Homework 4: A Scalable and Highly Available Twitter Service

200pts + BONUS 50pts (if Paxos is used)
Due: April 14th, 2017, at 11:59pm

1 Overview

The objective of this assignment is to develop the next version of the Twitter service (aka Chat Room Service with followers) that is scalable to a large number of users with significantly more workload, is fault tolerant and highly available (i.e., failures in the system are handled transparently to the user. A stress test client is provided to you, as part of this assignment. You are not required to use the provided client, but you are required to follow the provided gRPC interface. For this assignment the following must be considered:

1. The Chat Room Service functionality that was provided in Homework #2 is still required for this assignment. Thus, it might be wise to start with the provided solution for that homework.

2. Your service will execute on 3 servers/machines. Your system should contain three startup scripts that start your system.

3. You can assume that the server/machine on which the master process runs, is always available, although processes on it may still crash (i.e., killed by us).

4. The failure of ANY process in the system is possible and you will need to take this into account.

5. Data that has been stored persistently on hard disk will not be corrupted, i.e., you can rely on its accuracy.

6. You can not assume that the servers are time synchronized, i.e., we will modify the clocks of the servers.

7. Your implementation should start at most 10 processes.

8. We will "kill" at most 1 process within any 30seconds time window. Thus, we will not attempt to crash a second process immediately after

crashing a first process. We will wait 30 seconds. You can use this time interval to restart the crashed process.

9. You should not assume the communication (i.e., physical layer or the connection between servers) is reliable. We may disable the network interface from at most 1 server at a time. Your system should be able to recover

2 What to Hand In

Build your system starting with provided solutions for Homework #2 and #3.
Re-run the performance evaluation you have performed in Homework #3 Item 4 (when clients and followers are on different machines). Evaluate performance when 1 process is killed (this process can be the master, if you have one, or slave).

2.1 Design

Start with the provided code for HW2. Based on your design, you may find that significant portions of the provided code are no longer needed. Feel free to start from scratch with coding as well, but abide by the gRPC and protobuf interfaces.
Before you start hacking away, write a design document. The result should be a system level design document, which you hand in along with the source code. Do not get carried away with it, but make sure it convinces the reader that you know how to attack the problem. List and describe the components of the system.
If you use Paxos, you are eligible for 50pts bonus.
For this assignment, we WILL stress test your code. Stress testing will include, but not limited to: process failure, communication failure, time synchronization problems.

2.2 Source code

Hand in the source code, comprising of a makefile, source code files and startup scripts for starting your system on each of the 3 servers your system will run on.
The code should be easy to read (read: well-commented!). The instructors reserve the right to deduct points for code that they considers undecipherable.