CS425/ECE 428 | Distributed Systems | Spring 2013
Machine Problem 2 | Replicated State Machines

Stan Li, David Huang (sitanli1, huang157)

General Approach

For this MP, we decided to use passive replication, meaning that the frontend would talk to only one of the replication managers and then that replica manager will talk to the rest of its group. To create a state machine, the frontend pings all replica managers to find the one with the lightest load. It sends a create request to that manager. Then, the receiving manager does a similar search amongst its peers and sends a similar create request, along with a list of all 3 managers that will maintain the given state machine. Apply operations work in a similar manner.

This design was chosen for its inherent total ordering. The primary manager acts as a sequencer to enforcing proper ordering. In the event of its failure, the three managers are arranged in a queue and the next manager is promoted to be the new primary. This is implemented by simply selecting the manager with the next highest ID.

Replication

On creation, the primary manager is contacted by the frontend with a create request. It is then responsible for finding two other managers with the lightest load and repeating this request to them. All managers maintain a list of other managers that are hosting the same state machines. A separate list is kept for each individual state machine at each manager.

Writes are done similar to creates by repeating the request to all managers. Upon receipt, each manager simply applies the operation to its local state machine. A flag is sent along with each repeated request to identify that it is from the primary manager and therefore should not be repeated. Reads are simply done by contacting the primary manager of a state machine. The write and create protocols guarantee consistency so it's not necessary to contact all of the managers. Synchronization is accounted for by the use of a mutex on each replica manager. The mutex is locked whenever an operation is done that may change any of the locally stored state machines.

Load Balancing

We designed our state machine creation scheme with load balancing in mind. Before telling the back-end to create the state machine, the front-end will ask all replica managers how many state machines they are currently hosting. We then pick out the most lightly loaded (the one with the least number of state machines hosted) replica manager and issue the request to that manager. That replica manager then creates a replica of the state machine and repeats the create requests to next two most lightly loaded replica managers, along with a list containing the identifiers of all of the involved managers. This guarantees that we distribute the state machine replicas during creation.

Failover and Recovery

Failures are detected whenever a frontend request is issued. The receiving replica manager will verify that all three managers in the group are still functional by pinging them and waiting for an exception. When an exception occurs, the failed manager will be removed from the group and the next most lightly loaded manager will be brought in via a create request (using the current state of the machine in question). This will occur until there are three replicas of the state machine once again. This approach not only ensures the recovery of the state machine, but it also reassigns state machines with load balancing in mind.