Massachusetts Institute of Technology
6.852: Distributed Algorithms
Prof. Nancy Lynch

Handout 17
Thursday, November 19, 2009

Problem Set 6, Part a

Due: Thursday, December 3, 2009

Reading:

Chapters 9 and 18 from the Herlihy, Shavit book. (See course web site.)

Reading for next week:

Herlihy paper on wait-free synchronization
(Optional) Attiya, Welch, Chapter 15.

Problems:

1. Prove that the hand-over-hand locking algorithm for a list-based set guarantees atomicity and does not deadlock.
2. Suppose we change the hand-over-hand algorithm to use reader/writer locks so that an operation always acquires a lock for reading before reading a node (even if only to read the key), and then if it needs to write a node, it "upgrades" the appropriate lock before doing so. Prove that this algorithm is deadlock-free or give a counterexample execution.

   NOTE: A reader-writer lock can be held either for reading or for writing; the mode is specified when the lock is acquired. Any number of processes may hold a lock for reading, but when a process holds a lock for writing, no other process may hold the lock for reading or writing.

   In addition to acquire and release operations, a reader-writer lock provides an upgrade operation, which can be invoked only by a process that holds the lock for reading. A process that invokes upgrade waits until no other process holds the lock and then atomically changes its mode from reading to writing.
3. Give an execution for the lock-free lazy list algorithm with wait-free contains in which a contains operation cannot be serialized at or after reading the next pointer of the last node it accesses whose key is less than the key it is searching. Can you do this for a *contains* operation that returns true? Can you do this for the lock-based lazy list algorithm?
4. For the obstruction-free software transactional memory with invisible readers, give an execution in which a committed transaction cannot be serialized at the point that it changes its status from "active" to "committed".