**IS 651, Distributed Systems**

**Homework #1**

**Due Feb 28, 2020**

**130pts**

**Q1: (30%) Written Problem.**

De-coupling is an important concept in distributed systems. What is de-coupling? What are the benefits of de-coupling?

**Q2:** **(70%)**

Use **BOTH** methods (1) Socket-API, (2) XML-RPC and any programming languages.

References:

* Socket
	+ Python <https://docs.python.org/2/howto/sockets.html>
	+ Java <https://www.geeksforgeeks.org/socket-programming-in-java/>
* XML-RPC
	+ Java <https://www.tutorialspoint.com/xml-rpc/index.htm>
	+ Python2 <https://docs.python.org/2/library/xmlrpclib.html>
	+ Python3 <https://docs.python.org/3/library/xmlrpc.client.html>

**Please submit a report with screenshots, description of your code, and answers to the writing problems. For Q1, you only need to write comments and for Q2 you only need to add as minimum as two lines. For the optional question, please submit your source code with descriptions in comments.**

**Q2-1. (30%) Programming+Written Problem**

Compare the two examples we have seen in class (using RPC and socket-API). Write comments for each block and explain what each block does. For each block in socket-API, explain how RPC works to achieve the same function.

Socket blocks:

import socket

import sys

from io import BytesIO

import struct

import time

portnum = 5000

hostname = 'localhost'

Block 1:

def encode(m):

 (msgtype,content)=m

 buf = BytesIO()

 buf.write(struct.pack('<i',msgtype))

 if msgtype == 0: #string type, can directly write

 buf.write(content)

 else:

 buf.write(struct.pack('<i',int(content)))

 buf.seek(0)

 return buf.read()

Block 2:

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_addr = (hostname,portnum)

sock.connect(server\_addr)

Block 3:

msg = (1,99)

m = encode(msg)

print "Sending message: ", msg

sock.send(m)

reply = sock.recv(1024)

print "Reply: ", reply

sock.close()

Block 4:

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_addr = (hostname,portnum)

sock.connect(server\_addr)

msg = (0,"Hello There")

m = encode(msg)

print "Sending message: ", msg

sock.send(m)

reply = sock.recv(1024)

print "Reply: ", reply

sock.close()

print "done."

**Q2-2. (15%) Programming+Written Problem**

Run the two examples we have seen in class (using RPC and socket-API). Insert the time before client sends requests and after the client receives the reply. Compare the performance(latency) of the two methods for the two functions. (Screenshots of the time should be good enough. Please include the code in your scripts).

**Q2-3. (25%) Written Problem**

Compare the differences of socket-API and XML-RPC. If we are going to implement a distributed system where multiple servers interact with each other, which one is better? Why?

**Optional: (30%)**

Using both socket and RPC, implement a service with the following functionalities: 1) If the client sends a string message, server simply replies with “I have received your message XXX(message content)”. 2) If the client sends a request to add or multiply multiple numbers, server replies with the result.

Examples:

 Client sends: My name is Charlie.

 Server replies: Hi Charlie!

 Client sends: Multiply,[1,2,3,4]

 Server replies: Multiply([1,2,3,4]) = 24 (The format could be flexible. It’s fine if the server only returns 15)

 Client sends: Add,[1,2,3]

 Server replies: Sum([1,2,3]) = 6

***NOTE****: The messages from client or server should be generated by client/server, not printed at the other side.*

Hint: in socket programming, you need to serialize your message to send through the network. Multiple approaches are available besides the one shown in class.