**Distributed Computing with XML-RPC**

**Description**

Common tasks in distributed computing applications often require the ability of one computer to be able to remotely invoke a procedure on another computer in the distributed system. This assignment introduces this idea further using XML-RPC and Python.

XML-RPC is a protocol used to call procedures, (i.e. methods or functions) by one computer (client) on another computer (server). Its name results from fact that XML is used to encode the procedure calls. The means used to transport the XML from the client to the server is HTTP. While Python has built-in support for this functionality, it is important to note that support is not limited to Python but extends to most high level languagesadd. This is inherent in the design since the encoding is generic XML and transport is HTTP.

Finally, there are many ways to implement such functionality. For Python, while outside the scope of this course, the interested reader should explore projects such as Pyro, RPyC, and Fabric.

XML-RPC’s implementation in Python is found in the xmlrpc package. In this package, the required modules are xmlrpc.client and xmlrpc.server.

**Assignment**

This assignment requires you to develop two Python programs. One is a client, the other is the server.

*Server*

The server should “register” x procedures that the client will be able to call. It will then bind to the address “localhost” and port 8000. This is the address and port that the server will listen to for requests. Note that if you have binding errors, you may use another port as your computer may have an application that is using 8000. Most of the time, however, this will work. Your server invocation must be in the following form:

python server.py localhost 8000

These procedures to be supported are as follows:

name – returns the name of the server which is passed on the commandline during server invocation

help – returns a list of procedures that the server supports

servertime – returns the current time at the server in 24 hour format. I.e. 13:00:01

add(x,y) – returns the sum of x and y

sub(x,y) – returns x – y

mult(x,y) – returns x \* y

div(x,y) – returns x/y (be sure to handle the divid by 0 scenario)

*Client*

The client is to connect to the server using the server’s address and the port that the server is listening on (see above). It then will exercise each of the supported procedures using the values 8 and 6 for the values x and y respectively. Your client invocation must be in the following form:

python client.py host\_address host\_port 8 6

where host\_address and host\_port are the address and port that the server is listening on. If you are using a single computer for server and client computers just use “localhost” for the address and the port used above. The 8 and the 6 are the values for x and y.

**Example Output Of Client:**

8 \* 6 is 48.0

8 / 6 is 1.3333333333333333

8 + 6 is 14.0

8 - 6 is 2.0

8 / 0 is Infinity

13:50:22

**Test your code completely. At each step of the test take a screen shot and embed it into a Word document. At the completion of the test compress (ZIP) the Word document and your code into one ZIP file. Submit the ZIP file any to the appropriate assignment.**

Submit your Python source code for client and server by 11:59 p.m. (ET) on Monday of Module/Week 6.