****

558 Applied Distributed Computing

Yaqun Yu and Xiao Liang’s Execute Summary for Project 2

****

****

Assignment Overview:

****

For this assignment we are asked to enable our client and server to communicate using Remote Procedure Calls (RPC), which is much easier for communication, instead of socket. Remote Procedure Call (RPC) is a protocol allowing one program to invoke a remote procedure from another computer without understanding network details. Such a remote procedure call is also known as function call or subroutine call. The model RPC uses is the client/server model. We were using java for project 1, so for this one we continue using the same language, and use java RMI for RPC communication. The Java Remote Method Invocation (Java RMI) is a Java API that performs the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage collection. The RPC function can be called concurrently, consequently and can handle mutual exclusion. In the read-in test set, our client can do at least five of each operation: PUT, GET and DELET.

****

Technical Impression:

****

During the implementation, the first step is to understand RPC and java RMI. RMI API comprises of two programs: a server and a client. A classic server creates remote objects, makes references to these objects accessible and waits for clients to invoke methods on these objects. The RMI distributed application uses the RMI Registry to obtain a reference to a remote object. The server calls the registry to associate a name with a remote object. The client looks up the remote object by its name in the server’s registry and then invokes a method on it. So we first created an interface and defined all the functions that will be used by client. Each function throws RemoteException as required. Then we implemented this interface in RMIserver.java, along with the detail definitions for each function. After that we write our client, ensuring that the RMI registry is running and server started, then run one or more clients. We find the most difficult and essential part in this assignment is to understand the protocol's mechanism and pin client with server successfully. What we learned: first, RMI registry is essentially a directory service. Second, the "lookup name" is an arbitrary name registered by the server and looked up by the client to establish communication between two processes. It returns remote object associated with the specified name in the registry. Client call server through server’s IP address, and using interface to achieve correct functions. We overcame the difficulty of get into VM and other minor bugs.