# Design and Implementation of Lightweight RMI Framework Based On HTTP Tunnel

Long-da Huang
Power Automation Department
China Electric Power Research Institute
Nanjing, China
huanglongda@epri.sgcc.com.cn

Su-yan Long
Power Automation Department
China Electric Power Research Institute
Nanjing, China
longsuyan@epri.sgcc.com.cn

Jin Liu
Jincheng College
Nanjing University of Aeronautics and Astronautics
Nanjing, China
liujin_jc@nuaa.edu.cn

Jun-song Wang
Institute of Computing & Software
Nanjing College of Information Technology
Nanjing, China
wangjs@njcit.cn

***Abstract*—Java RMI has greatly enhanced the ability to develop java distributed applications, but it is still difficult when passeing through the enterprise firewalls in the public network. Meanwhile, it need extra deployment and registry besides application programs deployment. The paper designed and implemented a lightweight RMI framework, whose communication protocol used HTTP to replace JRMP to solve the above problems.The framework is designed on the base of java proxy pattern and is implemented by the technology of java object tunnel based on *HTTP* protocol and java object serialization. The communication of the framework between the client and the server is through the *HTTP* protocol port, so the framework can easily go through the firewalls of the enterprise. The skeleton in the server-side is implemented through java Servlet technology, and it can conviently be deployed as an popular web application in the web container. The lightweight RMI framework is used well in the application system.**

***Keywords- RMI; Proxy Pattern; HTTP Protocol;Java Object Tunnel; Java Object Serialization;***

## I. INTRODUCTION

With the rapid development of Internet, the world has entered the web-centric age, and distributed computing has become a hot topic in the computer research areas. Java RMI (Remote Method Invocation) has been widely used in the enterprise application development because of the advantages of cross-platform, high portability, security etc. RMI use JRMP (Java Remote Method Protocol), the client-side stub and the server-side skeleton. JRMP act as the media of customer objects and remote objects, which intercepts client requests, passing the call parameters, the request will be entrusted to the skeleton, and finally the return value is passed to the client stub [1-3]. Because this mechanism ignores the specific network details, the network system design has brought great convenience. RMI program development requires strict accordance with the design rules, for instance, after the definition of RMI remote interface, you also need to use rmic tool to generate server skeletons and client stubs, and then implement and register the interface in the server side [4], so the interface can be used in the client side. This series makes the process more complicated to use the traditional RMI, but also it will encounter difficulties in crossing the enterprise firewalls. In order to solve the above problems, a lightweight framework is proposed. It is designed on the base of java proxy pattern, and is implemented by the technology of java object tunnel based on the HTTP protocol and java object serialization. The lightweight RMI framework mainly uses java proxy pattern, java serialization, java Servlet and java reflection etc., which provides a programming interface in the client side, and offers skeleton in the server-side by deploying a web application in the server side. The communication tunnel between the client and the server is not JRMP protocol but HTTP protocol, so the framework can pass through the fireworks in the public network [5].

## II. PROXY PATTERN AND HTTP TUNNEL BASED ON JAVA

### A. *Java Proxy Design Pattern*

Design pattern is a convenient way to implement the software code reuse between different projects and programmers [6]. For instance, MVC design pattern has been widely used on the UI program of JDK and web framework program. The proxy pattern provides a proxy to control access to the target object. Under normal circumstances we have direct access to a target object providing service, and sometimes probably because of the position or the status of the target object, the target object cannot be directly accessed. Proxy object hides the interaction details of the customer and target object, and control client access to the target object [7-8].

Java dynamic proxy provides a simple-to-use framework for the realization of proxy design pattern. Java dynamic proxy mainly involves invoking handler interface and dynamic proxy class *Proxy*. The invoking handler is an interface that a concrete invoking handler class must be implemented. The declared method *invoke* of the invoking handler need be overwritten. The proxy instance is an instance of a proxy class. Each proxy instance has an associated invoking handler object. Through which a proxy interface method call on the proxy instance will be assigned to the *invoke* method of the handler instance, the *invoke* method of the invoking handler's implementing class will be invoked automatically. The invoking handler object handled method call in an appropriate manner, and it returns the result of the proxy instance to the client.

### B. *HTTP Protocol and Java Serialization Object Transfer*

JRMP (Java Remote Method Protocol) is the Java technology-specific protocol for looking up and referencing remote objects. It is a wire level protocol running at the level under Remote Method Invocation (RMI) and over TCP/IP. Currently java RMI uses JRMP as communication protocol. To pass through the firewall of enterprise and reduce the complication of the application deployment and registry, HTTP protocol is used replacing JRMP. HTTP (Hypertext Transfer Protocol) is an application-level protocol for distributed, collaborative, hypermedia information systems [9]. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. Java servlet is a small Java program that runs within a Web server. The Servlet interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. Servlets receive and respond to requests from Web clients across HTTP. Servlet interface provide a simple-to-use and effective communication method between the client-side and the server-side in the java platform.

Java serialization refers to the persistence of the class or type of basic data in the data stream, including the file byte stream, the network data flow. Java class enables its serialization function by implementing the *Serializable* interface. Object Serialization supports the encoding of objects and the objects reachable from them, into a stream of bytes. Serialization also supports the complementary reconstruction of the object graph from a stream [10]. Serialization is used for lightweight persistence and for communication via sockets. The default encoding of objects protects private and transient data, and supports the evolution of the classes. Serialization allows the serialized data of an object to be specified independently of the fields of the class and allows those serialized data fields to be written to and read from the stream using the existing protocol to ensure compatibility with the default writing and reading mechanisms. Java *ObjectOutputStream* objects and *ObjectInputStream* objects can write and read the java serialization objects. The data sending-side writes the serialized object through invoking the method *writeObject* of *ObjectOutputStream* class, and then the data receiving-side reads the serialized object by calling the method *readObject* of *ObjectInputStream* class.

## III. LIGHTWEIGHT RMI FRAMEWORK BASED ON PROXY PATTERN AND HTTP TUNNEL

### A. *Lightweight RMI framework design*

To improve the capability of passing through the enterprise firewall in the public network and reduce the complexity of deploying RMI service in the server-side, HTTP protocol is used to replace JRMP protocol as the communication protocol between the client and the server. Java client serialize the java object, using HTTP protocol write the serializable object into the byte stream and spread it on the Internet through the firewall in the HTTP tunnel, and submit to the Web Server in the HTTP port. Servlet in the web container read the serialized object by deserializing the java object. After completing the business logic operation, the servlet write the result object into the byte stream. The client gets the result object by deserializing the result. Serialized object transfer rules can be defined on the basis of HTTP protocol.

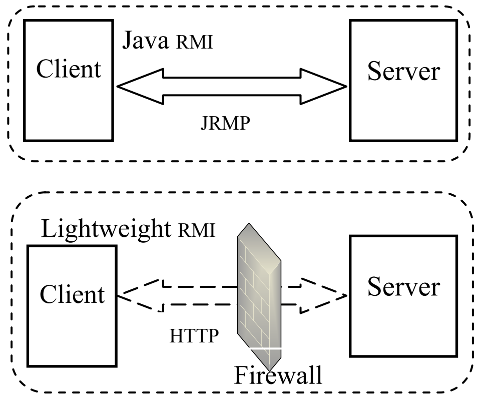

Figure 1. RMI vs Lightweight RMI

HTTP protocol is a stateless request/response protocol, and we need to create a new connection for each request. Java dynamic proxy is used to create a client proxy for the remote server object. When this method is called each time the client side proxy object will establish an HTTP connection between java dynamic proxy of client-side and the server-side. The result is returned to the client proxy object after the remote server object executes the relevant operations. Java dynamic proxy instance plays a role of the client RMI stubs, and java servlet plays a role of the service side of the framework. The framework mainly consists of server-side skeleton, a client stub and HTTP communication tunnel and remote object, shown in figure 2.

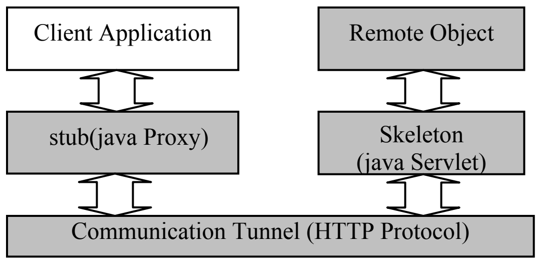

Figure 2. Lightweight RMI framework architecture

1) Remote object: the server-side business logic object that implements the business logic interface, the client application will program oriented-interface after the client proxy instance is created.

2) The service side framework: Java Servlet, after receiving a call request of the client proxy instance, create business logic objects and calls the corresponding method, and if the exception is thrown during the return to the client agent, the call result will be returned to the client;

3) The client stub: the client proxy instance, JDK dynamic proxy classes Proxy create the instance;

4) HTTP communication tunnel: Client handler is invoked to establish HTTP connection with server Servlet, and write java serialized objects connected to the output stream and read java serialized object from the input stream, thereby achieving the communication between the client HTTP proxy instance and server.

## *B. Communication structure based on HTTP*

1) When the client-side proxy instance writes the data objects, the writing order need predefined to communicate properly with the server-side. The order of the client-side writing is defined as follows: ①business interface full name, ②constructor parameter number of mapping class, ③constructor parameter objects of mapping class, ④invoking method name of the interface, ⑤ invoking method parameter number, ⑥invoking method parameter objects.

2) When the server-side servlet writes the data objects, the writing order also need predefined to communicate with the client. The order of the server-side writing is defined as follows: ① result flag object, ②result object.

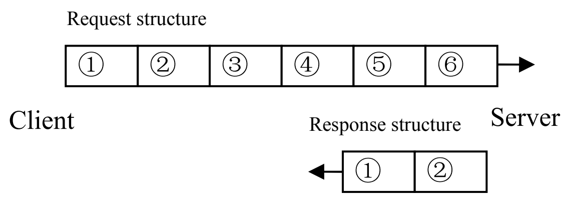

Figure 3. Communication structure based on HTTP

## *C. RMI framework mechanism*

The core of the framework design is to create a handler interface based on the HTTP protocol communication through java dynamic proxy class, on the base of which a proxy instance is created. The mechanism of the lightweight RMI framework can be described as follows:

1) Establish the connection between the client proxy instance and server Servlet. HTTP client component of Aapche open source project is used in the client-side;

2) Client proxy instance will write the interface name, interface class constructor function, the number of parameter object , the name of the called method , the number of parameters and parameter types , parameter object data to the output stream (see section *Communication structure based on HTTP*), waiting for the server response;

3) Servlet will receive the request input stream of the client instance and read the input stream according to the order that the client side wrote the object. The server side application will dynamically create the instance of the class implementing the interface according to the configuration file of the mapping between the interface name and the implementation class name, depending on the java reflection mechanism. The server side will return the result to the client side through the response method. The result mainly includes method call result flag and the returned object. If the server program throws the exception, the exception will be returned to the client. The communication mechanism from the server side to the client side is the same with the mechanism from the client to the server side.

4) The client agent reads the returned result flag. If the flag is successful, the subsequent returned object can be read, otherwise the subsequent object is an instance of exception object, and the client-side should catch the exception;

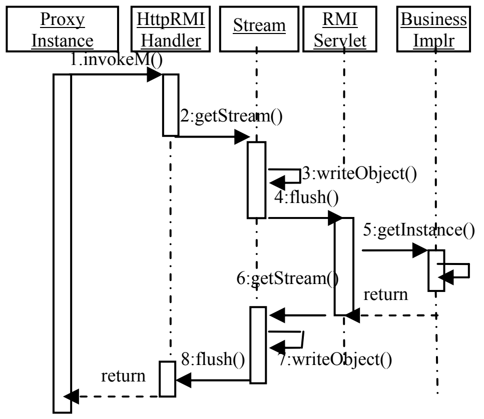

Figure 4. Lightweight RMI framework mechanism

## *D. RMI framework implementation*

The client proxy instance based on Java dynamic proxy *Proxy* is created, create a class that implements the invoking handler interface to achieve HTTP connection with the Servlet server. The object sent to the output stream is written and then waits for the server response. The Servlet of the server side receives client HTTP requests, mainly to resolve the request data stream object, and call the appropriate method and returns the results. The static view of the relevant class for the framework can be stated as follows:

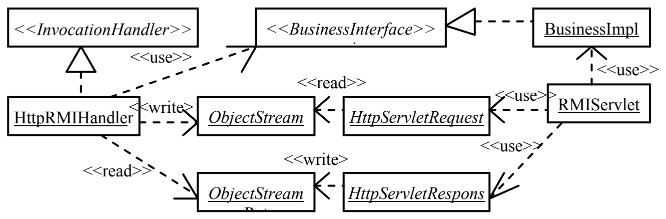

Figure 5. Lightweight RMI framework implementation

1) Define business logic interface *BusinessInterface*, where the provided operation method is declared. Client programming is interface-oriented, which does not concern for the concrete realization of business methods. Business objects in the server side must implement the interface *BusinessImpl*, that is, business methods declared in the interface must be implemented in the server-side. This framework is dynamically loaded server business object class by reading a configuration file mapping business logic interface and business object class.

2) Define RMI invocation handler *HttpRMIHandler*, which must implement *invoke* method. When using the *newProxyInstance* method of java dynamic proxy class *Proxy* to create a client proxy for remote business object instances, *HttpRMIHandler* is passed as an argument. So whenever a client proxy instance call the appropriate method of the business logic interface, the *invoke* method

of *HttpRMIHandler* class will be dynamically invoked, and the method will establish HTTP connection with server *RMIServlet*. After the HTTP connection is established, the business logic interface name, the parameter type of the interface class constructor and the parameter object, the name of the called method , parameter types and parameter objects are written to the output stream in the defined order, waiting for server response; If you read the first successful flag object, then read the subsequent result object. If the first returned object is failure flag object, then read the subsequent exception object and deal with it in the client side.

3) Define *RMIServlet*. When *RMIServlet* receives the HTTP request sent by the client proxy, it will first read the mapping file from the business logic interface to business logic object, and then in accordance with the client's written order of the objects, parsing the received objects, and then dynamically create the business object instance in the server side. If the server-side operations do not throw any exception, the result success flag and the result object will be written to the *object stream* in turn. If an exception is thrown in the process operation in the server side, the failure flag and the exception will be written to *object stream*, and then the client side can extract the exception object and deal with it.

## IV. CONCLUSIONS

The lightweight RMI framework is designed with java proxy pattern. Meanwhile, the lightweight RMI framework uses HTTP protocol to replace JRMP so that the data communication between the client and the server can pass through the firewall of the enterprise and reduce the complication of the RMI program deployment and registry. Java object tunnel based on Servlet technology and HTTP protocol achieves object message communication, so the lightweight RMI framework can easily be integrated into the application system. It provides for a simple and efficient solution to build distributed applications deployed in the Internet on the java platform. The lightweight RMI framework is used well in the application system.

## REFERENCES

[1] WANG Ye, WANG Chen, ZHANG De-fu. The Comparison Research on Java RMI and CORBA [J]. Computer Engineering and Applications , 2001,21: 96-98.

[2] YAO yue-hua, CHEN Yi-dong, LI Hui-fang. Application Research on the Distributed Object Models of CORBA and RMI [J]. Computer Engineering & Science, 2008, 20(9): 151-153.

[3] LIU Dan, CHENG Xiao, HOU De-lin. A Distributed Architecture Design Based on RMI [J]. Computer Applications and Software, 2007,9:206-208.

[4] LI Zeng-zhi, LI Gang, HAN Dong, WANG Zhi-wen. Based –java Distributed Application Developing Mechanism-RMI [J]. Computer Engineering and Applications, 2000, 9: 132-134.

[5] QIANG Liang, LI Bin, HU Ming-zeng. Research of Network Covert Channel Based on HTTP Protocol. Computer Engineering, 2005, 31(15): 224-225.

[6] JAMES W. COOPER. Java Design Patterns [M]. Boston: Addison-Wesley Professional,2000.

[7] Partha Kuchana. Software Architecture Design Patterns in Java [M]. New York: Auerbach Publications, 2004.

[8] LIU Yong-ping, HAO Zhi-feng, TIAN Xiang, Feng Ying-chang. Electricity Fee Payment System of Bank and Power Supply Enterprise Network Based on Proxy Design Pattern [J]. Computer Engineering, 2007,33(8): 259-261.

[9] Dan Connolly. Hypertext Transfer Protocol -- HTTP/1.1 [EB/OL] . [2014-4-22]. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[10] Oracle and/or its affiliates. Java Object Serialization [EB/OL]. [2014-4-22]. http://docs.oracle.com/javase/8/docs/technotes/guides/serialization/index.html.