Design Document for SNS (Social Networking System)

Pavitra Malhotra

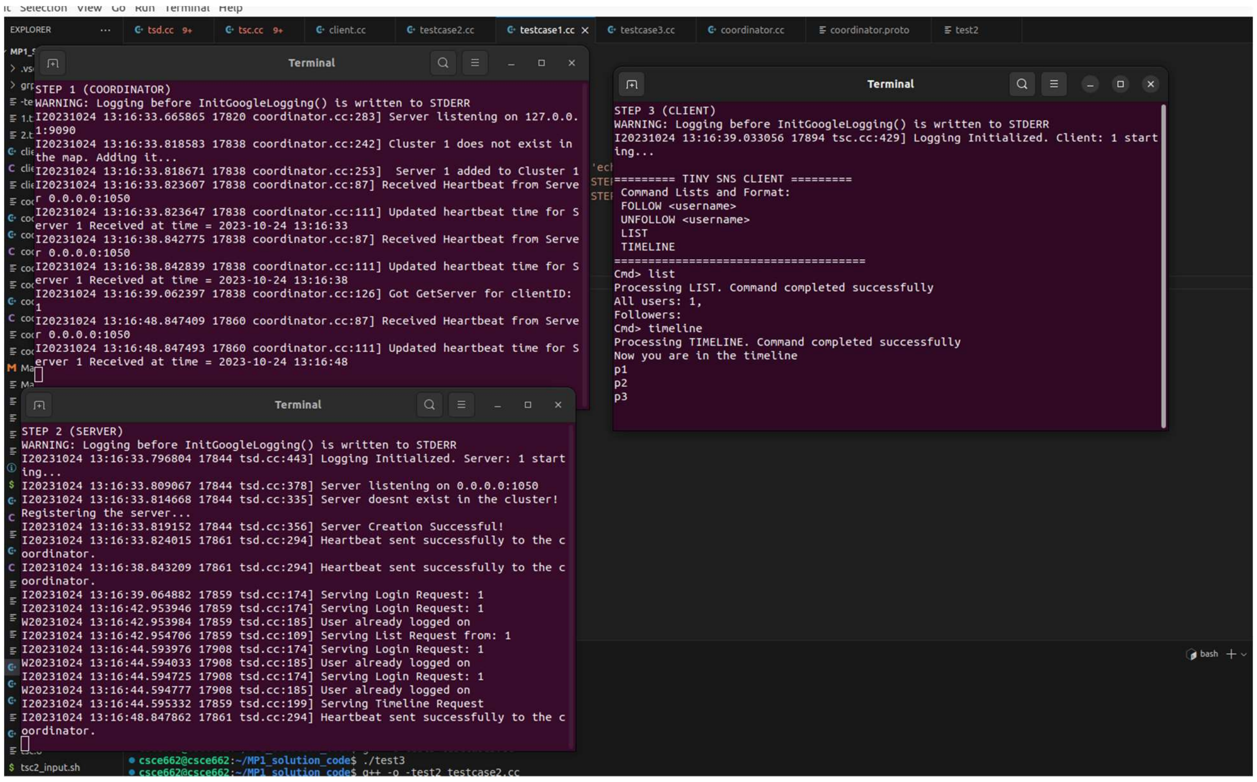
UIN – 634005665

Video Recordings of Testcases:

- TestCase 1 - https://www.youtube.com/watch?v=eOA9xZdxIcA&ab_channel=PavitraMalhotra
- TestCase 2- https://www.youtube.com/watch?v=WpSboE8CGvY&ab_channel=PavitraMalhotra
  TestCase 3 - https://www.youtube.com/watch?v=acYge_eQF9U&ab_channel=PavitraMalhotra

TestCase Screenshots:

TestCase 1



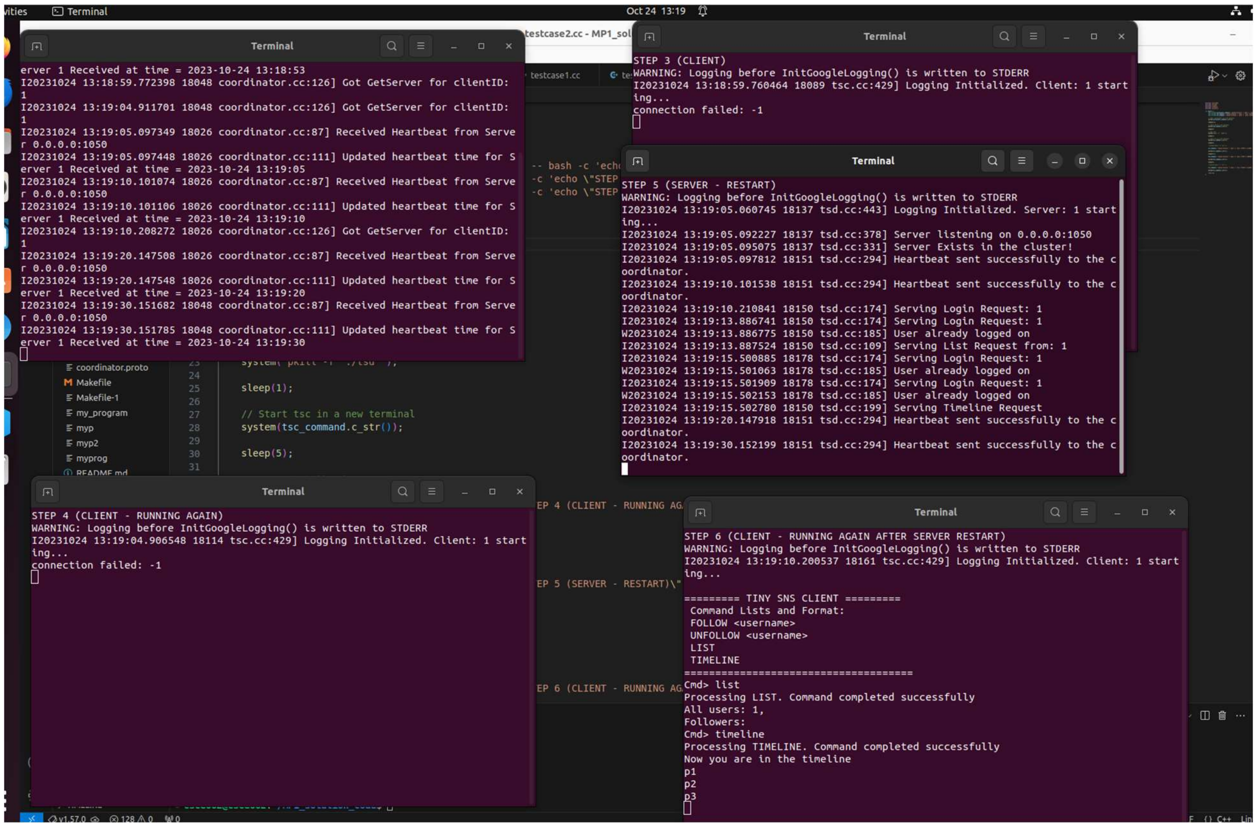
```
STEP 1 (COORDINATOR)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:16:33.665865 17820 coordinator.cc:283] Server listening on 127.0.0.
1:9090
I20231024 13:16:33.818583 17838 coordinator.cc:242] Cluster 1 does not exist in
the map. Adding it...
I20231024 13:16:33.818671 17838 coordinator.cc:253]  Server 1 added to Cluster 1
I20231024 13:16:33.823607 17838 coordinator.cc:87] Received Heartbeat from Serve
r 0.0.0.0:1050
I20231024 13:16:33.823647 17838 coordinator.cc:111] Updated heartbeat time for S
erver 1 Received at time = 2023-10-24 13:16:33
I20231024 13:16:38.842775 17838 coordinator.cc:87] Received Heartbeat from Serve
r 0.0.0.0:1050
I20231024 13:16:38.842839 17838 coordinator.cc:111] Updated heartbeat time for S
erver 1 Received at time = 2023-10-24 13:16:38
I20231024 13:16:39.062397 17838 coordinator.cc:126] Got GetServer for clientID:
1
I20231024 13:16:48.847409 17860 coordinator.cc:87] Received Heartbeat from Serve
r 0.0.0.0:1050
I20231024 13:16:48.847493 17860 coordinator.cc:111] Updated heartbeat time for S
erver 1 Received at time = 2023-10-24 13:16:48
```

```
STEP 2 (SERVER)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:16:33.796804 17844 tsd.cc:443] Logging Initialized. Server: 1 start
ing...
I20231024 13:16:33.809067 17844 tsd.cc:378] Server listening on 0.0.0.0:1050
I20231024 13:16:33.814668 17844 tsd.cc:335] Server doesnt exist in the cluster!
Registering the server...
I20231024 13:16:33.819152 17844 tsd.cc:356] Server Creation Successful!
I20231024 13:16:33.824015 17861 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:16:38.843209 17861 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:16:39.064882 17859 tsd.cc:174] Serving Login Request: 1
I20231024 13:16:42.953946 17859 tsd.cc:174] Serving Login Request: 1
W20231024 13:16:42.953984 17859 tsd.cc:185] User already logged on
I20231024 13:16:42.954706 17859 tsd.cc:109] Serving List Request from: 1
I20231024 13:16:44.593976 17908 tsd.cc:174] Serving Login Request: 1
W20231024 13:16:44.594033 17908 tsd.cc:185] User already logged on
I20231024 13:16:44.594725 17908 tsd.cc:174] Serving Login Request: 1
W20231024 13:16:44.594777 17908 tsd.cc:185] User already logged on
I20231024 13:16:44.595332 17859 tsd.cc:199] Serving Timeline Request
I20231024 13:16:48.847862 17861 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
```

```
STEP 3 (CLIENT)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:16:39.033056 17894 tsc.cc:429] Logging Initialized. Client: 1 start
ing...

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> list
Processing LIST. Command completed successfully
All users: 1,
Followers:
Cmd> timeline
Processing TIMELINE. Command completed successfully
Now you are in the timeline
p1
p2
p3
```

```
csce662@csce662:~/MP1_solution_code$ ./test3
csce662@csce662:~/MP1_solution_code$ g++ -o -test2 testcase2.cc
```

TestCase 2



```
erver 1 Received at time = 2023-10-24 13:18:53
I20231024 13:18:59.772398 18048 coordinator.cc:126] Got GetServer for clientID:
1
I20231024 13:19:04.911701 18048 coordinator.cc:126] Got GetServer for clientID:
1
I20231024 13:19:05.097349 18026 coordinator.cc:87] Received Heartbeat from Serve
r 0.0.0.0:1050
I20231024 13:19:05.097448 18026 coordinator.cc:111] Updated heartbeat time for S
erver 1 Received at time = 2023-10-24 13:19:05
I20231024 13:19:10.101074 18026 coordinator.cc:87] Received Heartbeat from Serve
r 0.0.0.0:1050
I20231024 13:19:10.101106 18026 coordinator.cc:111] Updated heartbeat time for S
erver 1 Received at time = 2023-10-24 13:19:10
I20231024 13:19:10.208272 18026 coordinator.cc:126] Got GetServer for clientID:
1
I20231024 13:19:20.147508 18026 coordinator.cc:87] Received Heartbeat from Serve
r 0.0.0.0:1050
I20231024 13:19:20.147548 18026 coordinator.cc:111] Updated heartbeat time for S
erver 1 Received at time = 2023-10-24 13:19:20
I20231024 13:19:30.151682 18048 coordinator.cc:87] Received Heartbeat from Serve
r 0.0.0.0:1050
I20231024 13:19:30.151785 18048 coordinator.cc:111] Updated heartbeat time for S
erver 1 Received at time = 2023-10-24 13:19:30
```

```
STEP 3 (CLIENT)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:18:59.760464 18089 tsc.cc:429] Logging Initialized. Client: 1 start
ing...
connection failed: -1
```

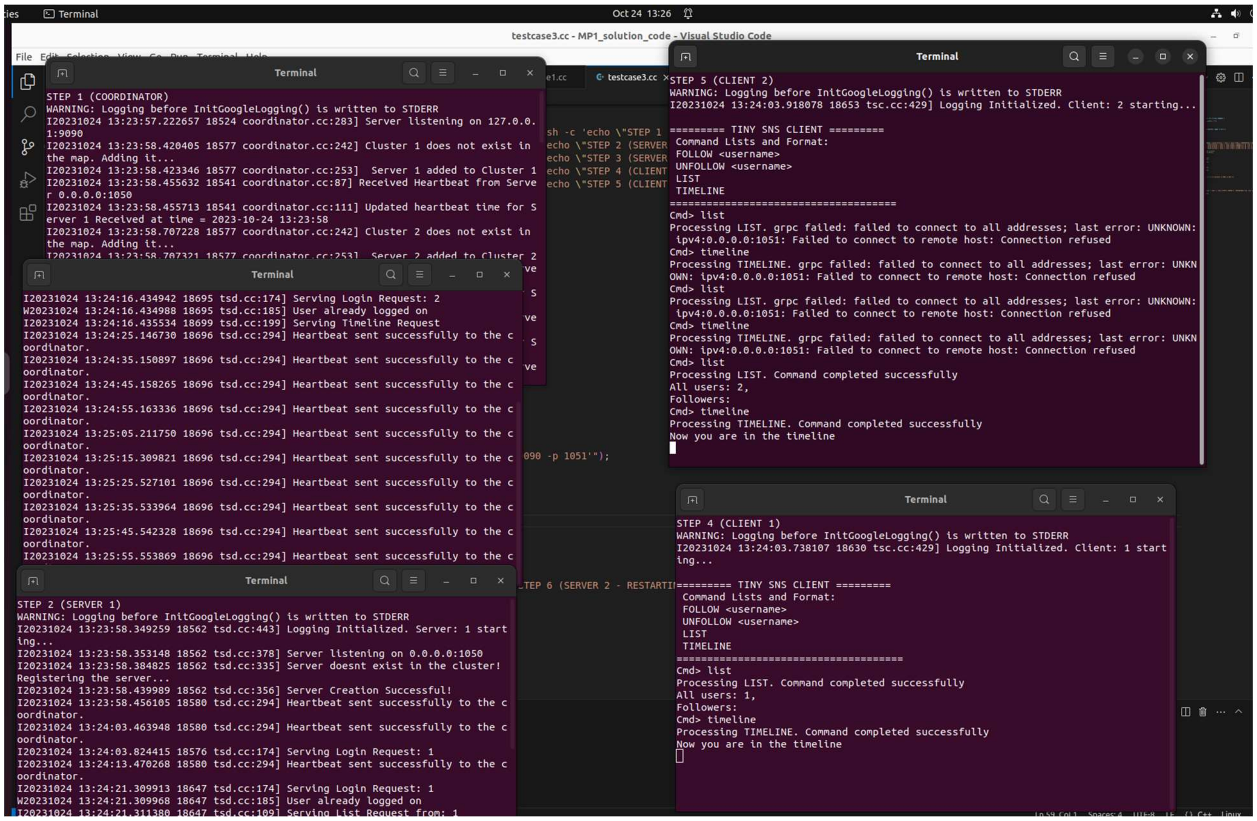
```
STEP 5 (SERVER - RESTART)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:19:05.060745 18137 tsd.cc:443] Logging Initialized. Server: 1 start
ing...
I20231024 13:19:05.092227 18137 tsd.cc:378] Server listening on 0.0.0.0:1050
I20231024 13:19:05.095075 18137 tsd.cc:331] Server Exists in the cluster!
I20231024 13:19:05.097812 18151 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:19:10.101538 18151 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:19:10.210841 18150 tsd.cc:174] Serving Login Request: 1
I20231024 13:19:13.886741 18150 tsd.cc:174] Serving Login Request: 1
W20231024 13:19:13.886775 18150 tsd.cc:185] User already logged on
I20231024 13:19:13.887524 18150 tsd.cc:109] Serving List Request from: 1
I20231024 13:19:15.500885 18178 tsd.cc:174] Serving Login Request: 1
W20231024 13:19:15.501063 18178 tsd.cc:185] User already logged on
I20231024 13:19:15.501909 18178 tsd.cc:174] Serving Login Request: 1
W20231024 13:19:15.502153 18178 tsd.cc:185] User already logged on
I20231024 13:19:15.502780 18150 tsd.cc:199] Serving Timeline Request
I20231024 13:19:20.147918 18151 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:19:30.152199 18151 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
```

```
    sleep(1);

    // Start tsc in a new terminal
    system(tsc_command.c_str());

    sleep(5);
```

```
STEP 4 (CLIENT - RUNNING AGAIN)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:19:04.906548 18114 tsc.cc:429] Logging Initialized. Client: 1 start
ing...
connection failed: -1
```

```
STEP 6 (CLIENT - RUNNING AGAIN AFTER SERVER RESTART)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:19:10.200537 18161 tsc.cc:429] Logging Initialized. Client: 1 start
ing...

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> list
Processing LIST. Command completed successfully
All users: 1,
Followers:
Cmd> timeline
Processing TIMELINE. Command completed successfully
Now you are in the timeline
p1
p2
p3
```

TestCase 3



```
STEP 1 (COORDINATOR)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:23:57.222657 18524 coordinator.cc:283] Server listening on 127.0.0.
1:9090
I20231024 13:23:58.420405 18577 coordinator.cc:242] Cluster 1 does not exist in
the map. Adding it...
I20231024 13:23:58.423346 18577 coordinator.cc:253]  Server 1 added to Cluster 1
I20231024 13:23:58.455632 18541 coordinator.cc:87] Received Heartbeat from Serve
r 0.0.0.0:1050
I20231024 13:23:58.455713 18541 coordinator.cc:111] Updated heartbeat time for S
erver 1 Received at time = 2023-10-24 13:23:58
I20231024 13:23:58.707228 18577 coordinator.cc:242] Cluster 2 does not exist in
the map. Adding it...
I20231024 13:23:58.707321 18577 coordinator.cc:253]  Server 2 added to Cluster 2
```

```
I20231024 13:24:16.434942 18695 tsd.cc:174] Serving Login Request: 2
W20231024 13:24:16.434988 18695 tsd.cc:185] User already logged on
I20231024 13:24:16.435534 18699 tsd.cc:199] Serving Timeline Request
I20231024 13:24:25.146730 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:24:35.150897 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:24:45.158265 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:24:55.163336 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:25:05.211750 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:25:15.309821 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:25:25.527101 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:25:35.533964 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:25:45.542328 18696 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:25:55.553869 18696 tsd.cc:294] Heartbeat sent successfully to the c
```

```
STEP 2 (SERVER 1)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:23:58.349259 18562 tsd.cc:443] Logging Initialized. Server: 1 start
ing...
I20231024 13:23:58.353148 18562 tsd.cc:378] Server listening on 0.0.0.0:1050
I20231024 13:23:58.384825 18562 tsd.cc:335] Server doesnt exist in the cluster!
Registering the server...
I20231024 13:23:58.439989 18562 tsd.cc:356] Server Creation Successful!
I20231024 13:23:58.456105 18580 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:24:03.463948 18580 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:24:03.824415 18576 tsd.cc:174] Serving Login Request: 1
I20231024 13:24:13.470268 18580 tsd.cc:294] Heartbeat sent successfully to the c
oordinator.
I20231024 13:24:21.309913 18647 tsd.cc:174] Serving Login Request: 1
W20231024 13:24:21.309968 18647 tsd.cc:185] User already logged on
I20231024 13:24:21.311380 18647 tsd.cc:109] Serving List Request from: 1
```

```
STEP 5 (CLIENT 2)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:24:03.918078 18653 tsc.cc:429] Logging Initialized. Client: 2 starting...

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> list
Processing LIST. grpc failed: failed to connect to all addresses; last error: UNKNOWN:
 ipv4:0.0.0.0:1051: Failed to connect to remote host: Connection refused
Cmd> timeline
Processing TIMELINE. grpc failed: failed to connect to all addresses; last error: UNKN
OWN: ipv4:0.0.0.0:1051: Failed to connect to remote host: Connection refused
Cmd> list
Processing LIST. grpc failed: failed to connect to all addresses; last error: UNKNOWN:
 ipv4:0.0.0.0:1051: Failed to connect to remote host: Connection refused
Cmd> timeline
Processing TIMELINE. grpc failed: failed to connect to all addresses; last error: UNKN
OWN: ipv4:0.0.0.0:1051: Failed to connect to remote host: Connection refused
Cmd> list
Processing LIST. Command completed successfully
All users: 2,
Followers:
Cmd> timeline
Processing TIMELINE. Command completed successfully
Now you are in the timeline
```

```
STEP 4 (CLIENT 1)
WARNING: Logging before InitGoogleLogging() is written to STDERR
I20231024 13:24:03.738107 18630 tsc.cc:429] Logging Initialized. Client: 1 start
ing...

========= TINY SNS CLIENT =========
 Command Lists and Format:
 FOLLOW <username>
 UNFOLLOW <username>
 LIST
 TIMELINE
=====================================
Cmd> list
Processing LIST. Command completed successfully
All users: 1,
Followers:
Cmd> timeline
Processing TIMELINE. Command completed successfully
Now you are in the timeline
```

How to run the program:

I have made the test scripts to run the program for the testcases.

Testscripts are named – testcase1.cc , testcase2.cc and testcase3.cc

Commands to run:

Coordinator - ./coordinator -p 9090

Server - ./tsd -c [ClusterID] -s [ServerID] -h localhost -k 9090 -p 3000

Client - ./tsc -h localhost -k 9090 -u [ClientUsername]

Objective

The aim of this task is to utilize understanding of Google Protocol Buffers and gRPC in developing a basic Social Networking Service (SNS) system. This system mirrors popular platforms such as Facebook or Twitter, permitting users to share and receive status updates. Important points to take into account encompass:

- **User Modes**: Users operate in two modes, namely "command mode" and "timeline mode." When the client program starts, it begins in command mode, enabling users to execute commands like FOLLOW, UNFOLLOW, LIST, and TIMELINE.
- **Following**: Users can follow other users using the FOLLOW command. When a user posts an update, their followers can see it on their timelines. The FOLLOW command links the current user to the specified user's timeline, with followers only seeing new posts.
- **Unfollowing**: The UNFOLLOW command removes the current user from another user's timeline.
- **Listing Users**: The LIST command retrieves lists of existing users and users who follow the current user.
- **Timelines**: Each user has their own timeline and can post updates to it. The TIMELINE command switches a user from command mode to timeline mode, allowing them to post updates to their timeline and their followers' timelines. Once in TIMELINE mode, users cannot return to command mode.
- **Timeline View**: Followers see the last 20 posts, which include the username, posting time, and the actual posting content.
- **Coordinator**: Servers first register themselves to the coordinator cluster and clients first connect to coordinator to get cluster and serverID and then clients connect to that server.

1. Coordinator

Introduction

The Coordinator is a crucial component in a distributed system designed to manage and coordinate multiple server instances. Its primary responsibilities include registering servers, monitoring their heartbeats to determine their availability, providing information about active servers to clients, and facilitating communication between clients and servers. This document outlines the design and functionality of the Coordinator component.

Coordinator Components Overview

Server Registration

- Servers register themselves with the Coordinator upon startup.
- Registration includes server identification (ID), hostname, port, and cluster information.
- The Coordinator stores server information in a data structure organized by cluster IDs.

Heartbeat Monitoring

- The Coordinator continually monitors the heartbeats of registered servers.
- Heartbeats are sent by servers at regular intervals to signal their availability.
- Servers that miss heartbeats for an extended period are marked as inactive.

Cluster Management

- Servers are grouped into clusters based on their cluster IDs.
- Each cluster has a vector of server nodes, each representing a registered server.
- The Coordinator maintains a map that associates cluster IDs with server vectors.

Client Interaction

- Clients contact the Coordinator to request information about active servers.
- Clients provide their unique IDs, and the Coordinator assigns them to clusters.
- The Coordinator responds with the hostname and port of an available server in the client's cluster.

CoordServiceImpl Implementation

Heartbeat

- The Heartbeat method receives heartbeats from servers.
- It updates the last heartbeat time for the respective server and marks it as active.
- The method also confirms the receipt of the heartbeat.

GetServer

- The GetServer method provides server information to clients.
- Clients pass their IDs to the Coordinator to request a server assignment.
- The Coordinator identifies the cluster based on the client's ID.
- If active servers are available in the cluster, the Coordinator assigns one to the client.
- If no active servers are found, it responds with a status indicating unavailability.

Exists

- The Exists method checks if a zNode (server) exists in a cluster.
- It verifies if a server with a given ID is registered in a specific cluster.

- If found, it responds with a status indicating existence.
- If not found, it responds with a status indicating non-existence.

Create

- The Create method registers a new zNode (server) in a cluster.
- It verifies if the server already exists in the cluster.
- If the server is already registered, it responds with a status indicating existence.
- If the server is not registered, it adds it to the cluster and responds with a status indicating success.

Heartbeat Monitoring

- A separate thread, checkHeartbeat, continually checks the heartbeats of registered servers.
- It iterates through clusters and zNodes, marking zNodes with missed heartbeats as inactive.
- Servers with missed heartbeats longer than 10 seconds are marked as inactive.
- This thread runs in the background to ensure servers are monitored continuously.

Error Handling

- The system performs error checks on both the client and server sides.
- Client-side checks ensure that client commands adhere to the specified interface.
- Server-side checks handle errors such as file handling, non-existent users, login attempts with existing usernames, etc.

Concurrency

- The Coordinator is designed to handle multiple server and client interactions simultaneously.
- It utilizes threads to manage heartbeat monitoring and respond to client requests concurrently.

2. **Overview**
   A Social Networking System is designed to enable communication between users through a Google RPC (gRPC) based client-server system. Users can login, follow each other, unfollow, post messages and view timelines of the users they follow. Coordinator handles the server registration and assigns clients the server to which they can connect.

3. **Client and Server Components Overview**
   Client Program:
   - Responsible for interacting with users and sending requests to the server.

- Provides command line interface for users to login, follow, unfollow, list users, post messages and see timeline.
- gRPC based communication is used to interact with the server.
- Login: Handles user authentication and logs in, setting the communication status accordingly.
- connectTo: Establishes a connection to the server and initializes the gRPC stub for service calls.
- processCommand: Parses user input to determine the requested action and returns an IReply with appropriate status.
- Follow: Sends a request to follow another user, updating the communication status based on the server response.
- UnFollow: Sends a request to unfollow a user, updating the communication status based on the server response.
- Timeline: Manages bidirectional communication with the server for real-time message updates in a separate thread.
- GetServer: Client connects to coordinator first to get the server info such as hostname and port. Client then connects to that server.

Server Program:

- Create: Server connects to coordinator first to register itself as a zNode in the coordinator cluster.
- Exists: Function to check if server already exists as a zNode in the coordinator cluster.
- sendHeartbeat: Server sends repeated heartbeats to the coordinator to let the coordinator know that its alive.
- Server acts as a centralized hub to manage user accounts, messages, and interaction.
- It listens to gRPC requests from different clients on a given port address.
- The server handles user login, follow, unfollow, list and timeline commands.
- User data, followers and messages are stored in the local memory.
- Login: Handles user authentication, allowing users to connect to the server.
- Follow: Establishes a following relationship between two users, allowing one user to follow the other.
- UnFollow: Removes a following relationship between two users, allowing one user to unfollow another.
- getClient: Retrieves a client object from the database based on the provided username.

- write_timeline: Appends a message to a user's timeline file, storing their activity.
- read_timeline: Reads the last 20 messages from a user's timeline file.
- Timeline: Manages bidirectional communication with the client for real-time message updates.

4. **Client Program Components**
   - Client Class: This class acts as the heart of the client program. It has jobs like connecting to the server, understanding what users want to do, and keeping track of what's happening in the user's timeline.
   - gRPC Communication: The client program uses gRPC to make a connection and ask the server for things. It talks to the server using special message formats called protobufs.
   - Command Processing: The client program understands what the user wants to do, puts together requests to send to the server, and deals with the answers that come back from the server.
   - Command Line User Interface: The client provides a command-line interface for users to input commands and view responses.

5. **Server Program Components**
   - Initializing Client: The client initializes by providing a hostname, username, and port. It establishes a connection with the server.
   - Login: The user logs in by sending a Login request to the server. The server checks if the user already exists and responds accordingly.
   - Follow/Unfollow: The client sends messages to the server to follow or stop following someone. The server changes the connection between users meaning changes in the follower/following database structures and tells whether it worked or not.
   - List Users: This command lists the users who are currently logged in.
   - Timeline: Users can post messages and view their timeline. When a user posts a message, it is timestamped and stored. Other users who follow the poster receive the message in their timeline.
   - gRPC Communication: The client and server communicate via gRPC using defined service methods, request, and response message types.

6. **Error Handling**
    - System checks for errors both on client side and server side.
    - Client side checks for command errors to make sure that client commands are in accordance with the given interface.
    - Server side checks for errors such as file handling, non existent users, login attempts with existing usernames, etc.

7. **Concurrency**
    - Server handles multiple clients at once, handling the login, follow, unfollow, etc., operations simultaneously.
    - Server incorporates threads, to simultaneously read and write messages between the users when in timeline mode.

8. **Conclusion**
    The SNS system provides a simple yet effective platform for users to connect, communicate, and share information with each other via centralized coordinator. It uses gRPC for efficient communication between clients and servers, ensuring real-time updates and interactions.