RMI APPLICATION FOR TRANSFERRING FILES

Nicolae Enescu, Eugen Dumitrascu and Gheorghe Marian
University of Craiova
Faculty of Automation, Computers and Electronics
5, Lapusului Street, Craiova, Romania
Email: {Nicu.Enescu; Eugen.Dumitrascu; Gheorghe.Marian}@comp-craiova.ro

Abstract: This paper deals with the issues involved in distributed applications. In particular, we describe an application that transfers a file between two computers on the Internet, one of them is a Home computer (source) and the other is a Remote computer (destination). A Server realizes the connection. It is important to mention that the addresses of Home and Remote are not known. We have to know only the Server's address. The technology used in this application is Java RMI.

Keywords: RMI (Remote Method Invocation), Java, JVM (Java Virtual Machine), Server, Home, and Remote

1. INTRODUCTION

Remote Method Invocation (RMI) is the object equivalent of Remote Procedure Calls (RPC). While RPC allows you to call procedures over a network, RMI invokes an object's methods over a network.

In the RMI model, the server defines objects that the client can use remotely. The clients can now invoke methods of this remote object as if it were a local object running in the same virtual machine as the client. RMI hides the underlying mechanism of transporting method arguments and return values across the network. In Java-RMI, an argument or return value can be of any primitive Java type or any other Serializable Java objects.

Remote methods are defined by remote interfaces. That is, a remote interface defines a set of methods that can be called remotely. Any object that wants some of its methods to be called remotely must use one or more remote interfaces.

An object that uses a remote interface is called a *server*. An object that calls a remote method is called a *client*. An object can be both a client and a server: These names indicate only who is calling in a particular instance and who is being called.

Once a remote interface is defined and an object that uses the interface is created, it still needs a way for the client to invoke methods on the server. Unfortunately, it is not quite as easy as instantiating a server object.

A *stub* needs to be created for the client. An object's stub is a remote view of that object in that it contains only the remote methods of the object. The stub runs on the client side and is the representative of the remote object in the client's data space.

The client invokes methods on the stub and the stub then invokes the methods on the remote object. This allows any client to invoke remote methods through normal Java method invocation. A stub is also called a proxy.

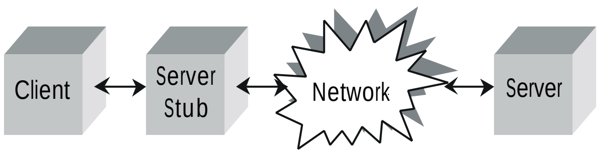


Figure 1: A stub invokes remote methods on behalf of a client.

RMI adds an extra feature that most RPC systems do not have. Remote objects can be passed as parameters in remote method calls. When a remote object is passed as a parameter, a stub is actually passed for the object.

The real object always stays on the machine where it was originally started. The stub that is passed then invokes methods back to the original object. Stubs can also be passed as parameters and work the same way.

In a distributed system, it is necessary to have a way so that clients to find the servers they need. RMI provides a simple name lookup object that allows a client to get a stub for a particular server based on the server's name. The naming service that comes with the RMI system is fairly simplistic but is useful for most cases.

2. THE APPLICATION

The application related in this paper contain three parts: Server application, Home application and Remote application.

The Server runs on a computer that has known IP address, the Home and Remote runs on other computers that have an unknown address. We can connect two computers from anywhere in Internet that they have an unknown IP address.

The Home is connected to Server (which has a known address), and after that the Remote is connected to Server too. In this way the Remote can "see" the Home, and also the Home can "see" the Remote too. After that we can transfer /copy a file from Home to Remote.

The Server participates to make the connection between Remote and Home, it doesn't participates to transfer.

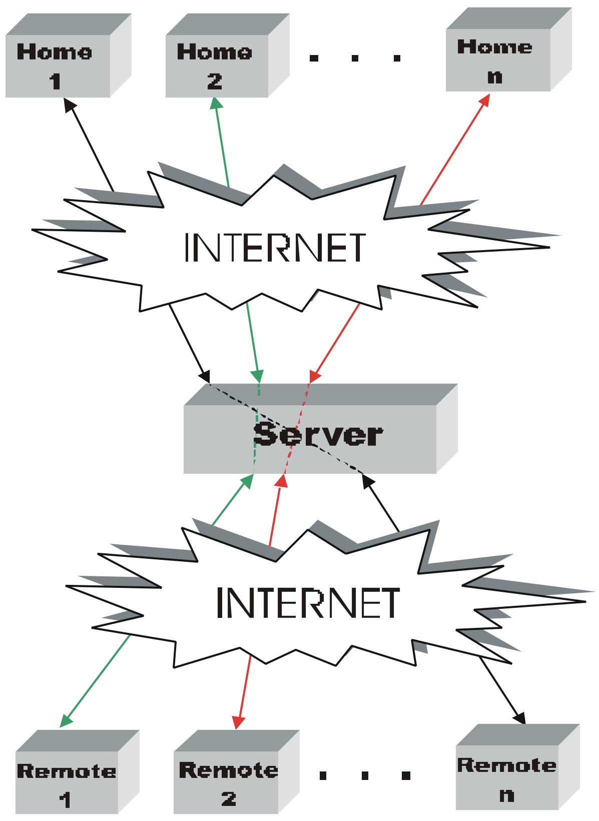


Figure 2: The block schema for the application

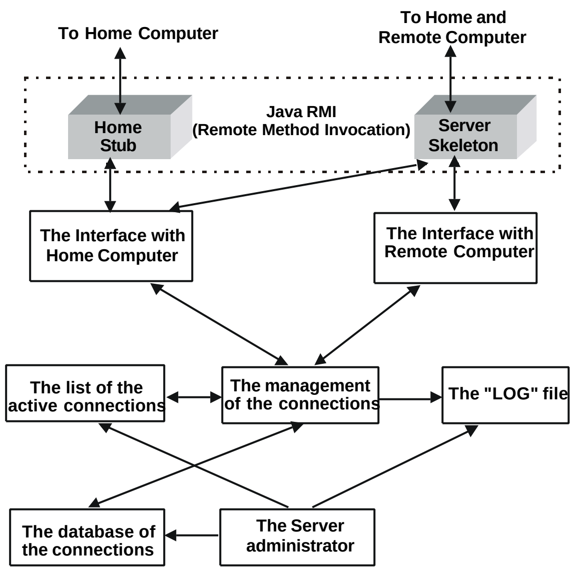
The Server knows all the connections that are done. It has a database in which it retains the connection name, the Home password and the Remote password for the authentication of the Home and the Remote.

We made the following suppositions:

- when a Home initiates a connection, the Server verifies the existence of the connection in the database and accepts that connection or not
- when the Remote wants to connect, first it is verified if the connection was initiated by Home and then it is accepted or not
- the communication between Home and Remote is made through Server.

2.1. Server application

The Server application makes the connection between Home and Remote applications. The following diagram represents the interaction between Server's operations.

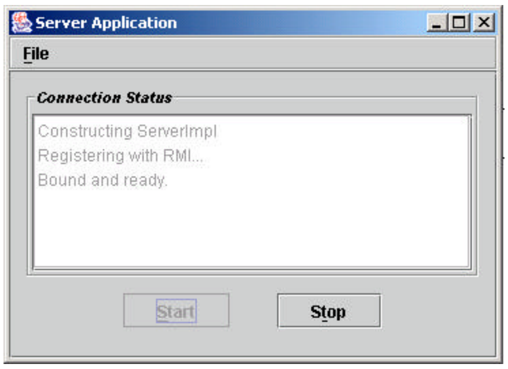


To Home Computer
To Home and Remote Computer
Home Stub
Java RMI (Remote Method Invocation)
Server Skeleton
The Interface with Home Computer
The Interface with Remote Computer
The list of the active connections
The management of the connections
The "LOG" file
The database of the connections
The Server administrator

Figure 3: The block diagram for the Server

"The interface with Home Computer" realizes the maintenance of the communications with the Home Computer, transmits the requests to the Home Computer, and receives the responses from the Home Computer.

"The interface with Remote Computer" realizes the maintenance of the communications with the Remote Computer, receives the requests from the Remote Computer, and transmits the responses to the Remote Computer.



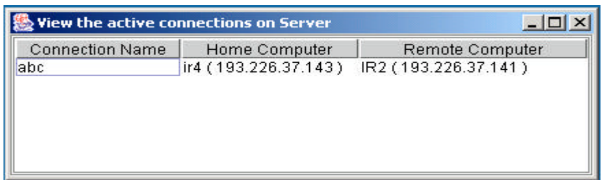
```
Constructing ServerImpl
Registering with RMI...
Bound and ready.
```

Figure 4: Server application

"The management of the connections" authenticates the Home and the Remote, establishes the relations between Home and Remote, and records the connections between Home and Remote in a “log” file.

"The list of the active connections" holds the active connections between Home and Remote Computers

"The database of the connections" is a database with all the possible connections between Home and Remote Computers. It memorizes the name of connection, the password of Home and the password of Remote. If a connection initiated by a Home Computer isn’t in this database then this connection will be rejected.

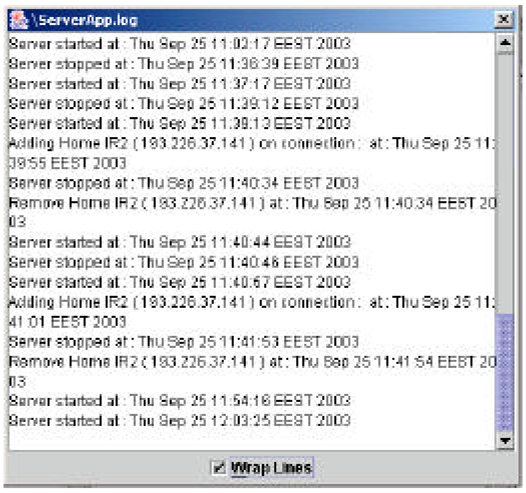


Connection Name	Home Computer	Remote Computer
abc	ir4 (193.226.37.143)	IR2 (193.226.37.141)

Figure 5: Active connections

The "LOG" file is a file in which all the operations between Server, Home and Remote Computers are memorized.

"The Server administrator" manages the database of the connections, and can also visualize both the active connections between Home and Remote Computers and the content of the "log" file.



```
Server started at : Thu Sep 25 11:02:17 EEST 2003
Server stopped at : Thu Sep 25 11:36:39 EEST 2003
Server started at : Thu Sep 25 11:37:17 EEST 2003
Server stopped at : Thu Sep 25 11:39:12 EEST 2003
Server started at : Thu Sep 25 11:39:13 EEST 2003
Adding Home IR2 ( 193.226.37.141 ) on connection :  at : Thu Sep 25 11:
39:55 EEST 2003
Server stopped at : Thu Sep 25 11:40:34 EEST 2003
Remove Home IR2 ( 193.226.37.141 ) at : Thu Sep 25 11:40:34 EEST 20
03
Server started at : Thu Sep 25 11:40:44 EEST 2003
Server stopped at : Thu Sep 25 11:40:46 EEST 2003
Server started at : Thu Sep 25 11:40:57 EEST 2003
Adding Home IR2 ( 193.226.37.141 ) on connection :  at : Thu Sep 25 11:
41:01 EEST 2003
Server stopped at : Thu Sep 25 11:41:53 EEST 2003
Remove Home IR2 ( 193.226.37.141 ) at : Thu Sep 25 11:41:54 EEST 20
03
Server started at : Thu Sep 25 11:54:16 EEST 2003
Server started at : Thu Sep 25 12:03:25 EEST 2003
```

Figure 5: Server log file

2.2. Home application

The Home application runs on a computer that represents the source from where we want to transfer a file. The following diagram represents the interaction between Home's operations.

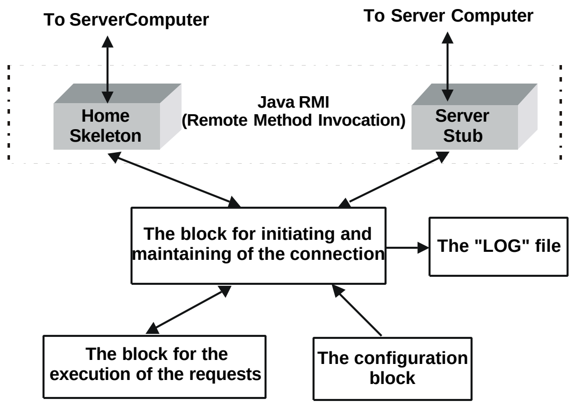
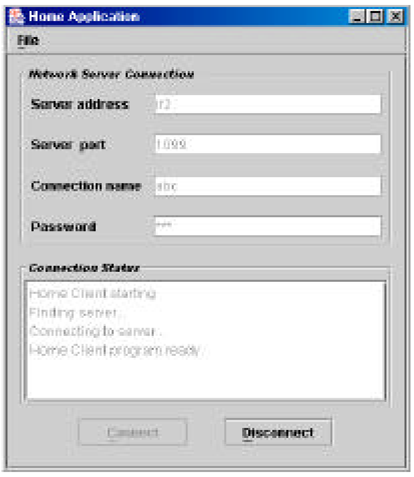

To ServerComputer
To Server Computer
Home Skeleton
Java RMI
(Remote Method Invocation)
Server Stub
The block for initiating and maintaining of the connection
The "LOG" file
The block for the execution of the requests
The configuration block

Figure 6: The block diagram for the Home

"The block for initiating and maintaining of the connection" initiates and maintains the connection with the Server, receives the requests from the Server and transmits them to the block for the execution of the requests in order to solve them, transmits the responses received from the block for the execution of the requests to the Server, records the operations between Home and Server in a "log" file.



File

Network Server Connection

Server address: ir2

Server port: 1099

Connection name: abc

Password: ***

Connection Status

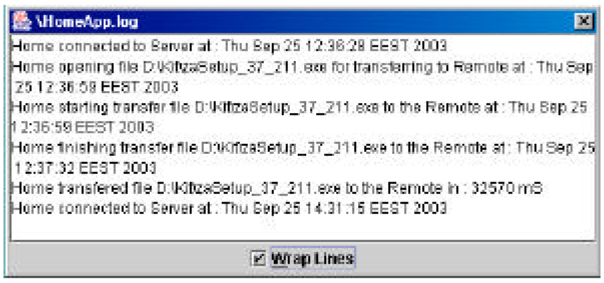
```
Home Client starting
Finding server...
Connecting to server...
Home Client program ready
```

Figure 7: Home application

"The block for the execution of the requests" assumes the requests and solves them establishing the responses

"The configuration block" establishes the connection parameters: server address, port, connection name and password.

The "LOG" file is a file in which all the operations between Home and Server are memorized.



```
Home connected to Server at : Thu Sep 25 12:36:28 EEST 2003
Home opening file D:\KitzaSetup_37_211.exe for transferring to Remote at : Thu Sep
 25 12:36:59 EEST 2003
Home starting transfer file D:\KitzaSetup_37_211.exe to the Remote at : Thu Sep 25
 12:36:59 EEST 2003
Home finishing transfer file D:\KitzaSetup_37_211.exe to the Remote at : Thu Sep 25
 12:37:32 EEST 2003
Home transfered file D:\KitzaSetup_37_211.exe to the Remote in : 32570 mS
Home connected to Server at : Thu Sep 25 14:31:15 EEST 2003
```

Figure 8: Home log file

2.3. Remote client application

The Remote application runs on a computer that represents the destination where we want to transfer the file. The following diagram represents the interaction between Remote's operations.

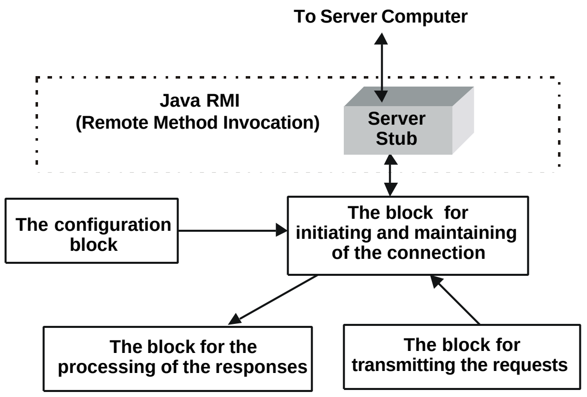
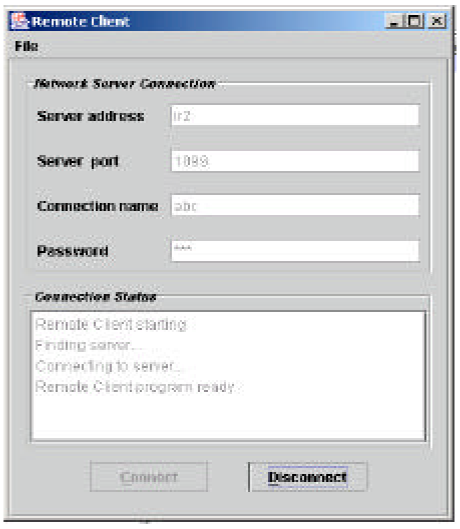

To Server Computer
Java RMI
(Remote Method Invocation)
Server Stub
The configuration block
The block for initiating and maintaining of the connection
The block for the processing of the responses
The block for transmitting the requests

Figure 9: The block diagram for the Remote

The main requests that the Remote transmits to Home are:

- The request for obtaining the tree of the files and directories
- The request for transmitting a file that is specified by name



Server address: lr2
Server port: 1099
Connection name: abc
Password: ***

Connection Status

```
Remote Client starting
Finding server...
Connecting to server...
Remote Client program ready
```

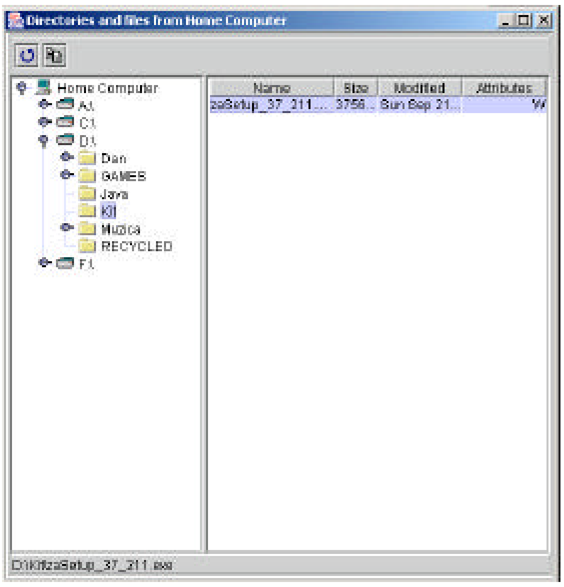
Figure 10: Remote application

"The block for initiating and maintaining of the connection" initiates and maintains the connection with the Server, transmits the requests from the block for the transmitting the requests to the Server, receives the responses from the Server and transmits them to the block for the processing of the responses.

"The configuration block" establishes the following connection parameters: server address, port, connection name and password.

"The block for transmitting the requests" establishes the request that must be solved.

"The block for the processing of the responses" processes the received responses.



Name	Size	Modified	Attributes
zaSetup_37_211...	3756...	Sun Sep 21...	W

D:\Kit\zaSetup_37_211.exe

Figure 10: The View of Home's directories tree from Remote application

REFERENCES

Bloomer J., *Power Programming with RPC*, O'Reilly, 1992

Boian Florin Mircea, *Programarea Distribuita in Internet. Metode si aplicatii*, MicroInformatica, Cluj-Napoca 2000

Crowcroft J., *Open Distributed Systems*, http://www.cs.ucl.ac.uk/staff/jon/ods/ods.html

Darwin Ian, *Java Cookbook*, O'Reilly, 2001

Gibson Brad, *Java Remote Method Invocation (RMI) Tutorial, Implementing a Remote Command Server in RMI*

Grosso William, *Java RMI*, O'Reilly, 2001

IBM Redbooks, *Visualage Java-Rmi-Smalltalk the Atm Sample from A to Z*, IBM Corp, 1999

Rickard Oberg, *Mastering RMI: Developing Enterprise Applications in Java and EJB*, John Wiley & Sons, 2001

Orfali R. and Harkey D., *The Client/Server Programming with Java and CORBA*, John Wiley & Sons, 1997

Pitt Esmond, McNiff Kathleen, McNiff Kathy, *Java(TM).rmi: The Remote Method Invocation Guide*, Addison-Wesley Pub Co, 2001

Qusay H. Mahmoud, *Distributed Programming With Java*, Manning Publications Co., 1999

Roman Ed, Ambler Scott, Jewell Tyler, *Mastering Enterprise Java Beans*, John Wiley and Sons, 2002

Sun Microsystems, *Java Remote Method Invocation*, http://java.sun.com/products/jdk/rmi/

Tanenbaum A.S., *Distributed Operating Systems*, Pretince Hall, 1995

Vaduva Calin Marin, *Programarea in JAVA*, MicroInformatica, Cluj-Napoca 2001

Wutka Mark, et. al, *Java Expert Solution*

Wutka Mark, *Special Edition Using Java 2 Enterprise Edition (J2EE): With JSP, Servlets, EJB 2.0, JNDI, JMS, JDBC, CORBA, XML and RMI*, Que, 2001

*** ftp.javasoft.com/docs

*** http://java.sun.com