BTN415 Lab 3 – Multithreaded TCP/IP

In this lab, you will interface with a multithreaded TCP/IP chat room. First, you will create a client that will send and receive messages to and from a server. Then, you will create your own chat room server.

# LEARNING OUTCOMES

Upon successful completion of this lab, you will have demonstrated the ability to:

* Create a multithreaded client application
* Create a multithreaded server application
* Implement a reliable data communication using the TCP/IP standards and protocols

For this lab, you should use as a starting point the codes that were discussed during our lectures on TCP/IP Client and TCP/IP Server. These source codes can be downloaded from our course’s Github repository using these two links: <https://github.com/marceljar/BTN415_Labs/blob/main/lab3/client.cpp> and <https://github.com/marceljar/BTN415_Labs/blob/main/lab3/server.cpp> After downloading the source codes, you should modify them in order to achieve what is asked in what follows.

## PART A – [2.0 marks]

## Update the server’s source code in order for it to act as the remote chat room. It should be able to accept up to 20 concurrent connections, receive messages from any one of those connections, and broadcast the received messages to all users. There is no need to identify the senders of each message. Hint: Modify the Run function to send back messages to all active connections.

**Client.cpp**

**Before:**

A computer screen capture

Description automatically generated with medium confidence

**After:**

A computer screen capture

Description automatically generated with medium confidence

**Output:**

**A screenshot of a computer

Description automatically generated with medium confidence**

## PART B – [3.0 marks]

## Update the client’s source code in order for it to communicate with a remote chat room. The remote server expects a client to connect, receive a “Welcome” message, and start sending messages. After receiving a message from one user, the server broadcasts it to all other users that are currently connected. Therefore, after connecting, your client should open a thread to send messages, and another one to listen and print incoming messages from the server. *Hint: Create a tx thread that will be used for calling a function to send messages, and a detached thread to call a function to receive messages. Make sure to use .join() on the tx thread to keep your client running. It is OK if your client receives a number of bogus messages from the server upon quitting.*

**Server.cpp**

**Before:**

**A screenshot of a computer

Description automatically generated**

**A computer screen capture

Description automatically generated with medium confidence**

**After:**

**A computer screen capture

Description automatically generated with medium confidence**

**A computer screen capture

Description automatically generated with medium confidence**

**Output:**

**A screenshot of a computer

Description automatically generated with medium confidence**

# SUBMISSION INSTRUCTIONS

*You only need to submit two source codes, one for the server( part A), and one for the client (part B).*