Problem Set 4, Part b

Due: Thursday, November 5, 2015

Readings:

Chapter 18. Lamport's "Time, Clocks,..." paper. Mattern paper. Chapter 19.
Next week:
Chapter 9 (skim). Chapter 10. Chapter 11 (skim).

Problems:

6. Consider an asynchronous system consisting of n user automata $U_1, \ldots, U_n$ and n server automata $S_1, \ldots, S_n$, where the servers can all communicate with each other via reliable FIFO channels. Each U_i can submit requests to S_i of the form $req_i, 1 \leq i \leq n$, and S_i should respond to each request with a "ticket number", which is a positive integer. Assume that each U_i waits for a response to its previous request before submitting another request.

   Design a protocol for the S_i that ensures that, in every fair execution of the system, each request is eventually assigned a distinct positive integer. Moreover, in any fair execution of the system in which the combination of all the U_i automata submit infinitely many requests (although some of the U_is may submit only finitely many requests), your algorithm should assign every positive integer to some request.

7. The Mattern paper describes a distributed algorithm that associates "weak logical times" with events of an underlying algorithm A, by maintaining and sending around vector timestamps.

   Recall the following definitions: A "point" for process i in an execution is a position between two consecutive events of process i in the execution, and is specified by a natural number representing the number of previous events at process i. A "cut" in an execution is a vector of points, one for each process. For cuts C, C', we say $C \leq C'$ if, for each i, $C(i) \leq C'(i)$. We say $C < C'$ if $C \leq C'$ and $C(i) < C'(i)$ for at least one i.

   Now consider a cut C, and let V_i be the timestamp vector of process i at point $C(i)$. Define a new cut V such that $V(i) = \max(V_1(i), \ldots, V_n(i))$ for each i. We then say that cut C is "consistent" iff $\forall i : V(i) = V_i(i)$.

   (a) Describe how to use Mattern's algorithm to solve the Maximal Consistent Cut problem, defined as follows: Initially, all processes receive the same cut C (that is, a vector of natural numbers representing a number of events at each process) as input. C is not necessarily consistent. Each process i is required to return its own entry $M(i)$ in a maximal consistent cut $M \leq C$ of the execution of A. "Maximal" here means that there should not be another consistent cut M' such that $M < M' \leq C$. Prove that your answer is correct, that is, that the M it produces is in fact consistent and maximal.

   (b) Describe a possible use for a solution to the Maximal Consistent Cut problem.

8. Exercise 19.5. Consider an algorithm A that begins in a quiescent global state (as does a diffusing algorithm) but that is used with an environment that can submit inputs at any number of locations (one per location). Design an algorithm to detect when A reached a quiescent global state. Now we say that termination is detected when *done* outputs are performed by all processes that have received inputs from the environment.

9. Exercise 19.13.

   Consider a collection of processes, each of which might be waiting for some of its neighbors. That is, each process has a fixed local value *wating-for*, indicating the set of neighbors for which that process is waiting.

   (a) Design (i.e., give precondition-effect code for) a distributed cycle-detection algorithm for this collection of processes. Your algorithm should determine whether or not there is a cycle of two or more processes, each waiting for the next in the cycle, with no messages en route from any process to its predecessor in the cycle.

   (b) Prove that your algorithm is correct and analyze its complexity.

   (c) Show how your algorithm can be used to detect deadlocks in an underlying asynchronous algorithm A, according to the problem description in Section 19.2.3.

10. Keep working on your project.