Distributed Systems and Algorithms CSCI-4963/6963

Project 2

Names: Zexin Wan, Wenyin San

Term:   Fall 2016

description of your project design, including how you handle network configuration, initialization, failures, and recovery.

**Introduction**

This project is the implementation of a highly available distributed service in Python for synchronizing access to small files. In a system of N processes, each process enables a user to perform four operations on a given file: create, delete, read, and append. The project design is based on ZooKeeper, a distributed coordination service that is part of the Apache Hadoop project.

**Project Design**

We have used Zookeeper Atomic Broadcast (zab) to update the state at all servers.

3 kinds of roles in ZooKeeper:

Server:

1. Leader
2. Learner: Follower

Client

1. Client

Model:

Leader: accepts all requests

if req = read, read

else write/create/delete and forward update

Follwer:

if req = read, read

else forward req

accepts uploads from leader

Election Algrorithm : Bully

Server with largest zxid becomes the leader

At failure to connect to leader:

the server sends election message to each followers

if a server has a higher zxid

it replies with an election message

then sends another election message

else if no server replies then server becomes the leader

and then sends out an elected message

In order to simulate a system of distributed applications, we used the Amazon EC2 service. 5 (Amazon Linux AMI / Ubuntu) t2.micro-instances were created in separate regions: