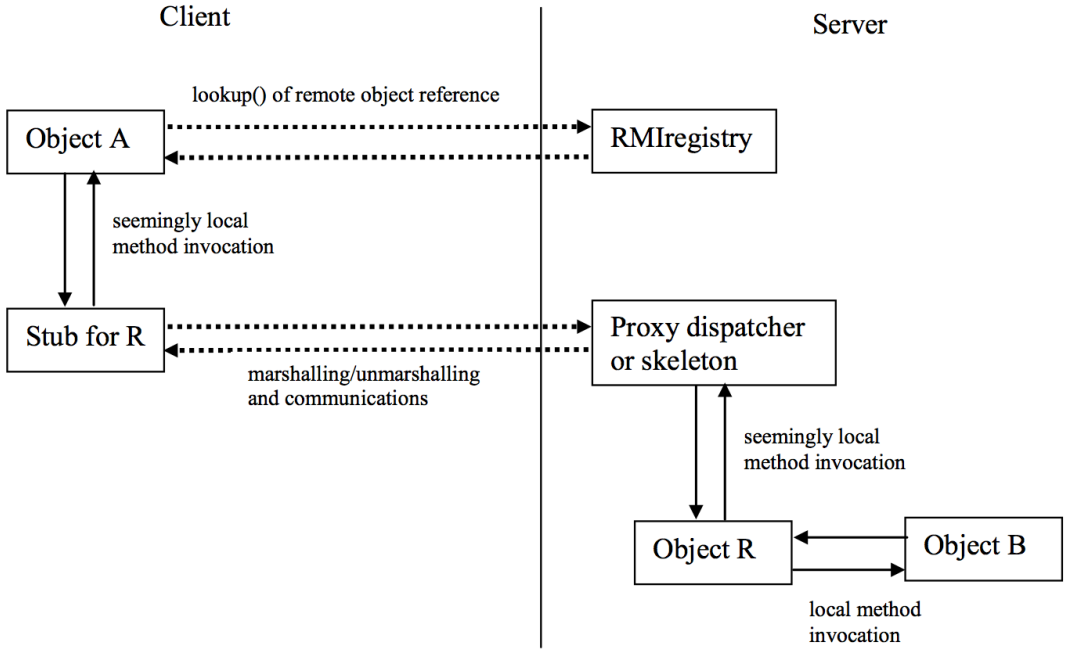
15640 Project 2 Report

Authors: Guanjie Chen (guanjiec)
Wenhan Lu (wenhanl)

1. Design

Our design follows the Java native RMI structure. The components and process can be described as follows:



In order to have better code reuse and project structure, we encapsulate our own "net" package which in charge of all network and concurrency related functions based on I/O multiplexing (which is proven the best practice in network programming) . This package will be reused to our later projects.

What is included in our design:

- Remote object references
- Registry
- Dispatcher
- Stubs
- Server
- Client
- Message

What is not included:

- Stubs Compiler
- Mechanism for downloading .class file
- Garbage collection for remote objects

2. Implementation

2.1 Project File Structure

./bin - Output .class files
./lib - External library. For this project, lombok.jar is used for automatically create getter, setter and constructors for structure classes
Under ./src
Net package
This package is logically encapsulated for concurrent network use. It contains 6 files. The main files are ***Server.java*** and ***Client.java*** which is explained by their name. ***NetObject.java*** is used to abstract network packets, and ***SocketInfo.java*** is an util for Server to track what clients are connected. ***TestServer.java*** and ***TestClient.java*** are unit tests for Server and Client. They perform as echo between client and server.
RMIClient.java
Client file contains all test examples we provided. Run client directly from terminal and you will see the result of the all tests.
RMIMessage.java
The message object which are used to transport messages between servers and clients.
RMIServer.java
Accept the command from clients to invoke a remote method, pass the message to dispatcher and return the result to client.
RMIRegistry.java
Store the information of remote object in server and accept the commands from clients to look up for a remote object, return the remote object reference if the required object exists.
RMIDispathcer.java
Invoke the methods requested by client.
RemoteObjectRef.java
The reference of a remote object
Remote640Stub.java
The parent class of the example_stub. The main function of this class is the invoke method function which will pack the information to invoke a certain method and send the message to server.
Example and Example_stubs
The real objects in server and the stubs in client.

2.2 RMIClient

Client contains the examples to invoke the methods from server. When client try to invoke a remote method, it will first send lookup request to the registry and get the reference of the remote object if it exists. Then it will send invoke request to the server. We include all tests in this file.

2.3 RMIRegistry

The registry will listen from two command: lookup and list. Lookup function will return the reference of an object to client and list will return all existing object names. Registry has two hash tables. One stores the object name and reference, the other stores name and real object. The dispatcher will get the real object from registry.

2.4 RMIServer

Server will start to listen on the incoming request from a client. If it reveive a invoke commands, it means that the registry has already confirmed that the object exists so the server will call dispatcher to invoke the method.

3. Deployment

First, enter the base directory and compile all the .java file, type:

```
./make.sh
```

You can also compile the .java file manually. Type:

```
javac -cp lib/lombok.jar -d bin src/**/*.java src/*.java
```

To clean out all the .class file and .obj file, type:

```
./clear.sh
```

After compilation, you need to go to bin directory

```
cd bin
```

you can start server in one unix machine, type :

```
java RMIServer
```

And the server will start to listen on port. The default server port is 15640 for server and 15440 for registry. Please make sure these port are not been taken.

Then turn to another unix machine and under ./bin, start client typing

```
java RMIClient #Server IP#
```

Please make sure you type the correct ip address of the server. Or the connection may failed.

Now you should be able to see the result.

Additional info:

1. In the client we provide two examples: one is to compute the pow(2,n) and n is set by the users. The second example is to reverse a string and the original string is also set by users. You can set these two variables in the client file.
2. We use a external jar file: lombok.jar. This jar can use simple annotation to include getter and setter which will make our code look clean.

4. Dependencies

Compiler: javac version 1.7
Running Environment: java version 1.7
One additional package: lombok.jar

5. Integration Test

We are sorry not getting time to make a console for testing and wrote all our tests inside one RMIClient file, but it works just well. As long as you run RMIClient, you can find out all test cases listed below passed.

Expected Result on RMIServer:

```
/Library/Java/JavaVirtualMachines/jdk1.7.0_67.jdk/Contents/Home/bin/java ...
Connection Established: On RMIRegistry
Connection Established: On server
Registry LOOKUP reference error: Remote640Exception: ObjectName Example not exist in registry!
Connection reset
Connection reset
```

Expected Result on RMIClient:

```
Start test1: Lookup
Remote Object Received: ExampleOne
Passed!
================================================
Start test2: Failed Lookup (Remote640Exception)
Exception: Remote640Exception: ObjectName Example not exist in registry!
Passed!
================================================
Start test3: List
ExampleTwo
ExampleOne
Passed!
================================================
Start test4: RPC
32.0
IMR :metsyS detubirtsiD
Passed!
```

01. Lookup Test

RemoteObjectRef ref = client.regLookup("ExampleOne");
System.out.println("Remote Object Received: " + ref.getObjectName());

Expected result:

Remote Object Received: ExampleOne

Test result:
Passed.

02. Exception Test
Lookup non-existent object:

```
try {
        RemoteObjectRef ref = client.regLookup("Example");
        System.out.println("Remote Object Received: " + ref.getObjectName());
       } catch (Remote640Exception e) {
        System.out.println("Exception: " + e.getMessage());
}
```

Expected result:

Exception: Remote640Exception: ObjectName Example not exist in registry!

Test result:
Passed.

03. List Test

```
ArrayList<String> list = client.regList();
```

Expected result:

ExampleTwo
ExampleOne

Test result:
Passed.

04. RPC Test
In the RMIClient file we set:

```
        Double test1 = 5.0;
        String   test2 = "Distributed System: RMI";
```

Finish deployment and start server and client. The RMIClient will create stub for ExampleOne and ExampleTwo and call e1.pow(double) and e2.reverse(String) and print output

Expected result:

32.0

IMR :metsyS detubirtsiD

You should see 2^5 and the reverse string of the original sting.

Test result:

Passed.