ID2010 Lab 1 - Chat

Andreas Hallberg
KTH Royal Institute of Technology
CINTE2010 / TSEDM 2013
Email: anhallbe@kth.se

I. Introduction

This report will give a short description of how I implemented a "Session Statistics" feature in a piece of chat software. The features I have added include:

- Let user see who is online
- Display the uptime of each user
- Notify everyone when a new user enters the chat
- Notify everyone when a user leaves the chat
- Notify when a user changes username

To do this I had to make a few small changes in the client software, and most of the changes were made in the server and its' interface.

II. Interface Changes

The following methods were added to the ChatServerInterface:

- **List<String>** ***registeredUsers()*** - This method is called by the client and returns a list of registered (i.e online) users.
- **void** ***changeName(REL rel, String newName)*** - This method is called by the client when it wants to change the username that is displayed to other clients. REL is in this case a RemoteEventListener, which is used to identify the client.

I also modified the *register(REL rel)* method to include the username of the client: *register(REL rel, String name)*. This is to make it easier to map the reference to a client with its associated username.

III. Client changes

To let the user list connected clients, it can now use the **.users** command.

```
Client> .users
User name                time (h:m:s)
-------------------------------------
alice                    0:0:29
bob                      0:0:34
```

The only significant changes to the client was to add a method *showRegisteredUsers()* which makes a call to the server and prints the output. The *setName()* method was also modified to make a *changeName()*-call to the currently connected server.

IV. Server Changes

The majority of the modifications were on the server-side. The interface changes were implemented, and a new data structure was introduced to keep track of user names and uptime.

A. Registering with the server

When a client calls the *register(REL rel, String name)* method, the server will put a new item in its *registeredClientMap*. The *rel* parameter is hashed and used as a key in the *registeredClientMap*. Each key is mapped to a **ClientWrapper** object with the following structure:

```
-----------------------------------
|           ClientWrapper         |
|---------------------------------|
|   -String username              |
|   -long   connectionTime        |
|---------------------------------|
|   +getUsername()                |
|   +getConnectionTime()          |
|   +setUsername(String n)        |
|   +setConnectionTime(long t)    |
|                                 |
-----------------------------------
```

The connectionTime of a client is the timestamp received from System.currentTimeMillis() at the time of client registration. This is used as a reference when the client wants to know how long a user has been connected.

The mapping is removed when the client unregisters (i.e disconnects) from the server.

B. Getting registered users

When a call is made to *registeredUsers()*, the server will simply gather all the values in registeredClientMap, calculate the time-difference between the call and connectionTime and fetch the name of each user. A list of strings with the format "username h:m:s" is returned which the client can print without any modifications.

C. Notifications

To notify other users that someone has connected, a message is simply added to the queue used for client-to-client communication every time someone calls *register()*, *unregister()* or *changeName()*. I did not find it necessary to have a separate queue for these messages. Each message contains

a prefix to separate client-to-client messages from server-to-client messages.

V. Testing

In this scenario I'm running one server (chatserver -n s1) and two clients named Alice and Bob. Alice is already connected to the server. Bob joins the server, changes his name, and then disconnects. Alice uses the **.users** command in between Bob's actions. Her chatclient output is depicted in Figure 1.

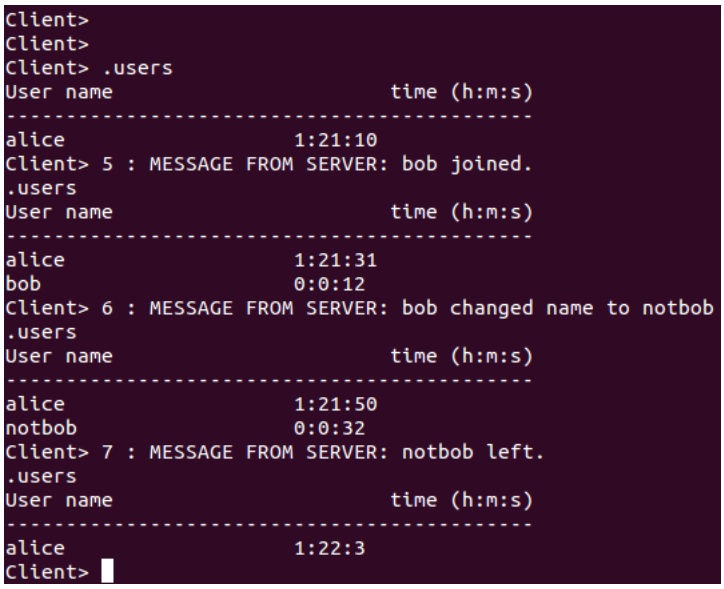

Fig. 1: Output from Alice's chatclient. Alice uses **.users** to see who's online. She also receives notifications when Bob connects, changes his name to notbob, and disconnects.