CS61A Lab 12: Client/Server Programming

Week 12, 2012

CS61AChat

We have written a nice little client-server set of programs for doing instant messaging! You can find the code in the following 4 files:

chatclient.py - Client program for running CS61AChat.
chatserver.py - Server program for running CS61AChat.
chatcommon.py - Common code that both the client and server programs use for running CS61AChat.
ucb.py - Used for the `@main` decorator it provides.

Enter the following commands to copy the relevant files over (the -r part is important for later).

```
mkdir lab12
cp -r ~cs61a/public_html/sp12/labs/lab12/* lab12
```

To run the chat program, first start the server program on a computer:

```
# python3 chatserver.py
```

This will ask you to find out your IP address and will tell you the port the server is using (it's randomly selected every time the server starts).

Note: If you're doing this from home, you're probably connected to a local router, in which case it won't work unless you're doing "port forwarding" (if you don't know what this is, don't worry about it). To handle this, log onto one of the school servers and use the server's full name (eg. star.cs.berkeley.edu) for the IP address when connecting the client.

Then start a client!

```
# python3 chatclient.py
```

This will ask you for a username, the server's address, and the server's port. Afterwards you can use `/help` to see all of the possible commands for the client. After logging on, you will receive any messages that others have sent you after each command (so you have to hit enter to receive new messages).

Question 0

Both the client and the server use the `Message` class to send messages to each other. A `Message` has four components:

> A source - who sent the message.
> A destination - who the message was intended for.
> An action - what type of message this represents.
> A body - any extra data the client or server wants to send, such as the body of a message.

Take a look at the code and list all the kinds of message actions that a `Message` sent from either the client or server can have. Who sends what types of messages?

Question 1

Modify the server program to understand a new type of message, which will have action `broadcast`. The server program should handle messages with action `broadcast` by sending the message on to all clients it has connected to it.

Once you have modified the server, add a new `/broadcast [message]` command to the client, which sends a broadcast message to the server.

Question 2

Modify the client program to have a blocked users list. The client should not see any messages from anyone on their blocked users list! Add three new commands for the client:

`/block [username]` - Add the given username to the block list.
`/unblock [username]` - Remove the given username from the block list.
`/block-list` - List the users the client has blocked.

Question 3

We've modified the pig gui from last week so that you can play pig over the network with a friend! The **/challenge [username]** and **/accept** commands challenge a player to a game of pig and accept a challenge from the last player who challenged you, respectively. However, the server doesn't quite know how to handle pig related messages (Hint: their actions all start with "_pig-"). Modify the server to pass these messages along correctly, and challenge a friend to a game!

The gui is run in a separate thread, which allows messages to be sent and received while you're playing. We haven't covered threading yet, but when we do you can come back to this lab for a simple example. For now, just be content with the fact that the gui works as described.