Project Report

Multiple User based Concurrent Online Calendar

By

Muhammad Ahad Ul Alam

Rupak Roy

Samiha Sharmin Shimmi

Introduction

Imagine a scenario where the employees of a common workplace need to use their shared resources like clusters and machines in a way so that no conflict occurs. Miscommunication among them might lead to erroneous performance evaluation. For this reason, for our project, we designed multiple users based concurrent online calendar that can be used by any user within the same local area network. With our application, multiple users can add events to a shared calendar. The employees can add their scheduled time to use the shared resources and thus avoid conflict. Multiple users can register and log into our system. Then, they can add events. A collaborative calendar is not a new idea at all. There are few online calendar applications available to the users like Google Calendar, Microsoft Outlook calendar etc. However, with our limited timeframe and limited resources, we wanted to implement a calendar that has basic features of a collaborative calendar and also can demonstrate our theoretical knowledge gathered in the Concurrent, Parallel and Distributed Programming class.

Basic Objectives

Basic objectives of our project is as follows:

- See how the practical implications of our knowledge gathered in Concurrent, Parallel and Distributed programming class works in real life.
- To develop a multi-user event management calendar in C++ that works in Local Area Network and can achieve concurrency via socket programming.
- To analyze some current technologies like Google Calendar and Outlook calendar and design a simple application which has the basic functionalities of a collaborative calendar.
- To learn GUI design in Qt creator using C++.
- To be acquainted with software design lifecycle- Design, Development and Testing. deployment.

System Model

Our system consists of two basic components:

1. Server
2. Client

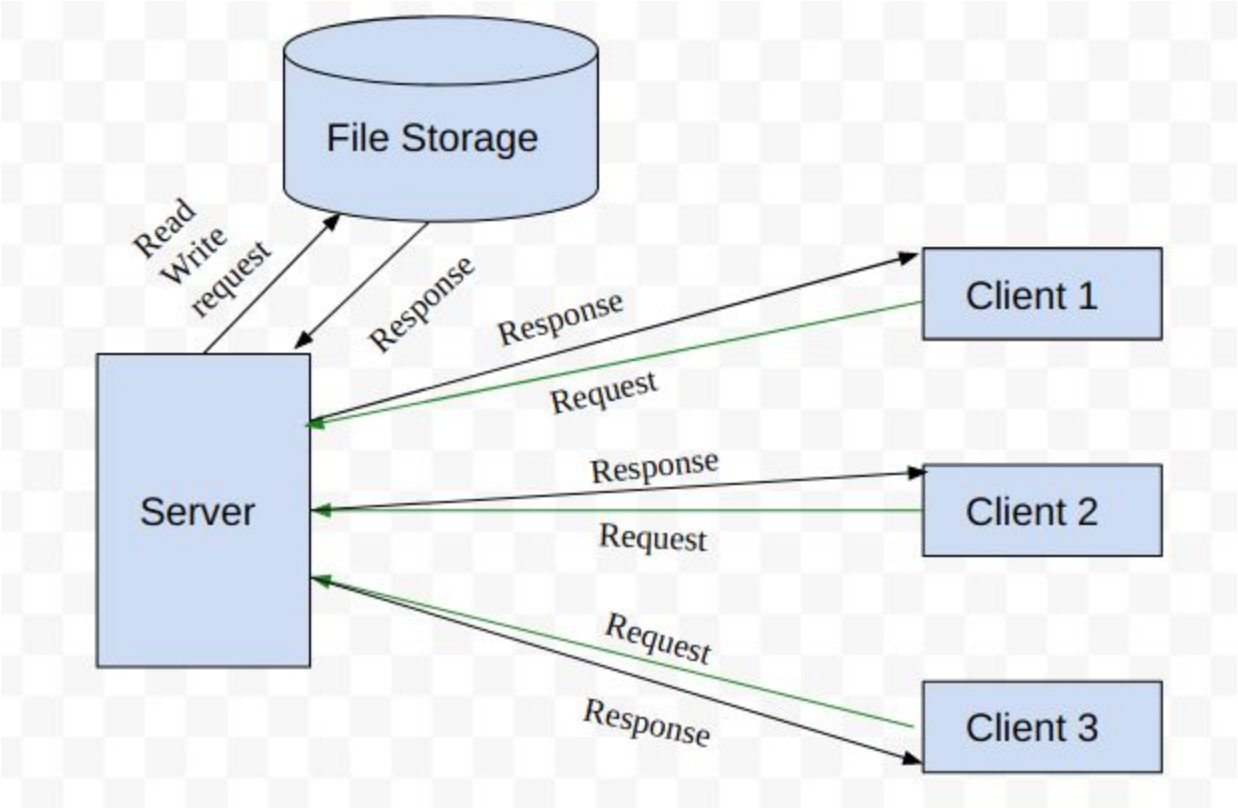

Fig: System model of concurrent calendar

The above figure shows the basic overview of our system. Each of our components are described below:

The Server

The server contains the text file where we stored our central calendar. It also contains user information in another file. It resides in the same LAN as the clients. When a user first login to the server, the server provides the last updated file to clients so that user can see the events of other users to mitigate conflicting. The server has been implemented using Socket programming

.IO Multiplexing was done using the ‘select’ system call . We actually used the prefork server mechanism while implementing server side and the maximum user is configurable.

The Client

We have designed an interface for clients. In the starting window, Users have options for registration and log-in. After the registration, the user will see an interface to edit event date, event time and event descriptions. He can also see existing events. When a user submits a new event, It will be added to his own list as well as the event list of other online users.

Platform

We have developed our calendar using C++. The server has been implemented using Socket programming . Visual Studio Code(VScode) IDE is used to develop and run the server side. The client has been implemented using Qt Creator framework . The text file, where the calendar is stored ,can be edited simultaneously by multiple clients opened from Qt interface.

Basic Features

In our concurrent calendar, we have the following features:

- User registration- Users have to register with username and password before logging in. It has been ensured that two persons will not be able to register with same username. He will be prompted to select a different username. Registration data will be saved in the file after successful registration.
- User login- After the registration is successful, users can login to their account to add events in the desired date and time.
- Date and time picker - Users can select date and time from date and time picker.
- Add event by multiple users - Multiple users can concurrently add events in their desired date and time. The events will be added to the common event list which will be shown to all logged in users.
- Refreshing the events added from other users -A refresh button has been added so that the users can refresh the event list whenever necessary.

- Logout- After logout all the data entered by user will be save in the file and when he will log in again data will be pulled from the file and displayed to the user.
- Exit- User can close the whole application if he wants.

Snippets of our Calendar:

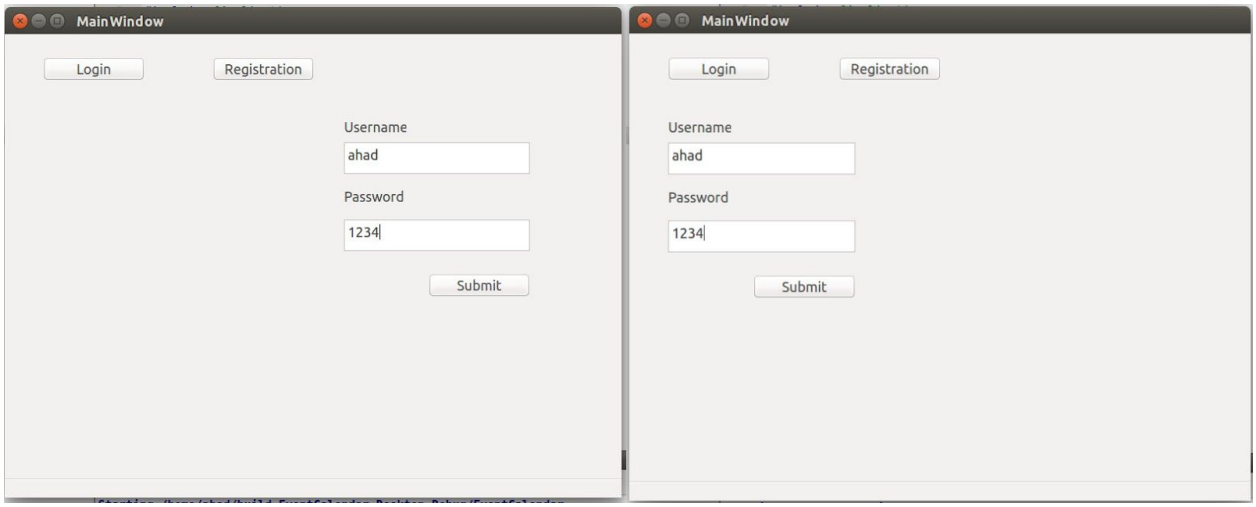

Fig1: Registration window and Login Window of the Calendar

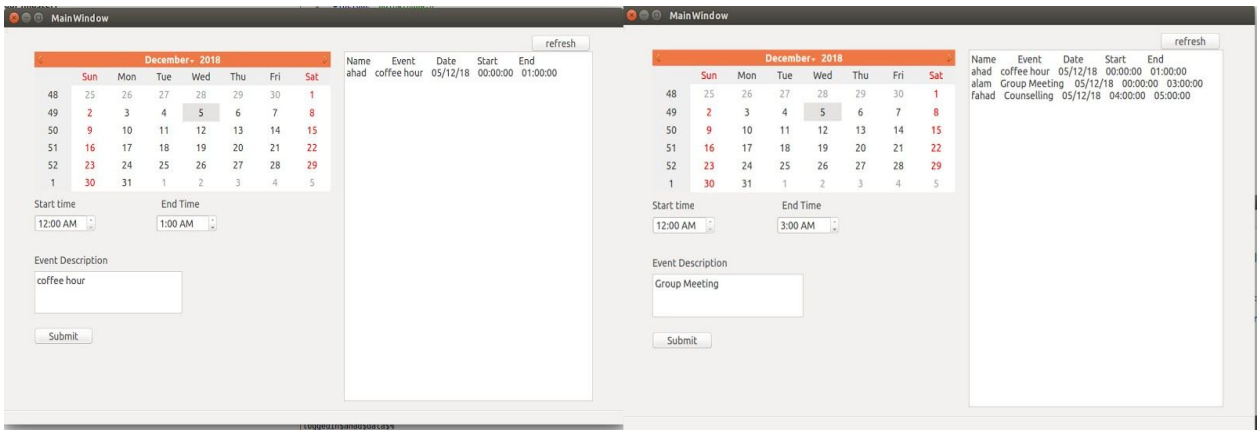

Fig2: Calendar Window after logging in and edition of events

Future Works

We want to improve our calendar in future. Currently we can add events. We want to add features like editing or deleting events. Also we want to improve our interface to make it more user friendly. We also intend to add social plugins(e.g.Facebook) with our calendar so that those events can be linked to the events of social networks. A push notification for the start of event can also be convenient for the user.

Limitations

- ➢ We have not added any **interval scheduling** algorithm to prevent users from editing event at the same date and same interval.
- ➢ In that case, we assumes that, when a user adds a new event he first checks the earlier events that are added by other users. Though still, there is a chance two user can simultaneously add future event at the same time.

Conclusion

We designed a multiple user based online collaborative calendar. We tried to implement our knowledge of Concurrent and Parallel programming we gathered in the class. We can use this calendar to share resources within a group of people within the same local area network. However, we want to improve it and add some more unique features that can distinguish our calendar from other existing calendars.

References

1. “Unix Network Programming Volume 1: The Sockets Networking API,'‘ 3rd Edition, by W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff. 2003.
2. “The C Programming Language”, 2nd Edition, by B. Kernighan and D. Ritchie.
3. Jeff Huang, Charles Zhang, and Julian Dolby, "CLAP: recording local executions to reproduce concurrency failures", SIGPLAN Not. 48, 6 (June 2013), pp. 141-152.

4. Lowry, Paul Benjamin and Curtis, Aaron Mosiah and Lowry, Michelle Rene, A Taxonomy of Collaborative Writing to Improve Empirical Research, Writing Practice, and Tool Development (2004). Journal of Business Communication (JBC), Vol. 41, No. 1, pp. 66-99, 2004
5. Class lecture slides.