**Broker-Based Fault-Tolerant Publish-Subscribe Using ZMQ, Mininet, and ZooKeeper**

In this assignment we will be able to reuse most of the code from assignment 1. The big difference is that instead of directly going to the broker node, each entity will now consult ZooKeeper. Publishers and subscribers ask for the broker node from ZooKeeper. Join and leave of entities is now handled via ZooKeeper using watch mechanisms. Moreover, you could have more than one broker and when that happens the brokers participate in a “leader election” using ZooKeeper. The publishers and subscribers will also get hold of only the leader broker. If broker leadership changes, these entities will need to know the change. Any entity may fail: a publisher, subscriber, a broker. As before, publishers and subscribers can come and go at any time. All these failures must be handled (possibly via ZooKeeper). It is possible that during recovery, some samples published by the publishers are lost. Understand these implications.

Everything else is same as before i.e., we will still use Approach #1 and Approach #2 for the dissemination part, and the experiments are the same. The assignment continues to be team-based. The specification is reproduced below.

Discovery of who the current leader broker is, is handled via ZooKeeper support. Watch mechanisms are to be used wherever needed. The broker is still the matchmaker; no matchmaking happens via ZooKeeper.

Dissemination continues to use both approach #1 and #2 as in Assignment 1. This remains a configurable parameter.

**Assignment-Specific Demonstration:**

* Should work with all the properties mentioned in the assignment but this time using ZooKeeper instead of using the intermediate broker as the manager of information.
* Should work with multiple publishers publishing multiple different publications (i.e., topics) and multiple subscribers, all of them distributed over different hosts over different kinds of network topologies that we can create in Mininet (instructor may provide their topologies).
* Publishers and Subscribers may join any time and leave any time.
* Any of the mentioned entities may fail at any time
* Do end-to-end measurements (time between publication and receipt of information; since the clock is the same on all emulated hosts, we do not have the issue of clocks drifting apart from each other). Plot graphs and have them saved in the repo that you maintain.
* **Optional** (but try for your own sake): ZooKeeper itself can run in an ensemble i.e., it itself is fault tolerant. So it is possible that some zookeeper server may die. The ensemble will take care of fault tolerance.

**Rubrics: Total 100 Points**

Please refer to the generic programming assignment grading guidelines for the grading rubrics. Thus, ensure that your assignment fulfills all the requirements as specified and meets the rubrics requirements including the README.

**Feedback to Instructor**

The “grading team” should send a detailed report to the instructor on how the submitted assignment met the grading requirements and grade received by the submitting team along the specified rubrics they used. They should also get a sense of the effort put in by each team member.