AUBURN UNIVERSITY

M2: Programming Assignment

Objective of this assignment:

- To get you familiar with developing and implementing TCP or UDP sockets.

What you need to do:

1. Implement a simple TCP Client-Server application using your preferred language[1]
2. Implement a simple UDP Client-Server application using your preferred language

Objective:

The objective is to implement a client-server application using a safe method: start from a simple **working** code for the client and the server. You must slowly and carefully *bend* (modify) little by little the client and server alternatively until you achieve your ultimate goal. You must bend and expand each piece alternatively like the way a black-smith forges iron. From time to time save your working client and server such that you can roll-back to the latest working code in case of problems.

For this programming assignment, you are advised to start from the simple echo client and server to implement a very simple application.

Part I: TCP "Reverse" Client-Server

Implement the following Client-Server application that will use two programs: a client program myFirstTCPClient.java and myFirstTCPServer.java

a) **Client: myFirstTCPClient.java**

This program must take two **command arguments**: a hostname H and a port number P. The hostname H is a name or a decimal dotted-quad IP address of the server Sv. The port number P to which the server Sv binds is any allowed port number on Tux machines. The range of allowed port numbers on Tux machines is **10010-10200**.

This program must:

1) Create a TCP client socket connected with the server Sv running on the machine with hostname (or IP address) h bound to Port number P.

2) Repeatedly perform the following actions:

i) Prompt the user to enter a sentence S

ii) Send the sentence S to the server Sv. We assume that a string is represented using **UTF-16BE** encoding scheme.

iii) Receive the response from the server

iv) Measure the duration between the time when the sentence S was sent and the time a response was received.

v) Display the following information: the message received and the time expressed in milliseconds.

To implement the client myFirstTCPClient.java, you should consider start with the program *TCPEchoClient.java* (provided on Canvas with this programming assignment). Do not forget to change the name of the class inside the program *TCPEchoClient.java*.

b) **Server: myFirstTCPServer.java**

This program must take one **command argument**: a port number P. The port number P is any allowed port number on Tux machines. The range of allowed port numbers on Tux machines is **10010-10200**.

This program must:

1) Create a TCP server socket

2) Wait for a client to connect, receive a message, display it with the IP address and port # of the client, "*reverse*" the message, display the *reversed* message, and echo back the "*reversed*" message. ***Reversing*** a sentence means to spell it backward word by word. Reversing the sentence "Hello World" yields "World Hello". Sentences will not include punctuation.

To implement the server myFirstTCPServer.java, you should consider start with the program *TCPEchoServer.java* (provided on Canvas with this programming assignment). Do not forget to change the name of the class inside the program *TCPEchoServer.java* if you are using Java.

[1] Already available on Engineering Tux machines.

Part II: UDP "Reverse" Client-Server

Repeat Part I using **UDP** sockets. Call the client and server programs myFirstUDPClient.java and myFirstUDPServer.java, respectively.

To implement the server (respectively, client) myFirstUDPServer.java (respectively, myFirstUDPClient.java), you should consider start with the program *UDPEchoServer.java* (respectively, *UDPEchoClienTimeout.java*) (provided on Canvas with this programming assignment). Do not forget to change the name of the class inside the program if you are using Java.

Programming Language to Use

Feel free to use any programming language as long as:

1) It is already available on Tux machines. The TA will not have install to install any new package to compile and execute your programs

2) your programs can be compiled and executed on Tux machines. **I strongly advise you to check whether the programs you turn in compile and execute correctly on Tux machines.**

Report

- Write a report that will report your results. The report should not exceed half a page.
- Your report must contain the following information:
  - whether the programs work or not (this must be just ONE sentence)
  - the directions to compile and execute your program

What you need to turn in:

- Electronic copy of your source programs (separately standalone)
- Electronic copy of the report (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.

Grading

1) TCP client is worth 20% if it works well: communicates with YOUR team server.
2) TCP client is worth 5% extra if it works well with a working server from any of your classmates.

All other server and clients (TCP server, UDP client, and UDP server) will be graded the same as the TCP client (20% + 5%).