AUBURN UNIVERSITY

Programming Assignment I

Objective of this assignment:

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

What you need to do:

1. Implement a simple TCP Client-Server application
2. Implement a simple UDP Client-Server application
3. Collect and analyze round trip time measurements for each of the above applications.

Objective:

The objective is to implement a simple 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 "Capitalize" 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 is any valid port number where the server Sv is bound to.

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

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.

vi) Collect the round trip time.

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 argument: a port number P. The port number P is any valid port number.

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, "*capitalize*" the message, display the *capitalized* message, and echo back the *capitalized* message. ***Capitalize*** a sentence S means to change any lowercase letter into an uppercase in the sentence S. Capitalizing the sentence "Hello World!" yields "HELLO WORLD!".

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*.

Part II: UDP "Capitalize" 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.

Data collection and analysis

For **each** application (UPD and TCP), report separately the min, average, and max round trip time.

The programs MyFirstTCPEchoClient.java, MyFirstTCPEchoServer.java, MyFirstUDPEchoClientTimeout.java, and MyFirstUDPEchoServer.java all compiled and ran. In order to run the programs, you would first run the TCP/UDP server program, making sure to input a port number as a parameter (an integer between 10010 and 10200 if using the Auburn tux machines). Then, in a separate window, run the client version of the corresponding TCP/UDP server. The parameters for running the client are: name_or_tux_machine_that_server_is_running_on.eng.auburn.edu, the message that you want to send to the server, and then the corresponding port number that you assigned to your server.

We used a sample size of 100 for our time measurements. The TCP average was 0.640089 ms, the TCP maximum was 1.359539 ms, and the TCP minimum was 0.552472 ms. The UDP average was 0.558483 ms, UDP maximum 1.054772 ms, and UDP 0.453883 ms. Across the board, the TCP values are higher than the UDP values which indicates that UDP takes less time to send over the Auburn network. UDP is faster than TCP because TCP takes precautions to make sure that data is received while, UDP attempts to resend if there is corrupted bits. In this case, UDP is more reliable because there are less switches to go through and the medium that it is going over is more reliable because it is WiFi on a campus (LAN).

Report

- Write a report that will report your results. The report should not exceed half a page.
- In addition, 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 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%).