Homework 3.2

A distributed tiny social network service

CSCE 438

Curtis Green - 422008537

Caleb Edens - 822007959

Overview:

Design and create an SNS using gRpc that contains all the features from previous homeworks while being distributed between multiple servers on multiple machines. The clients must be able to connect to the routing server and regardless of which server they are routed to, should be able to send and receive posts to other users.

Server:

The servers have the previous functionality of homework 3 with the addition of every master server being available for clients; this means that the routing server sees who has the least clients connected to it and sends the clients to that server in order to balance the load per server. When the servers receive a follow/unfollow command from the client they broadcast this info to the router which relays this data to the other servers, keeping them up to date. For the timeline mode, each server sets itself up in a linked list manner that is managed by the router; each server has a server that it sends to and a server it receives from. When a message is received from the client it opens up another stream to its designated target to let it know the new messages. This pattern continues to where the servers are connected in a ring formation; messages are continually streamed to the other servers until it makes a full loop. This reduces stress by making the servers only need to send to the client and one other server.

Client:

The client has not received as many changes compared to the server. The only additions to the client was the addition of being able to reconnect to a server in timeline mode without having to close and reopen. This was done by checking on the reader and writer thread connections to the server and monitoring if they are closed. If they are closed the client recalls the server through the connect and routing functions, then creates a new stub and stream and continues its ability to write but on a different server. One issue with this is that the client writer cannot tell if the server has failed until the user sends a message and it fails. There is no non-blocking standard input function in C++ so we are unable to let the reader know the thread has died. This can be easily remedied however, by sending a single message from each user after the client has reconnected.