Jake Sanders

Hampton Terry

CSE 452 Project 3+4 writeup

The way

We initialize the current **round** to 0 whenever a paxos or server node first initializes. This has many useful purposes. This means that the first request that any node makes will always conflict with the very first action undertaken by the system, which greatly simplifies startup/catchup. This allows a paxos node to simply return the value learned for a given round if a request is received relating to that round. This also means that a given machine cannot leapfrog over updates, since the **round** that a given machine believes it to be is exactly 1 more than the last update that machine has received. This is pivotal in our implementation of project 4, since this essentially blocks new paxos/server nodes from harming the state of the system after they initially start up, while providing a convenient mechanism to automatically detect tardiness.

In order to bootstrap a paxos group, call “n startPaxosGroup a b c” on a node after it assigning it the paxos role. This explicitly sets the nodes that should be considered in the paxos group. At the very least, you should use “n startPaxosGroup n” to bootstrap the creation of the initial group. From there, you can use “m joinPaxosGroup n” to have additional nodes join n’s group.

Once you have a paxos group set up, you can use the “n paxosLearn (value) (**round**)” command to have the specified node act as a proposer, accepting your value/round request as it would a server’s, and running the paxos protocol to attempt to reach consensus.

You can use the “clingToLife” command on a node in order to get it to start a callback with spins until you call “embraceDeath” on that same node. This extends the life of the simulation until a point of your choosing.

You will probably need to eliminate all of the saved state on disk between executions if you wish to get a clear picture. On the plus side, the contents of all of the files are stored as plain text (json), so they’re a convenient way to know the state of a file/paxos instantiation.

After nearly 40 hours of work myself on these two projects, and with other deadlines looming, I know when I need to call it quits. This final stage has defeated me. I wish that either the labs had been assigned in reverse order (to be bottom-up instead of top-down), or that we has spent a lot more time designing a truly maintainable codebase (probably both). Either way, integrating Paxos into our existing code has proved to be an insurmountable task. We can only hope that the scripting hooks I have given you guys to prove that the paxos implementation and the code to join nodes to/ remove them from an existing paxos group is solid, and that you will bestow your mercy upon us.

Thanks for a very interesting quarter.