# CSE 490h/CSE M552
# Project 3
# Due: 5pm, February 11, 2011

In this assignment, you are to add transactional fault tolerance to your distributed storage system.  Clients can group operations together, e.g., to update a friend list for two separate people; the operations commit together or not at all. In the case of an inopportune failure, the updates are rolled back to the state before the transaction started, and the client can simply retry the transaction.

Transactions provide a clean way to deal with failures in the presence of cache coherence. A client modifying some data can acquire exclusive access permission to the various files that need to be updated, update them, and then commit. Once the updates and the commit are recorded at the server, the transaction is done, and that can be reported back to the user. If a client does not reply to an invalidation request (because it has crashed or just appears to have crashed), the server can unilaterally revoke the exclusive access, and allow another client to proceed. When the original client tries to commit, the server can then tell the client that their transaction has failed and that they must restart.

For this assignment, you only need to handle one transaction at a time per client, so the user commands you will need to add are “txstart”, “txcommit”, and “txabort”, with no arguments. (To support multiple concurrent transactions on a single client, txstart would need to return a transaction ID, which could then be used as an argument in the various file operations.)

The turn in instructions are the same as the previous assignment, except that there is no need to run Synoptic, as the client/server protocol should be straightforward.