Lab 2 – Instant Messaging with RMI

CC5212-1

March 16, 2016

We're gonna build an instant messaging service using Java RMI. There will be a central directory listing all user names to co-ordinate everything. Everyone sets up an RMI server on their local machine to receive and print messages. Everyone uploads their username and the details of their server to the central directory and gets the list of active users. Then users can choose whom to message or can spam everyone.

- Download `http://aidanhogan.com/teaching/cc5212-1-2016/lab/02/mdp-lab02.zip`.
- I will set up a central directory somewhere in the class that you can connect to. I will give the details of the hostname and port in the class.

  If you are doing this assignment outside the lab, it will be a little boring in that you will have to send messages to yourself, but at least you can create multiple users or something so you can pretend you're not. As a first step, you will need to set up the directory yourself, for example on your local machine. In the code, there is `rmi.jar` in the `dist/` folder.

  - `java -jar rmi.jar StartRegistryAndServer -r -s 1`

  This starts a central directory. You need to keep this running, e.g., in a separate window (hence it is easier to run as a jar then in Eclipse). You can connect to this directory through localhost. The default port is 1099.
- First we'll quickly test the connectivity to the central directory using the example directory client `org.mdp.cli.UserDirectoryClient`. There are two todos.
  - Replace `DIR_HOSTNAME` and `DIR_PORT` with the details of the central directory on the board. (If running at home, you can leave these as they are since the default is localhost.)
  - Replace `"ahogan"` and `"Aidan Hogan"` with your own username and real name. Don't worry about IP or port for now.
  - Run the class. It will add you to the directory, delete you again and list the current users at each step. This is just to test that you can connect.
- Next we are going to implement the message client/server you are going to run on your local machine. Go to `org.mdp.im.InstantMessagingServer` and implement the `message(.,.)` method. *What should it do?*
- Next head to `org.mdp.cli.InstantMessagingApp`. A bunch of input/output stuff is done for you. What you need to do is implement the methods at the bottom. You may find useful code examples at the end of Monday's slides: `http://aidanhogan.com/teaching/cc5212-1-2016/MDP2016-02.pptx` or `http://aidanhogan.com/teaching/cc5212-1-2016/MDP2016-02.pdf`.
  - `startRegistry(.)` is done for you. It opens an RMI registry on your local machine on the port you will decide later.

  - Once you have a registry, you can register a skeleton for the `InstantMessagingServer` you just wrote and let a remote machine call it. In `bindSkeleton(.)`, you will add `skel` to the registry you just opened with key `skelname`. Now users will be ready to find your server and call the method to message you.
  - Next you need to implement the methods to send messages to other users. In `messageUser(.,.,.)`, you'll need to open up a user's registry and then retrieve an `InstantMessagingStub` instance from the registry using the given `stubName`. Finally, on that instance, you can call `message(.,.)` to message that user. Return the time returned by the remote user.
  - The last step is to open the connection to the directory. This is done in `connectToDirectory(.,.)` for you. It opens another registry, this time for the central directory, finds the stub for the directory class, and returns it.

- Now we have code to receive messages locally, to setup a local registry that others can connect to, to connect to others' registries, to locate and call their message servers, and as well, to connect to a local directory where we can add our details and find out the details of others.
- Please review the `main` method before you start. It primarily just handles inputs and outputs.
- Also note down your local IP address. Run `cmd` and type `ipconfig` (look for the value on the `IPv4 Address` line). (If you are running this at home, you can just set localhost, don't worry.)
- Finally try running the `InstantMessagingApp`: first set the argument `-n` `DIRECTORY HOSTNAME` where the location will be given on the board (if you are running this at home, the default is localhost, you don't need to change anything.). Then run the application. Enter a username, your real name, your IP address and a port. Add yourself to the directory, get the list of names from the directory, try message some users (... if you're not the first; if you are, you can still message yourself).
- OPTIONAL: In `messageUser(.,.,.)` it opens a connection every time you message. Maybe you can cache connections instead?
- QUESTIONS: Let's say you sent a nasty message about me to someone. Could I read that from the central server later and hurt my feelings? What effect would it have if the central server died? What effect would it have if a peer died? What happens as the number of peers grows?
- Submit the `InstantMessagingApp` and `InstantMessagingServer` classes in a zip to ucursos, deadline on Monday.