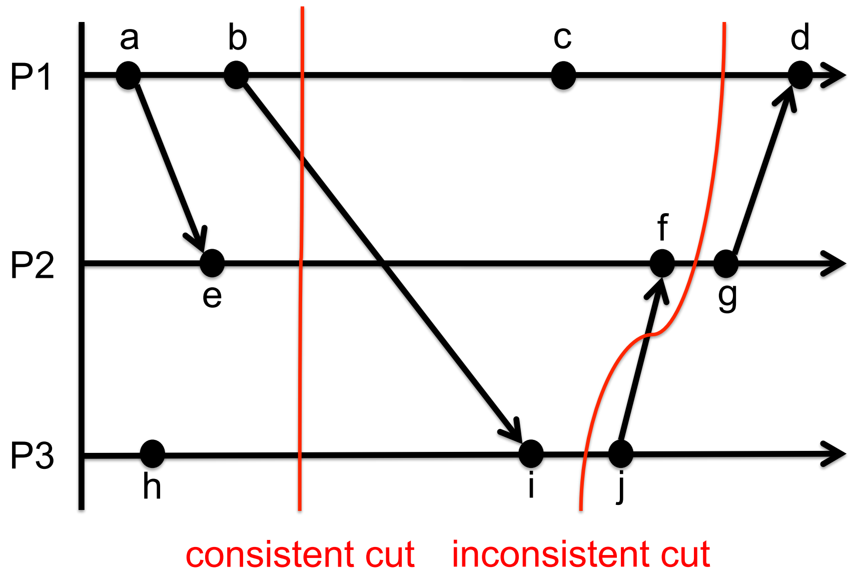



(a) Use the vector clocks to set timestamps for all the events. Assume that all clocks are initially set to (0, 0, 0).
(**Grading**: 7 points)
**Answer:**

| Event | Vector Clock |
|---|---|
| a | (1, 0, 0) |
| b | (2, 0, 0) |
| c | (1, 1, 0) |
| d | (2, 2, 3) |
| e | (0, 0, 1) |
| f | (2, 0, 2) |
| g | (2, 0, 3) |

(b) Look at events *c* and *g*. Can you determine whether they are concurrent or one happens before the other using Lamport's clock? Explain briefly.
(**Grading**: atomic 3 points)
**Answer:**
They are concurrent, but we cannot determine that with Lamport's clock. Using Lamport's clock, event *c* is 2, and event *g* is 4.

3. Consider the following diagram for processes P1, P2, and P3.




(a) Illustrate one consistent cut. Use the figure above to draw the cut.
(**Grading**: atomic 3 points)
**Answer:**
One example is illustrated above.

(b) Illustrate one inconsistent cut. Use the figure above to draw the cut.
(**Grading**: atomic 3 points)
**Answer:**
One example is illustrated above.

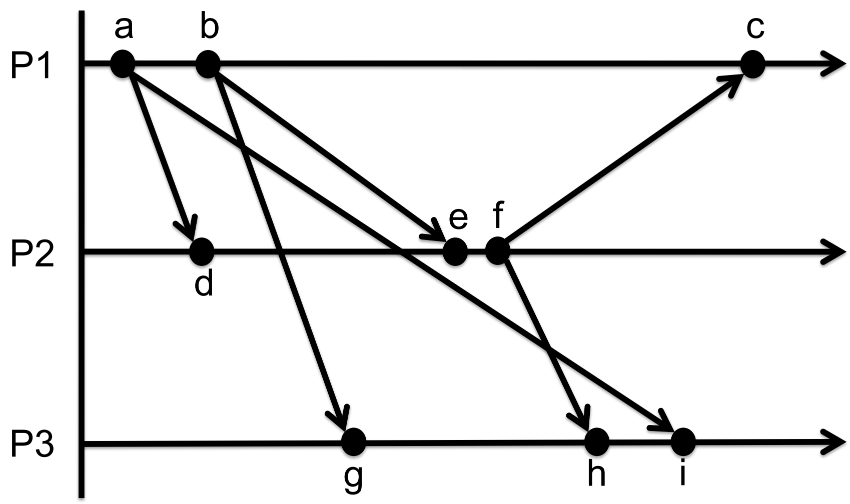
4. Assume that four processes communicate with one another with causal ordering. Their current vectors are shown below.

| | |
|---|---|
| Process A | (3, 5, 2, 1) |
| Process B | (2, 5, 2, 1) |
| Process C | (3, 5, 2, 1) |
| Process D | (3, 4, 2, 1) |

If process A sends a message, which process(es) can deliver it immediately? Why?
(**Grading**: atomic 4 points)

**Answer:**
Both A and C can deliver the message immediately.

5. Using the multicast algorithm that provides FIFO ordering discussed in class, mark the timestamps (using sequence vectors) 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. (**Grading**: 9 points)




**Answer:**

| Event | Sequence vector | Action |
|---|---|---|
| a | (1, 0, 0) | send |
| b | (2, 0, 0) | send |
| c | (2, 1, 0) | accept |
| d | (1, 0, 0) | accept |
| e | (2, 0, 0) | accept |
| f | (2, 1, 0) | send |
| g | (2, 0, 0) | buffer |
| h | (2, 1, 0) | accept |
| i | (1, 0, 0) | accept |
| | | accept (2, 0, 0) |

6. Assume that there are four processes, P0, P1, P2, and P3, running Maekawa's algorithm for mutual exclusion. The voting sets are as follows:

| Process | Voting Set |
|---|---|
| P0 | P1, P3 |
| P1 | P0, P2 |
| P2 | P1, P3 |
| P3 | P0, P2 |

Using the four processes and their voting sets, describe a concrete scenario (i.e., a step by step action sequence) where Maekawa's algorithm does not provide liveness.
(**Grading**: 12 points)

**Answer:**
One scenario is as follows:

| Step | Action |
|---|---|
| 1 | P0 and P2 each send a request to their voting set concurrently. |
| 2 | P0's request gets delayed to P1, but arrives at P3. |
| | P2's request gets delayed to P3, but arrives at P1. |
| 3 | P1 votes to P2 and P3 votes to P0. |
| 4 | P0's request arrives at P1 ad P2's request arrives at P3. |
| 5 | P1 puts P0's request into its queue and P3 puts P2's request into its queue. |
| 6 | P0 and P2 cannot proceed and the system is deadlocked. |