CS 436

Assignment 1

Introductory Socket Programming

Due Date: Monday, February 25th, 2019, at midnight (11:59 PM)

Work on this assignment is to be completed individually

1 Assignment Objective

The goal of this assignment is to gain experience with TCP socket programming in a client-server environment (Figure 1). You will design a client/server protocol with **out-of-band-control** and implement the client program (`client`) and the server program (`server`) to communicate between themselves. Python, Java and other languages are accepted as long as it is support by the undergrad environment `linux.student.cs.uwaterloo.ca`

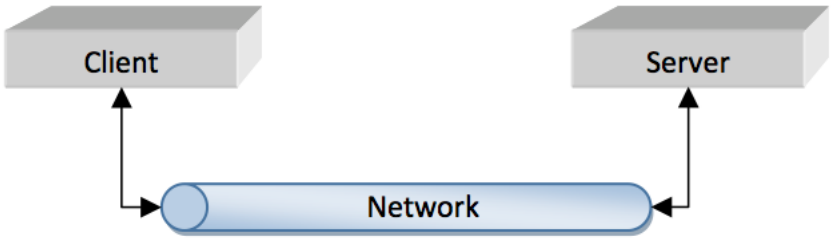

FIGURE 1

2 Assignment Specifications

2.1 Summary

In this assignment, the client will send requests to and receive responses from the server over the network using sockets.

This assignment uses a two-stage communication process. In the *negotiation stage*, the client connects to the server to request a command (`<command>`) for later processing a string over a given negotiation port (`<n_port>`) of the server. Later in the *transaction stage*, the server connects to the client on a transaction port (`<r_port>`) for actual data exchange.

2.2 Communication

Communication in this project is done in two stages.

Stage 1.

Negotiation using a persistent TCP connection (control connection): In this stage, the server creates a server socket on the local port number `<n_port>` that we will call negotiation port number. The client creates a TCP socket using `<server_address>` as the server address and `<n_port>` as the negotiation port number (where the server is listening). After connecting to the server, the client sends a request (`<command>`) read from the keyboard. For simplicity, the client will send "GET `<filename>`" or "PUT `<filename>`", for downloading or uploading a file `<filename>` respectively to/from the current folder. For simplicity use text files, no bigger than 5KB.

Once the server gets this request in the negotiation port, it will reply back with "OK".

The client creates then a server socket on an available local port number `<r_port>` that we will call transaction port number and sends this port number over the control connection along with its address `<client_address>`.

Stage 2.

Transaction using a non-persistent TCP connection (data connection): In this stage, the server creates a TCP socket and binds it to the remote `<client_address>` and `<r_port>`. After accepting the connection, the transfer of `<filename>` begins. The file must be saved locally, and the transaction connection closed after file transfer. The client loops back to Stage1.

2.3 End of communication

The user can end the client process by typing "EXIT" in Stage1. When "EXIT" is read the client closes the control connection to the server.

2.4 Client Program (`client`)

You should implement a client program, named `client`. It will take the command line inputs: `<server_address>` and `<n_port>` in the given order.

2.5 Server Program (`server`)

You should also implement a server program, named `server`. It will take the command line inputs: `<n_port>`.

2.6 Example Execution

On **ubuntu1604-002.student.cs.uwaterloo.ca** :

```
server 6789
```

On **ubuntu1604-006.student.cs.uwaterloo.ca** :

```
client ubuntu1604-002.student.cs.uwaterloo.ca 6789
```

3 Hints

You can use the sample codes of TCP socket programming in Python from the **Socket** slides on LEARN.

Below are some points to remember while coding/debugging to avoid trivial problems.

- Use port id greater than 1024, since ports 0-1023 are already reserved for different purposes (e.g., HTTP @ 80, SMTP @ 25)
- If there are problems establishing connections, check whether any of the computers running the server and the client is behind a firewall or not. If yes, allow your programs to communicate by configuring your firewall software.
- Make sure that the server is running before you run the client.
- If both the server and the client are running in the same system, 127.0.0.1 or localhost can be used as the destination host address.
- You can use help on socket programming from any book or from the Internet, if you properly refer to the source in your programs. But remember, you cannot share your program or work with any other student.

4 Procedures

4.1 Due Date

The assignment is due on Monday, February 25th 2019, at midnight (11:59 PM).

4.2 Hand in Instructions

Submit all your files in a single compressed file (YourStudentIDA1.zip, .tar etc.) using LEARN in dedicated Dropbox.

You must hand in the following files / documents:

- *Source code* files.

- *README* file: this file ***must*** contain instructions on how to run your program, which undergrad machines your program was built and tested on.

Your implementation will be tested on the machines available in the **undergrad environment**

4.3 Documentation

Since there is no external documentation required for this assignment, you are expected to have a reasonable amount of internal code documentation (to help the markers read your code).

You **will** lose marks if your code is unreadable, sloppy, or not inefficient.

4.4 Evaluation

Work on this assignment is to be completed individually.