Jacob Sanders, Terry Hampton

CSE 452 project 2 writeup

In order to accomplish ACID properties in transactions, we wrap all of the twitter operations with a RPC\_TXN\_START and RPC\_COMMIT remote procedure calls.

* When the server receives a RPC\_TXN\_START call, it creates a new, unique transactionID based on the current sequence number on the server, and creates a new log in memory for that node. It then returns that tid in reply to the client. The requestID which requested a new transaction is also recorded, so that RPC\_TXN\_START requests are idempotent (receiving multiple RPC\_TXN\_START requests on from the same request ID will return the same transaction ID)
* If the server receives a request that is not RPC\_TXN\_START with a transaction ID that differs from the one currently on the server, an RPC\_ABORT message is automatically sent.
* Once a transaction has been established, all subsequent read/delete/append requests are stored in the log, ordered by their requestID. The state of files on the server is not altered until a valid commit is received.
* Whenever an operation is appended to the log, the server will iterate over the log and verify that the files on disk have not had their state altered since the transaction was started (i.e. the file exists and the version of the file <= the transactionID). If the state is inconsistent, an RPC\_ABORT is sent.
* When the server receives an RPC\_COMMIT, it performs the same check, and if the transaction can be committed, it iterates over all of the changes and compiles the final state that all of the files should assume. It then writes these states to the write-ahead log, replies to the user that the commit was a success, and finally performs the actual writes.
* If a client receives an RPC\_ABORT command from the server, the startTxn-operations-commit set of commands is restarted, and repeats until a successful commit is made.

This means that a transaction will commit and persistent state will be modified iff all of the files that were accessed during that transaction are in the state they were at the time that the transaction started.

In our implementation, we assume that nodes numbered 0-127 are servers and 128-254 are clients.

On top of that, we did some refactoring and re-organizing of our code from project1 which should hopefully make your jobs easier.