Distributed Systems

Spyros Voulgaris

Lab Assignment 1: Sockets and RMI

Grade: **10%**

Deadline: **Monday 20 October (2 weeks time)**

IMPORTANT: This assignment is intended to help you learn, not to scare you away ! ☺ And I am here to help you in learning. So, take advantage, don't hesitate, and ask me questions!!

Goal

Communication between processes is the cornerstone of distributed computing. In this lab assignment you will gain acquire experience with two types of communication in Java: Sockets (TCP) and RMI.

Task

You are asked to implement a very simple FTP server and client.

The **server** (applies to both TCP sockets and RMI) should take the list of available files as command-line arguments.
For example:
```
      java tcp.FileServer picture.jpg exercise.doc test.txt
```
or:
```
      java rmi.FileServer picture.jpg exercise.doc test.txt
```

The **client** should take the server address and port as command-line arguments.
For example:
```
      java tcp.FileClient 127.0.0.1 5555
```
or:
```
      java rmi.FileClient mydesktop.ifi.uzh.ch 5555
```

Then, the client should wait for user input from STDIN, and should support the following three commands:
- `dir` --- retrieves the list of files offered, and prints them on the screen
- `get <filename>` --- retrieves the file with the given filename
- `bye` --- closes the communication with the server and exits

Any other input should be ignored (not exiting!) with a simple "syntax error" message printed on the screen.

Socket interface

For the TCP sockets implementation, your client should connect to the server, and send it a String (use ObjectInputStream and ObjectOutputStream) containing one of the following commands, and expecting the respective responses from the server:
- `dir` --- The server sends back a String[] containing the filenames of available files
- `blocks filename` (for instance: **`blocks picture.jpg`**) --- The server sends back an integer (use `ObjectOutputStream.writeInt()`) denoting the number of 1024-byte blocks in that file. For instance, if a file has 5000 bytes, it has 5 blocks (4 blocks of 1024 bytes and a final block of 904 bytes)
- `get filename block` (for instance: **`get picture.jpg 4`**) --- The server sends back a 1024-byte-long block of the file. The last block of the file may have less than 1024 bytes.

RMI interface

For the RMI implementation, your server should support the following simple API:

- `String[] dir()` --- returns an array of String containing the filenames of all available files
- `int blocks(String filename)` --- returns the number of blocks that the given file has
- `byte[] getBlock(filename, blockNumber)` --- returns a 1024-byte-long block of the file. The last block of the file may have less than 1024 bytes.

Deliverables

Send me by email (please don't forget "**DS:**" in the subject) a ZIP or TAR file with all your code. Please make sure you add sufficiently enough comments in your code that will allow us to examine it.

Resources

TCP sockets: http://java.sun.com/docs/books/tutorial/networking/sockets/index.html

RMI: http://java.sun.com/j2se/1.3/docs/guide/rmi/getstart.doc.html

I will provide you with help in file reading/writing, if you need. Play honest: give it a try **yourself** first, and **ask me** if you get stuck.

Try to check interoperability with your friends' implementation. E.g., put a couple of funny pics on **your** ftp server, and let your friend download them using **his/her** ftp client, and vice-versa. Promise not to reveal the pictures unless downloaded through your own code!