CS317 – Computer Networks

Socket Programming

Due Date:________________________

Purpose:

Learn application layer protocols in Internet protocol stack, and socket programming by implementing a client/server application.

Description:

The *Addressbook* service is a client-server application. A client sends a query message to the server containing an email address. The server responds to the client with a message that contains the full name which corresponds to the email address.

A client process *Addressbook_client* submits query messages to the server. The query message contains an email address.

The server process *Addressbook_Server* runs on a known host (say 'kim-cs317.dwc.edu ') and listens on a well-known port number for client requests. When a request comes in, the server looks in a file to find the full name which corresponds to the email address, and sends the full name to the requesting client.

The following message format is used for query and response:

Message Type (1 byte)
AString Length (1 byte)
AString (≤ 255 bytes)

Message Type is set to "Q" (ASCII 81) for queries, and "R" (ASCII 82) for responses.

AString Length is an unsigned 8-bit integer.

AString is a character string with up to 255 characters.

Requirements:

- *Addressbook_Server* and *Addressbook_Client* must be able to run on different machines on the Internet.
- The database of the server must be able to handle an arbitrary number of entries.
- The server must be able to process multiple queries. You get full credit if incoming queries are processed sequentially.
- You get 10% extra credit if your server can handle multiple concurrent clients.

Task:

1. Both server and clients are to be implemented exclusively with sockets (Internet domain, stream sockets).

2. Test your client implementation with the server specified in the requirements.

3. Deliver a write-up which describes your implementation approach, a description of your test strategy, and known bug list.

Hints:

1. Implement the server process as a background process (``daemon''), which is *listen*ing on a well-known port number.

2. A client process contacts the server by connecting to the well-known port number on the server machine.

3. When you transmit messages, make sure that you only send the required text string.

4. The server process *accept*s the connection request and then handles the *Addressbook* request.

(*Note:* Alternatively, the server process may *create* a new thread process that handles the request using a new socket (with a number that is different from the well-known port number.)