Luke Grammer / Taylor Williamson

Professor Radu Stoleru

CSCE 438

2/07/20

Homework 2 Design Document

Proto Interface

The interface between the server and client be the TSN (Tiny Social Network) service with 5 remote procedure calls defined to add users, list users, follow and unfollow, and process a user's timeline. Most of the messages passed between the server and the client will consist solely of a username or status, with a few notable exceptions. For example, the follow and unfollow requests will contain the username to follow and unfollow respectively. In addition, a PostMessage message will be defined that contains the attributes of a timeline post, namely a time, poster, and the message contents. This will be the interface that the clients will use to communicate with the server and vice versa.

Server

The server will contain a definition for each of the RPC's defined in the interface. In order to keep track of users, the server will also have a vector of user objects in order to keep track of registered users. Each user will have a status flag indicating if they are active, a username, a timeline, and a list of all the usernames of the users that they follow. In order to add data persistence to the server, most operations will also be logged to files. For example, one file will keep track of the entire list of registered users. In addition, there will be more files for each user that contain the list of users that they follow, as well as their complete timeline history. When the server starts up, it will read the contents of these files and use them to re-initialize the vector of user objects in order to re-create the same state that it had when it shut down. To process the timeline, the server will use two separate threads. One thread will read messages from the user and post them to the timelines of every user following the poster (in addition to writing them to their timeline files), and the other thread will use a semaphore to sleep until a message is posted to the current user's timeline, then wake up and send them off to the client.

Client

The client is much more simple than the server and starts by processing the input arguments using getopt(). After processing these arguments, a client object is initialized and the run_client() function is called. This function will start by connecting to the server and initializing a client stub, then it will collect user input and attempt to parse a command from the input and display the result. The process command function will simply parse the user input and call the appropriate server function through the defined interface. It will then take the results and return them to run_client() as an IReply structure. If the command is 'timeline', the processTimeline function will be called to allow the user to make timeline posts, as well as display timeline posts from followed users, until ^C is entered into the shell. This processTimeline function will also be double threaded. One thread will read input from the user and write it to the server, and the other thread will read input from the server and display it for the user.