Computer Science 455/555
Distributed Calendar (Phase I)
Programming Assignment 3 (100 points)

1 Description

In this assignment, we will implement a few pieces of a *distributed calendaring system*. We will keep the features of the system at a minimum so you can concentrate on the distributed aspects of the project. Some user interaction will also be quite primitive–but it is intended to make your implementation issues simpler.

The first phase will consist of a single server that can handle multiple clients. In the second phase we will have multiple servers and clients. Note that the server adds on to the functionality of the identity server that you have developed in the previous assignment. Hence, you can reuse the code from the previous project.

2 Specifications

2.1 Layout of the calendar files

In this project, we will use a simple text file format for storing the calendars. This makes it easily readable outside of the program. It also makes it portable across various clients or calendaring system. This idea is used by various calendaring systems as well as various Personal Information Management (PIM) applications. One well known standard is the `vCalendar` format. See the website `http://www.imc.org/pdi` for details of the `vCalendar` format. In our project, we will use a real simple format of storing a calendar entry per line in a text file. In a production system, we would have used the `vCalendar` format.

Each user will have a local calendar file named `calendar`. First, the current directory will be checked for the `calendar` file. Failing that, the file is checked for in the user's home directory. For the sake of readability, we will use the format `yyyy/mm/dd time` to record the time for appointments. A local `calendar` file will have entries of the form:

17#2006/11/4 4:55pm#public#The fun begins now#90#

- The first entry is a unique sequence number which can be used as a reference number (e.g., to delete a message). At startup, your calendaring client could scan these numbers so as to avoid duplicates if a new entry is added.
- The second field is a time value in the format `yyyy/mm/dd time`. You may assume that the time and date is always stored in current locale although it would be better to use UTC time for portability.
- The third field identifies the entry as private or public.
- The fourth field is a description for the calendar entry. We will assume an upper limit of 128 characters for the size of this field.

- The fifth field is the duration of the appointment in minutes.

Any entry in a client calendar file will normally reside in a corresponding global server file `CALENDAR`. You can have the global calendar file reside in the server's working directory. In this assignment, there is only one copy of the global calendar file. For the previous `calendar` file entry, the corresponding entry in the global file should have the form:

8953a618-3c1e-4bd8-a80a-0fe19f926394#17#2006/11/4 4:55pm#public#The fun begins now#90#

- The first field is the user's Universally Unique IDentification. The other fields are the same as in the local calendar file.

2.2 Client program

The client program should be interactive and may be either command line based or GUI based. It should provide for at least the following operations:

- At startup, the client first authenticates the client with the server.
- Permits a user to display his/her local `calendar` entries (do not use the global file when a user asks to display his/her own `calendar` entries).
- Permits a user to delete an entry via a sequence number. The corresponding message must also be deleted from the global calendar file (via a suitable remote method/procedure call).
- Permits a user to get a list of all users in the calendaring system.
- Permits a user to display the calendar entries for another user. This should be done via a suitable remote procedure/method call which uses the login name for the other user. Remember that "display" means that the server must send back the information which can then be displayed by the client. There should be two options. Either only global calendar entries for the other user are shown or both global and private entries are shown but the private entries are shown only as blocks of time that are busy without the client finding out what the other user is doing.
- Permits a user to add a new entry to their `calendar` file. The entry should normally be propagated to the server's calendar file but the user should be able to restrict the description field in the entry to their local file, creating a private entry. For a private entry, the entry is propagated to the server but the last field is kept private by withholding the information. Here is what the server stores for the example shown earlier (if it was private instead of public).

  8953a618-3c1e-4bd8-a80a-0fe19f926394#17#4/11/2006 4:55pm#private##90#

2.3 Single Server

- Initially, implement only one server. The server is RMI based. The server must be safe in multi-threaded mode (default in RMI).

- The server presents a suitable RMI interface for clients to access the calendaring and authentication functions. Remember that multiple clients may call the same methods in the server at the same time so you will have to worry about race conditions. Simply making each method/function be a critical section in its entirety would defeat the point of making it multi-threaded.
- All communication between server and clients is encrypted using Secure Sockets Layer under RMI.
- The server periodically checkpoints all the accounts (same as in the identity server). There are two options for the global calendar information: either the server reads/writes the `CALENDAR` file directly for each operation or the server buffers the `CALENDAR` in memory. In the latter case, the `CALENDAR` file must be check-pointed to disk periodically. Note that the second case would have better performance and would be preferable. In a production system, it won't be feasible to buffer the entire global calendar in memory and a more efficient solution incorporating caching might be used.

3 Additional Features

- Have the client use the duration field to test a new entry for validity. So if a new entry overlaps with an existing entry, the client warns the user but permits it anyway.
- Add an option to ask for all free time slots for a user in a given time range (e.g. 8am to 5pm).

4 Documentation (15 points)

- You must document clearly how to setup and test your project. Describe clearly how much of the functionality is available. If you did any extra stuff, please document that clearly as well. You can do all this in the README file or use HTML or PDF or OpenOffice or MS Word format.

5 Submitting the Assignment

Copy over all the code (and corresponding makefiles, `README` files etc) to a directory on `onyx`, then change your current directory to that directory and invoke submit as follows:

submit amit cs555 3