Yingyi Wu and Sunyi Ping
Distributed Systems - CSCI 6510
Project 1
10/14/2018

Distributed Shared Calendar Application

Wuu-Bernstein replicated log and dictionary algorithm

Part I - File Structure

Our application uses Python3 and consists of four parts:

Event.py

- Defines a Meeting class with all properties of a meeting such as name, day, participants, etc., time comparing functions as well as a function handling conflicts.
- Defines an Event class representing the event that a user gives command of.

host.py

- Realizes all the initialization including UDP socket creation, site information setting, etc.
- Implements the text UI.

SharedCalendar.py

- Deals with crash recovery.
- Defines a SharedCalendar class, including insert, delete, update, hasRec and many other functions. This is where the Wuu-Bernstein algorithm is implemented.

Utils.py

- Implements two functions to convert time string and date string into time object and datetime object, which is more convenient for comparison.

Part II - Data Structure & Implementation

This project uses UDP and multi-threading technology. Each site will have a sub-thread to listen to and process the information sent by other sites. This mechanism allows the main thread of each site to accept user commands.

Data Structures

Each site will have the following data structures:

The calendar uses a list structure that contains the meeting objects and contains all the information needed for the meeting, including date,start time, end time, name, and participants.

The log uses another list structure, which contains event objects, including meeting operation type and the site.

There is also a two-dimensional timetable in each site, which records the extent to which the site knows about each unused site.

Insert

When a user tries to create an meeting, the site will first filter all the items in the calendar to ensure that there are no duplicate names or conflicts. After making sure there is no conflict, the site will add this meeting to the local calendar and update the log, then send the new log and timetable to all participants to update their information.

Delete

The delete operation is similar to the insert operation. This site will check the local calendar first to ensure that the event to be deleted exists. It then deletes the meeting, then updates the log, and sends all the updated information to all of the meeting participants.

Send

When there is another participant in the creation or cancellation of the meeting, the site needs to send all the latest information to those participants. The information includes the latest log, the latest timetable.

Receive

When a site receives information from another site, it first needs to compare it with its own log and then filter out information it does not yet know. It then uses this new information to update its local calendar (create or delete).

In the Wuu-Bernstein algorithm, a site may not always have the most up-to-date information about the other sites' logs and dictionaries. Therefore, the algorithm described above may result in the scheduling of conflicting meetings.This situation will be detailed in the following chapters.

hasRec

As mentioned in the Wuu-Bernstein algorithm, this function is the tool we use to determine if another site knows an event. Through this function, we can reduce the size of the sent message to improve the transmission efficiency.

Part III - Failure and Recovery

To maintain the calendar in case of a failure or crash, the log, dictionary and timetable of each site are saved to three different text files named "log.txt", "dictionary.txt" and "timetable.txt". Each time an event of insertion, deletion, sending or receiving happens, a record function will be called to update the information in three files, that means the files are completely most up-to-date each time an event of interest happens.

To recover a crashed site, the system detects three text files in the host.py, reads from them and restores the data of log, dictionary and timetable recorded before the crash.

Part IV - Conflicts Handling

Due to a problem with this algorithm, we will deal with some embarrassing situations. Sometimes the information received will conflict with the local calendar of this site. In this case, the site must cancel some of the meetings. Here are a few rules that deal with this type of problem.

1. Each site only checks for conflicts related to itself.
2. If there are conflicts between two meetings, the meeting with the name that is lexicographically second should be cancelled.

3 If there is a meeting which is just received conflicts multiple local meetings, this site will delete as few meetings as possible to avoid conflicts.