Overall, MP2 involves an addition to MP1 whereby we can connect multiple clients to multiple servers rather than one in MP1. Given there are multiple servers, a centralized coordinator is set up to route each client to the correct server by performing a calculation on the user id.

In all the files, we first have to update the main function to take in different command line arguments. In the client, we obtained the coordinator IP and coordinator port from the user, which was used to create a separate Coordinator Service stub. This helps to call server-side functions in the coordinator service, unlike directly calling the SNS service in MP1. By calling GetServer in the main as a stub, we obtain the server port and server hostname that the coordinator wants to route the client to. As a result, we use this to Login to the right server in connectTo() function. In addition, the timeline function does not always return success when called by client. This is because a server can have downtime, which is why we called List() that would return fail if a server is down. This happens in processCommand() under the if check for timeline. Besides all these, the rest of the functionality remains the same as in MP1.

In the Coordinator file, we have implemented a series of functions.

The main Heartbeat function on the coordinator site is a thread-safe function that updates a server's entry in the database of clusters in the coordinator when invoked by the server's stub. Since server database information is sensitive, we implement a mutex lock during updates. However, outside the lock, when a new server is to be added, we populate the information from serverinfo that the client passes through the coordinator stub. that comes from the server stub into the respective cluster.

The GetServer function in the coordinator assigns a client to a server by populating serverInfo information with the information from its cluster database so the client retrieves the right port to connect to.

The RunServer function serves a core purpose in the MP2 by calling the thread function checkHeartbeat, which is a thread-safe function that turn runs indefinitely to monitor if a server does not send a heartbeat message to the coordinator for more than 10s, indicating a downtime.

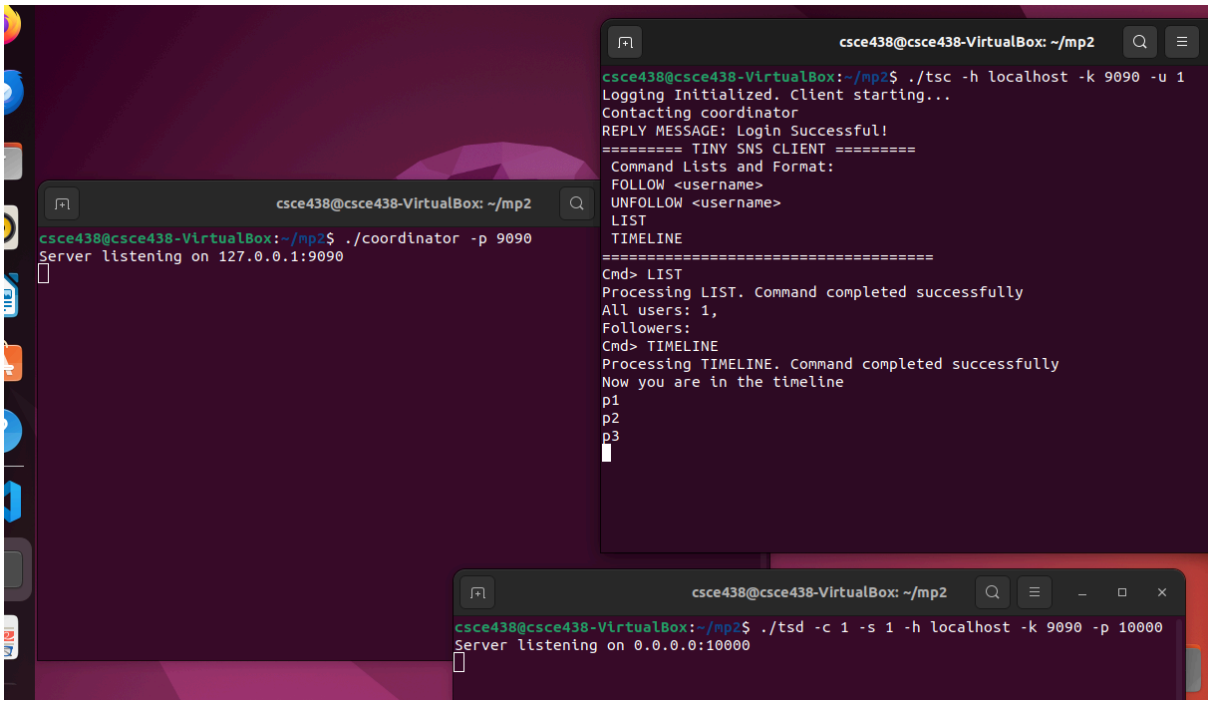
In the Server file, we have implemented a series of functions.

First, we have introduced a new server mutex to allow thread-safe function calls, especially for heartbeat which will be explained.

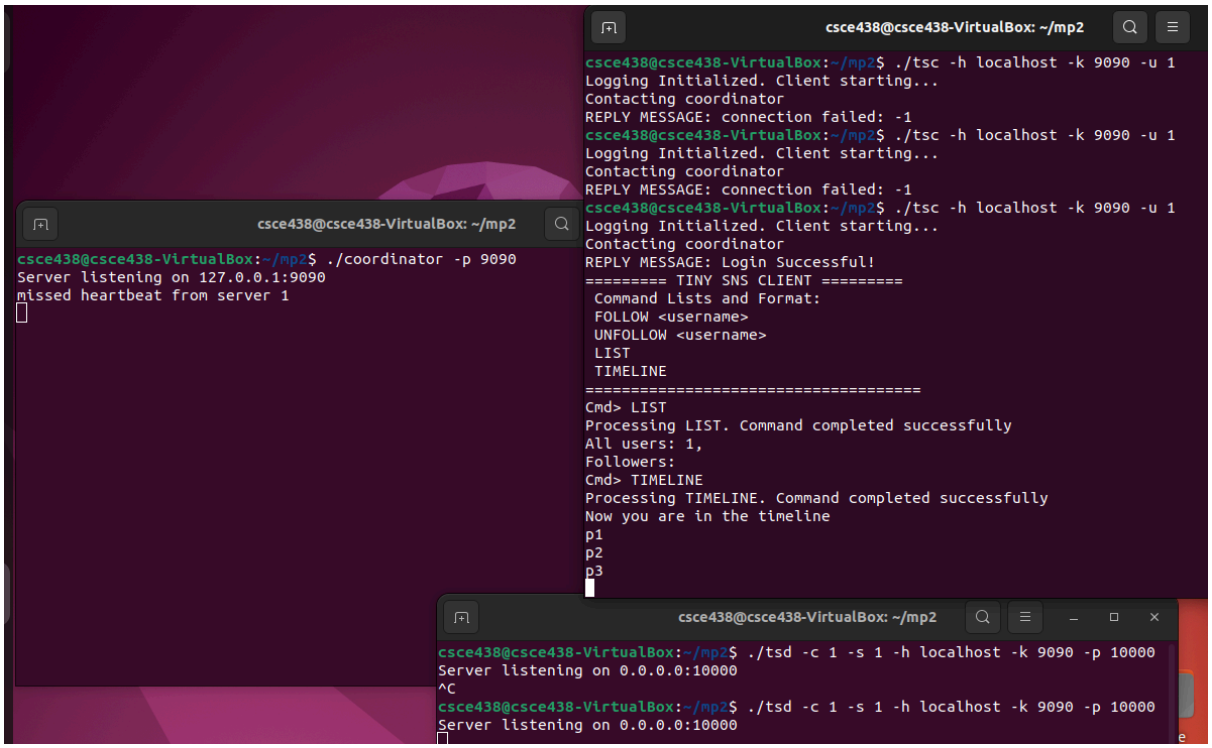
The RunServer function has additional arguments for coordinatorIP and coordinatorPort and cluster ID and portNum. While executing the server, we call a custom made heartbeat thread function using all the input arguments that calls a HeartBeat helper function for an indefinite amount of time. We have a gap of 1 second between each heartbeat between coordinator and server as a constant short fixed interval to inform that server is active to coordinator.

In the HeartBeat helper function, we populate the ServerInfo object using the input arguments from the user. The server acts as a client to the coordinator and calls the client stub (as a sns server) to the coordinator side heartbeat function passing the server information. This step sends a heartbeat to the coordinator to know server is active, otherwise it cannot process client commands.

The following is the output of the first test case for the sanity check.



The following is the output of the second test case where the server is killed before the client. We initialize 1 coordinator, server 1, and client 1.



The following is the output of the third test case where the server is killed after the client. We initialize 1 coordinator, server 1, server 2, and client 1 and client 2.