Problem Set 6, Part b

Due: Thursday, December 1, 2005

Reading:

Borowsky-Gafni-Lynch-Rajsbaum paper. Attiya-Welch, Section 5.3.2 (optional).

Reading for next week:

Chapters 17 and 21. Lamport's "The Part-Time Parliament". Begin Chapter 22.

Problems:

1. In class, we described a simplified version of Herlihy's universality construction for wait-free consensus objects. It implemented an arbitrary n-process atomic object using an infinite sequence of n-process consensus objects to decide on successive operations, and an "announce" array of read/write registers to ensure fairness.
   Describe (in clear English) a mechanism that you can add to this algorithm to try to achieve a good time upper bound for all operations. Try to achieve $O(n\ell)$, where ℓ is an upper bound on process step time.

2. As noted in class, the BGLR paper has a liveness bug in the main protocol. Namely, a simulating process i may repeatedly decide to select the same process j to perform a snapshot, using safe-agreement, neglecting some other process j'.

   (a) Why doesn't the task structure of process i, which has a separate task for each simulated process, ensure progress for all the simulated processes?

   (b) Give a simple modification to the given code that would fix this problem, and guarantee that all the simulated processes get fair turns.

3. Consider the *approximate agreement* problem, expressed as a decision problem as follows: The value domain V is the set of rational numbers. For any input vector I of elements of V, the allowable output vectors are those for which (a) every element is in the range of the values in I, and (b) the difference between any two output values is at most one.
   Suppose we are given a 10-process, 2-fault-tolerant asynchronous shared memory algorithm A that solves approximate agreement, using read/write shared registers. Describe clearly how we can use algorithm A and the BG-simulation results to obtain a 3-process wait-free asynchronous shared memory algorithm to solve approximate agreement, again using read/write shared registers.