wang.chengg@husky.neu.edu
Name: Chenggang Wang

Executive Summary

Assignment Overview

- The purpose of the project is to implement the RPC(Remote Procedure Calls) for server and client to communicate with each other instead of UDP or TCP sockets. The concept of RPC represents a major intellectual breakthrough in distributed computing, with the goal of making the programming of distributed systems look similar, if not identical, to conventional programming to achieve a high level of distribution transparency. To understand the most fundamental design issues for RPC, this project practice make me acknowledge the style of RPC programming to program with interfaces and the transparency of RPC. The implementation of RPC in Java is associated with RMI(Remote Method Invocation) system. It gives me a thorough understandings of client stub procedures, server stub procedures, binding services, and registry. The practice gets us in hand with models, objects, exceptions, and APIs in java.rmi package to establish the connection between server and client.

Technical Impression

- To extend the code of RPC(RMI in Java), I decided to use the interface of Server and Client that I have written in project1. Thus the RMIServer has methods of start, setPort, setService and RMIClient has methods of connect and send. Also, the interface Service and class Store keep functioning in project2. I chose to pass the Store to the RMIService, and keep it in RMIStore as a remote object for clients to invoke functions. The interface RMIService extends the Remote class and RMIStore implements that interface as a remote object to be exported. They have the function process to deal with requests of client and identify each client with its hash code. The RMIServer creates the registry and binds the object. And the RMIClient gets the registry and invokes the process method of the service of the store. Besides this, to make the store more robust, I decided to make it singleton. There is only one store created during the process in case that clients' requests are sent to different stores.
- I've found several useful tutorials to get familiar with APIs provided by java.rmi. The whole procedure and coding are straightforward and transparent. Core part of project 2 is compressed into several lines like createRegistry, export, binding of the server and getRegistry, lookup, invocation of the client. However, I've faced a vital bug in my development environment. The biggest challenge of project2 is to dig out the connection exception thrown by the program. The connection of the server and the client kept throwing exception no matter which method I had tried to fix it. I had used running rmi registry terminal in background rather than creating registry in codes, used naming to export and lookup remote objects, and used System.setProperty to change "java.rmi.server.hostname" to connect to the right host name. These all failed and I was struggling to figure out where I've done wrong. Finally, I found that I could look up host files because the exception told me that the program was trying to connect to a different host rather than localhost causing the timeout exception, using telnet to find that the connection was not established. I found that there was an additional line in hosts file. I use terminal to nano change the file and the problem was solved by then. Therefore, in this programming project, I notice the environment variables are really important in this situation and desires the efforts to be corrected.