Distributed Systems Labs

Graham White

Week 3, January 2017

The goal of this lab will be to set up, and run, a Hello World program using RMI. There is very little programming, but the configuration details are somewhat tricky and involved, and unless you get them right it will not work. We will be running this from machine to machine in the ITL, and because of this we will have to use port numbers in the range 20000 and up: we choose (merely as an example) port 20005, and this number occurs in the code several times. If you want to change the port number, then you have to change it in those places. (By default, `rmiregistry` uses port 1099, but this will not work in the ITL.)

1. Get yourself set up in the ITL, and choose two machines, one for your client and one for your server. You don't have to sit at the server: just log in to it using `ssh`

2. Download the code: you should have

   (a) The interface for the server class, `HelloInterface.java`

   (b) A java snippet, `mainmethod.java`, which should be the main method of your server class. Note that several lines in here are commented out: if you uncomment them, you will get debugging information. You will have to fill in the hostname of your server machine at one place.

   (c) A java policy file, `server.policy`. The path after the `file:/` on the first line should be replaced by the path to the directory on the server where everything ends up. You will have to fill in the IP address of one of your machines in the `SocketPermission` line (the `/16` after it says that only the first 16 bits are significant, so it doesn't really matter which machine you pick).

3. Write a file `Hello.java` which should implement `HelloInterface`; it should have the following imports

   ```
   import java.rmi.*;
   import java.rmi.registry.*;
   import java.rmi.server.*;
   ```

   The method `say()` should return `Hello World` Cut and paste into it the code from `mainmethod.java`; again, this needs a little customisation.

4. Write a client class `HelloClient.java`; the magic line

```
HelloInterface hello = (HelloInterface)
  Naming.lookup (''//hostname of my server:20005/Hello'');
```

will give you a `Hello` object, and calling `hello.say()` will return the string `Hello World`, which you can print.

5. Now put the files `Hello.java`, `HelloInterface.java`, and `server.policy` in a convenient place on your server. Compile the java files.

6. Put the `HelloClient.java` in a convenient place on your client, together with the `HelloInterface.class` which you got by compiling the java files on the server.

7. On the server, run rmiregistry in the background/as a service (`rmiregistry 20005 &` in Linux). Start the server: you will have to run it with a command like

   ```
   java -Djava.security.policy=server.policy Hello
   ```

   to give yourself the right permissions.

8. On the client, run the client program. You should see output. If you see error messages, start debugging.

9. Once you've done this, modify it so that the client will give the `say` method an argument, which is the name of a language. The server should respond with `Hello World` in that language.