CPSC 416: Distributed Systems, 2013 W2

Assignment #4, Due Monday (midnight), April 7th, 2014

Introduction

Apache ZooKeeper is a distributed open-source coordination service for distributed applications. It is widely used by companies such as Yahoo, eBay and others as a scalable, fault-tolerant framework upon which to build distributed applications. The objective of this assignment is to give you hands-on experience with Zookeeper by running a leader-election application.

Getting Started

Get familiar with Zookeeper by reading overview:
http://zookeeper.apache.org/doc/trunk/zookeeperOver.html

Now set-up Zookeeper
http://zookeeper.apache.org/doc/r3.3.4/zookeeperStarted.html

Now try the sample leader-election application on Zookeeper as described in the following tutorial
http://cyberroadie.wordpress.com/2011/11/24/implementing-leader-election-with-zookeeper/

When you clone Git repository given in the tutorial, your working folder will be called "zookeeper-leader". It should contain "src" folder and "pom.xml" file. From this folder build this application as "mvn clean install" and execute several instances of this application (in a separate terminal) using the following command:

```
mvn exec:java -Dexec.mainClass="net.spike.zookeeper.SpeakerServer"
-Dexec.args="message_for_instance1 3000"
```

You can look at the constructor of SpeakServer for a description of the arguments. Briefly, they are used to indicate the following:

- message_for_instance1: can be changed for each instance;
- 3000: frequency of the message output (in milliseconds);

Once you have successfully executed the previous command, observe that a "out.txt" file was created in the main folder. Amongst several instances, only one will be elected as a leader and the contents of "out.txt" is being constantly updated with the leader's message and counter.

For example, if you execute three instances, they will have process IDs in following format:

pid-12345.00001, pid-12346.00002, pid-12347.00003.

You can check them by executing “ls /” command in Zookeeper’s command line tool (zkCli.sh). In this sample application, instance with the smallest ID is elected as a leader and will get the right to update "out.txt" file. In our example, the content of the file will be as "message_for_instance1: {counter_integer} pid-12345.". If the instance with smallest ID is terminated, then the next instance with the smallest ID is elected. Observe this behavior for different scenarios. Try to understand source code of this application.

Assignment

1) Change the application so that instance with highest ID gets elected as a leader.

2) Observe that instance IDs are allocated sequentially in an ascending order, such as pid-12345.00001, pid-12346.00002, pid-12347.00003 respectively. Change the application so that middle part of the instance ID is a random number between [0..1000] exclusively, and the leader is the instance with the largest random number.

For example when three instances are executed, they will have pid-521.00001, pid-667.00002, pid-238.00003 respectively and leader pid-667.00002 writes to the file. If this instance crashes, pid-521.00001 will be elected as the next leader and writes to the file.

Note that in order to successfully compile the application after a change, you need to change the test suites too. They are located at *src.test.java.net.spike.zookeeper.NodeMonitorTest.java* file.

Deliverables:

Archive your entire "zookeeper-leader" folder as "assign4.zip" and submit via handin.

Good luck!