Carlo De Guzman, Leon Wallace

HW 3.1 Design Document

Routing Server

The routing server is built off of its own gRPC service, with three different functions defining its interactions with message clients and servers. The routing server keeps track of all master servers which are connected to it along with the current server designated as the available server; the master servers are stored in the form "IP-address:port" so that it can be directly sent to clients without any assembly on their side.

- rpc AddMaster(Message) returns (Response) {}
    - This function represents communication between a message server and the router server. The routing server saves the IP-address and port of the message server, and sets it as the available server if there is currently none.
- rpc GetAvailable(Message) returns (Response) {}
    - This function is mainly used by the message client to find out where the message server is. The message server also calls this function to determine whether it should stay in standby mode or transition into becoming the available server.
- rpc DualPing(stream Message) returns (stream Message) {}
    - This function establishes a continuous connection between a message server and the routing server. When a message server dies, this allows the routing server to remove it from the list of masters and update the available server as needed.

Master Servers

The initial SNSService is extended to include the following function

- rpc GetFollowerInfo (Request) returns (MapReply) {}
    - Whenever a server is on standby, it requests all active follower information from the available server and saves it in its own client_db; this allows it to immediately take over timeline mode if the current available server fails. For each client, the available server converts the relevant follower and following information into a single string, then stores it in a map to be sent to the standby servers.

Available Server Selection

Since the routing server will never fail in this assignment, it is responsible for selecting the next available server whenever one goes down, which it does by simply choosing the first available master in its list of masters.

Restarting Killed Masters

The message server process is run from the fork of another program, which simply waits on the message server and restarts it on failure. This program is itself run from a Bash script, which restarts that program whenever it fails. As a result, there may be instances where a master is running without the slave monitoring it; however, when that master dies the new master-slave pair will become the main processes.

Clients

- Instead of connecting directly to a message server as in the previous assignment, clients now first connect to the routing server and get the name of an available server before connecting to that available server and processing commands.
- In timeline mode, whenever the available server dies, the client will contact the routing server again and get the name of another available server before reinitializing the stub and resuming timeline mode.