Problem Set 4, Part a

Due: Thursday, November 3, 2011

Readings:
Chapter 18, Lamport’s “Time, Clocks,...” paper, Mattern paper
Chapter 19

Next week: Chapter 9 (skim); Sections 10.1-10.8 in detail; 10.9 (skim).

Problems:

1. Exercise 18.5.
   You may use pseudocode for this one, but of course you can use Tempo if you like.
   For part (c), you may just describe the mechanism for choosing an appropriate logical time $t$ informally, without including it in the pseudocode. That is, you can write the pseudocode so that it just fails if the chosen $t$ doesn’t work.

2. 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 from class: 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 fix 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 $\forall i : V(i) = V_i(i)$.

   Describe how to use Mattern’s algorithm to solve the “maximal consistent cut” problem, defined as follows:

   After algorithm $A$ has been executing for a while, each process receives the same (not necessarily consistent) cut $C$ of the current execution of algorithm $A$ as input. 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 $M < M' \leq C$.

3. Exercise 19.4. Use Tempo to write the combined algorithm.

4. Exercise 19.11.