15-440 Project #2

10 October 2013

Vinay Balaji vbalaji@andrew.cmu.edu
Ankit Chheda achheda@andrew.cmu.edu

Design

The design we chose for this lab is very similar to Java's original RMI design.

At least three actors are needed: a client, a server, and a registry.

To begin, the registry listens for all incoming RMI requests, such as list, lookup, bind, and rebind.

The server(s) registers its remote objects by going through a registry messenger, which tells the registry to register a remote object with the specified ID and stub (interface). Server methods invoked remotely can return other remote object references by either already having the remote reference in the class containing said method, or can look up the appropriate reference in the registry.

The client does not query the registry directly either, but rather goes through the same messenger used by the server. Through the messenger, the client queries the registry for a list of remote object references available, and then requests one or more of them. The remote object reference returned has a method to return its stub, from which we can finally make remote method invocations.

The stubs construct an invocation message to send off to the server, which then uses an execution skeleton to actually invoke the method. Th return value (or exception) is packaged into a message back to the stub, which finally provides the method's return value to the client (or throws an exception).

Major Design Decisions

Trade-offs

Problems