MP1: A Tiny Social Network Service (SNS)

150 points

1 Overview

The objective of this assignment is to exercise your knowledge about Google Protocol Buffers and gRPC by building a Tiny SNS, similar in concept with posting and receiving status updates on Facebook or Twitter. In this service, the following must be considered:

1. A user operates in two modes: "command mode" and "timeline mode". When the client program starts, it automatically starts in command mode. The commands the client can send are: FOLLOW, UNFOLLOW, LIST, TIMELINE.
2. **LOGIN**: To see how a user/client login to the SNS, please see the README file and the first test case in TestCasesMP1.xlsx on Canvas. Please note that we don't allow duplicate login, i.e., a user cannot log in twice to the SNS.
3. **FOLLOW**: A user is allowed to FOLLOW any other user after submitting a FOLLOW command. When a user posts an update to his/her timeline, followers can see it on their timelines. The "FOLLOW <username>" command adds the current user to the given user's timeline. After submitting the FOLLOW command, the user sees only new posts. That is, the follower can not see any posts that were posted before the follower started to follow.
4. **UNFOLLOW**: The "UNFOLLOW <username>" command removes the current user from the given user's timeline.
5. **LIST**: The LIST command retrieves from the server the list of existing users and the list of users who follow the current user. In other words, the List command is issued by a user. It outputs all users and the followers of the user issuing the List command.
6. **TIMELINE**: Each user(client) has his/her own "Timeline." The user is allowed to post to his/her own timeline. That is, whenever a new

user appears, the server needs to create a new timeline for the new user. The TIMELINE command switches a user from command mode to timeline mode and allows the user to post some updates to both his/her and followers' timelines. Once a user issues the TIMELINE command, it can not return to the command mode, and it will stay only in TIMELINE mode.

7. **TIMELINE**: After submitting the TIMELINE command, a user u1 enters into the TIMELINE mode and u1 immediately sees the last 20 posts as follows: "username", "posting time", and "actual posting". These 20 posts should be the posts from other users that u1 is following. It's important to note these 20 posts only include the posts that are made by other users after u1 becomes one of their followers.

8. **TIMELINE**: After u1 enters TIMELINE mode, if other users are still posting, u1 must immediately see those posts on u1's timeline. In other words, we are using synchronous streaming and synchronous ReaderWriter in TIMELINE mode for u1 to immediately see posts from other users who are followed by u1.

9. **Server Persistency of Timeline** All timelines on the server side must be persistent, i.e., their contents must be stored in files in the local file system. The format for a posting is the following:

```
T 2009-06-01 00:00:00
U http://twitter.com/testuser
W Post content
Empty line
```

   This format describes the Requirement 1.7 for easier manipulation of subsequent MPs. The Timeline format mentioned in Requirement 1.6 can be your design (e.g., one line containing "username", "posting time" and "actual posting"), which can be different from the format required in 1.7.

All communications will be using Google Protocol Buffers v3 and gRPC. The client and server must be started in the following ways:

```
$ ./tsd -p <port_number>
$ ./tsc -h <host_name> -p <port_number> -u <username>
```

2 Programming Environment

For all Machine Problems in this class you will use Ubuntu installed on VirtualBox/UTM. Using an environment like this will significantly reduce the likelihood of environment-related errors for your submissions (when compared to the environment we will be using to test your code).

To setup the virtual machine(VM), please follow the steps below:

2.1 Mac M1/M2 users

You want to follow the instructions in this video. Here is a list of highlighted steps for Mac M1/M2 users.

1. Download and install UTM.
2. Download and install : Ubuntu 22.04.3 Server for ARM. During the installation, you MUST create a username "csce438". Once it is installed, you can start the VM and log into the machine using the password you set during installation.

   Make sure you give your machine at least 4-5 GB of Memory and at least 4 processors. Also, make sure to give it at least 35 GB of storage.
3. Log into the Ubuntu server. From the command line, execute the following:
   - $sudo apt update
   - $sudo apt upgrade -y
   - $sudo apt install ubuntu-desktop
   - $sudo reboot now
4. After rebooting the machine, you will have a Ubuntu Desktop running. From a terminal, continue from Step 3 in Section 2.2 below.

2.2 Windows and Mac x86

1. Install VirtualBox 7.0 on your machine.
2. Install Ubuntu 22.04.3 AMD64 Desktop in your VirtualBox. Please check the "Skip Unattended Installation" checkbox when you install the VM. During installation, when prompted to "Pick a username," you MUST create a username "csce438". Once it is installed, you can start the VM and log into the machine using the password you set during installation.

   Make sure you give your machine at least 4-5 GB of Memory and at least 4 processors. Also, make sure to give it at least 35 GB of storage.
3. Log into the Ubuntu virtual machine. From the command line, execute the following:
   - $sudo apt update
   - $sudo apt upgrade -y

2.3 All Operating Systems after Steps 2.1 or 2.2

1. Download the provided file ".bash_aliases" and place it in your home directory: "/home/csce438/". It's important to note: after you download ".bash_aliases" from the Canvas, the "." may disappear. Please make sure the file is "/home/csce438/.bash_aliases".
2. Make sure that both "setup-438-env.sh" and ".bash_aliases" are under the directory "/home/csce438".
3. Start a new terminal window and execute:
   - $echo $MY_INSTALL_DIR

     Make sure you see the following output: /home/csce438/.local
   - $mkdir -p $MY_INSTALL_DIR

     Make sure that you are copying the '_' if you are copying and pasting the above commands into your terminal.

4. Make sure you are currently in the '/home/csce438' directory and install the csce438 environment by following the step below:
   - $chmod 755 ./setup-438-env.sh
   - $./setup-438-env.sh

Once the installation is done (Sections 2.1 or 2.2), and the environment is setup (Section 2.3), you can start working on this MP.

3 What to Hand In

The running system will consist of the tiny SNS server (tsd.cc) (and possibly other servers) and the tiny SNS client (tsc.cc). As platform, you should use the provided virtual machine where Google Protocol Buffers v3 and gRPC are installed, and develop the program in C++.

3.1 Design

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: Client/Server Program, and their interaction.

3.2 Grading criteria

The 150pts for this assignment are given as follows: 15pts for complete design document, 15pts for compilation, 120pts for 6 test cases (the test cases have different weights). Refer to provided 6 test cases above which cover most scenarios but these are slightly different with the test cases for grading.

3.3 Source code

Hand in a zip file including a design document and source code (comprising of a makefile, a file tsd.cc, a file tsc.cc, and any auxiliary files, for example, Google protocol buffers). The design document must precisely indicate the commands used for each test case and the terminal output screen shot for each command. If the TA/grader cannot compile/run your code on TA/grader's virtual machine, your score will be either deducted or you have to come to TA's office hours to do an in-person demo.

The code should be easy to read (read: well-commented!). The instructor reserves the right to deduct points for code that he/she considers undecipherable.

We encourage you to submit testing videos (either include them in the zip file or upload them to YouTube). We expect the videos to show 1) the whole process of typing commands on terminals; and 2) the terminal output after typing each command. From our previous experiences, this reduces our grading workload and minimizes the risk that we encounter errors when running your code.

The zip file mentioned above must be submitted through Canvas.