CS6210 - Project 3 - MultiRPC Using XML-RPC

Due 11:59 pm on Nov 1st, 2014

Goal

You are to add the MultiRPC idea, which is explored in the Coda paper, to an existing RPC mechanism. The RPC mechanism in this case is XML-RPC, which uses XML to encode remote procedure calls. You are to implement MultiRPC with both asynchronous and synchronous semantics using XML-RPC. The RPCs should include the ability to specify whether the caller is interested in all responses, any responses, or a majority of responses to be gathered before returning.
You may work in pairs on this project and you are encouraged to do so. You may not share code between groups, though.

XML-RPC Overview

You will first need to get access to the XML-RPC package. You should first check out the XML-RPC web site. The XML-RPC web site has information on what XML-RPC is and how to use it. Next, you should download XML-RPC version 1.06.30, which is the latest stable version. You should then install XML-RPC locally. You can do this as follows:
./configure --prefix=*installation_base_directory*
make
make install

Your XML-RPC library, include files, and binaries will all be installed in *installation_base_directory*/lib, /include, and /bin, respectively. There are examples under the XML-RPC's examples directory that are also built.

Asynchronous MultiRPC

First, you will need to implement an asynchronous call to multiple URLs for your MultiRPC implementation. Your function should be a replacement for the ***xmlrpc_client_call_asynch*** function located in the src/xmlrpc_client_global.c file. Your asynchronous RPC will need to include a variable number of servers as input. The asynchronous RPC must wait until all responses are received before invoking the callback handler.

Synchronous MultiRPC

Next, you will need to implement a synchronous MultiRPC mechanism. Like the asynchronous RPC, you should add in a variable number of server URLs to be accepted by the xmlrpc_client_call function. The xmlrpc_client_call function is also located in xmlrpc_client_global.c . The synchronous MultiRPC mechanism should block until all of its responses are received before allowing the caller to continue.

Any and Majority Semantics

Finally, you should implement semantics for how many responses the RPC should wait on before considering the call complete. The first such semantic is the "*any*" response, which means that the RPC waits for the first response before continuing. The second such semantic is the "*majority*" response, where the RPC call only waits for a majority of responses before continuing. Note that this is important in distributed services, where a majority of the calls must return the ***same*** response before the call is complete. In this case, you do **not** have to wait on a majority of the responses to be the same; your synchronous and asynchronous MultiRPC mechanisms just need to wait for a majority of the responses to arrive.

Write Up

You need to prepare a short write up that describes how you implemented the MultiRPC, how to compile your code, and how to run your code. You should provide at least one example for each call to demonstrate that they work. That means you will need an example for your asynchronous MultiRPC using any, all, and majority return semantics. The same goes for your synchronous MultiRPC.

Deliverables

You must upload your write up and your code in T-square by Sunday, Nov 1st at 11:59 pm.