CMSC 327 Distributed Systems

Project 1: Client-Server
Due September 17, 2010

For the first assignment, you will explore with the client-server architecture for distributed systems. You can choose whatever service you'd like to offer. For example, you might build a server program to spell check a word or hold a conversation – it's up to you. The primary objectives of this assignment are to become familiar with network programming and evaluation. Each student should submit their own code, but it's fine for your clients and servers to interact with each other (i.e. use each others' services).

Implementation

Submit your code as two separate programs called server.py and client.py (assuming you are using python). Your server should fully implement your desired service, and your client code should be instrumented to collect latency data. Both programs should be properly documented and the server program should include a description of the service it is providing.

Evaluation

You will evaluate the latency of your distributed service. You should collect data concerning the latency of the interaction when the client and server are colocated on the same machine and when they are on different machines. In both scenarios, vary either the input payload or output payload and investigate how the latency is impacted. The data should be collected and visualized effectively (I recommend gnuplot). Write a paragraph or two reflecting on the results.

Learning Objectives

- Use TCP sockets
- Implement a Client/Server System
- Empirically Evaluate Software

Deliverables

Submit a zip file with the following directory structure:

`cmsc327_proj1_LASTNAME_FIRSTNAME/`

`src/server.py`	server program
`src/client.py`	client program
`README`	simple text file explaining how to run your code
`results.pdf`	your evaluation

Python Client-Server Example

I'd recommend using Python for this assignment, but it's up to you. Below are basic client-server programs in Python taken from the documentation[1]. Moreover, the `clock` function in the time module is useful for latency measurements.

[1] http://docs.python.org/library/socketserver.html

```
#### server.py

import SocketServer

class MyTCPHandler(SocketServer.BaseRequestHandler):
    """
    The RequestHandler class for our server.

    It is instantiated once per connection to the server, and must
    override the handle() method to implement communication to the
    client.
    """

    def handle(self):
        # self.request is the TCP socket connected to the client
        self.data = self.request.recv(1024).strip()
        print "%s wrote:" % self.client_address[0]
        print self.data
        # just send back the same data, but upper-cased
        self.request.send(self.data.upper())

if __name__ == "__main__":
    HOST, PORT = "localhost", 9999

    # Create the server, binding to localhost on port 9999
    server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)

    # Activate the server; this will keep running until you
    # interrupt the program with Ctrl-C
    server.serve_forever()

#### client.py

import socket, sys

HOST, PORT = "localhost", 9999
data = " ".join(sys.argv[1:])

# Create a socket (SOCK_STREAM means a TCP socket)
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to server and send data
sock.connect((HOST, PORT))
sock.send(data + "\n")

# Receive data from the server and shut down
received = sock.recv(1024)
sock.close()

print "Sent:     %s" % data
print "Received: %s" % received
```