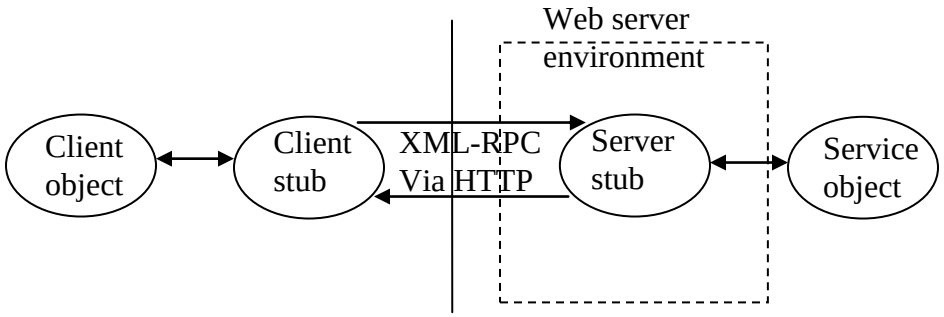
15-440: Distributed Systems
Project #2: Design and Implementation of an XML-RPC-based RPC for Java

Overview

This project asks you to implement an XML-RPC based RPC mechanism for Java.

Requirements

You are to write a tool, *rpcgen*, that given a Java interface specification, creates both the server-side and client-side stubs. The client-side stub provides a local proxy object for the client which communicates with the server-side stub object. The server-side stub provides an interface between the local service-providing object and the network.



In order to achieve this, you'll probably want to define a static method that creates a new instance of the client stub, given the URI of its XML-RPC interface.

Implicit in these requirements is that you must have a mechanism for supporting the HTTP POST requests. This can be your own mini-Web server, or it can integrate with a Web server that can be run in the Andrew environment.

You'll notice that the XML-RPC convention places each service at its own URI. So, on the server side, you'll need a way to map from the URI to the right skeleton object. This might, for example, be by allowing only one remote object of each type and using the class name as part of the URL, or it might be by having remote objects registry, as is done with Java's RMI.

Notes

1) The messages should be marshaled using the XML-RPC spec.
   http://www.xmlrpc.com/spec
2) You may find the XPath, DOM, or SAX libraries helpful.
3) The messages should be transported via HTTP POST.

4) You'll need to be able to parse either the Java interface source file or the compiled Java interface binary file. If you want, you can also explore a solution that dynamically generates the stubs from a live instance. In any case, this is pretty challenging.
5) It is acceptable to serialize all objects sent over the wire, whether as arguments or return values, instead of implementing remote object references