CSE 490h/CSE M552
Project 4
Due: 5pm, Monday, February 28, 2011

In this assignment, you are to add high availability to your transactional, cache coherent distributed storage system, using the Paxos algorithm. After assignment 3, clients can proceed if any other client fails by asking the server to revoke any exclusive access cached files, but if the server fails, all transactions must stall until it recovers. With this assignment, your system should support linearizable client commits, even though any single server node has failed.

Instead of a single server, Paxos spreads the server function across a static set of nodes. During testing, for simplicity we advise you to use a separate set of nodes for clients and for servers, although there is no logical need for a distinction. The set of servers use Paxos to decide on the order in which to commit transactions; Paxos tolerates individual node failures, allowing progress even if any server in the group has failed. When the failed server restarts, your code should re-integrate it into the group to re-establish fault tolerance to individual server failures.

The turn in instructions are the same as in assignment 2. You should draw the state machine you expect, run Synoptic on your implementation, and explain any differences.

In your writeup, please also address the following design questions:

a) In your design, how is callback state maintained? Do the servers vote on changes to the callback state using Paxos, do they elect a leader to centralize the callback state, or do you use some other mechanism? Is it even necessary for correctness or performance that the server callback state be consistent with the state of the client caches?
b) In your design, how is storage state maintained? Does every server participating in Paxos store every modification of every file, do they elect a primary/backup, or do you use some other mechanism?