CS 3251 - Computer Networks I
Programming Assignment 1 - Sockets Basics (Client for Relational Database)

Introduction

In this assignment you will design and write your own client-server application using network sockets. The first goal of the assignment is to provide you some experience with simple socket programming. The second goal is to prepare you for the second programming assignment, which will be significantly more challenging.

Assignment

You will implement two different versions of a client-server application. One version will be based on TCP. The other will use UDP. You will implement both the client and server sides of the application. You can use Python, Java or C/C++.

The application you are developing is a client for a Relational Database Server. The server will start in passive mode listening at a specified port for messages from clients. The server command should be called "dbengine" and will be given a port number as the only command-line argument. The client command should be called "dbclient". Your client's command line should include the server IP address, port and query.

Consider the following relational database of Georgia Tech students. Note that GPA is a floating point number and equal to quality_points/gpa_hours.

ID	**first_name**	**last_name**	**quality_points**	**gpa_hours**	**gpa**
903076259	Anthony	Peterson	231	63	3.666667
903084074	Richard	Harris	236	66	3.575758
903077650	Joe	Miller	224	65	3.446154
903083691	Todd	Collins	218	56	3.892857
903082265	Laura	Stewart	207	64	3.234375
903075951	Marie	Cox	246	63	3.904762
903084336	Stephen	Baker	234	66	3.545455

The last argument of the "dbclient" query, will be the ID of the student followed by one or more column name(s). For instance, the following command would start the client, and produce the following output:

```
dbclientTCP 127.0.0.1:8591 903084336 first_name last_name
From server: first_name: Stephen, last_name: Baker
```

In this example, the client's query is sent to the server application running at the same local host as the client (represented by the special IP address 127.0.0.1), listening at port 8591. Your server should be able to support any combination of column names. An error message should be returned if the student ID is not present in this database or if the client refers to a non-existing attribute (column). Feel free to hard code the given database snapshot in your server code or store it in a file and read it when the server starts.

You will implement two versions of the assignment: one using TCP and another using UDP. The TCP application should close the connection after the response is received. The UDP client should set a timeout for receiving a response. If the client doesn't receive a response within 2 seconds, it should retry the same query 3 times. After 3 unsuccessful requests, the client should print an error message and exit. For instance, the following command would start the client, and have the following output:

```
dbclientUDP 127.0.0.1:8591 903084336 gpa
The server has not answered in the last two seconds.
retrying...
From server: gpa: 3.545455
```

To test this functionality try executing the UDP client without having started the server.

Note: The command line must work exactly this way to make sure the TAs can easily test your program.

Submission

You will implement two complete, separate versions of this assignment: a client and a server using TCP, and a client and a server using UDP. Your final submission should include all the source code and instructions to produce 4 separate executables from the provided source code. For simplicity in grading, please call them: dbclientTCP, dbengineTCP, dbclientUDP and dbengineUDP. **Your programs are to be written in Python, Java or C and we should be able to run them at the CoC educational labs.**

Please turn in well-documented source code, a README file and a sample output file called Sample.txt. The README file must contain:

- Your Name and email address
- Class name, date and assignment title
- Names and descriptions of all files submitted
- Detailed instructions for compiling and running your client and server programs
- Any known bugs or limitations of your program

You must submit your program files online.

- Create a ZIP archive of your entire submission.
- Use T-Square to submit your complete submission package as an attachment.

An example submission may look like as follows-

```
pa1.zip
  | -- pa1/
        | -- dbclientTCP.py
        | -- dbengineTCP.py
        | -- dbclientUDP.py
        | -- dbengineUDP.py
        | -- dbengineUDP.java
        | -- dbengineTCP.java
        | -- README.txt/.pdf
        | -- sample.txt
```

Grading

The grading will be divided as follows:

- 20% Documentation
- 40% TCP application
- 40% UDP application

You will get 100% of the points if:

- You can successfully exchange messages using both TCP and UDP.
- You cover correctly all error cases (including unexpected termination of the client or server, host order/network order transformations, etc).
- Your TCP and UDP applications behave as expected.
- Your code is well documented.
- The TA can successfully run your code.

Bonus points

We encourage you to develop the application in two different languages. In that case, please submit an extra version of the client or server to gain 20% extra points. If you do so, your "extra" client or server application should be able to communicate with your "standard" server or client, respectively.

For instance, suppose that your original server application is written in python:

```
python dbengineTCP.py 8591
```

And your extra client application is written in JAVA:

```
java dbclientTCP 127.0.0.1:8591 903076259 first_name last_name
quality_points gpa_hours gpa
From server: first_name: Anthony, last_name: Peterson,
quality_points: 231, gpa_hours: 63, gpa: 3.666667
```

If they intercommunicate successfully you will increase your grade by 20%.

Don't forget to add both the TCP and UDP implementation of your "extra" applications in the zip package.

Changes:

- Timeout only in case of UDP, not in case of TCP
- Submit only source code and no executables
- Added example submission folder structure
- Deadline is 9PM (not 11:59 PM)
- Do not use libraries such as json/xml (talk to a TA if you are using any encoding/decoding library to avoid any confusion)
- Do not forget to add the file (if any) that you may have used to import data into the server