BTN415 Lab 1 – TCP/IP Clients

In this lab, you will create your own TCP/IP client to connect, send messages, and receive messages to/from a TCP/IP server following a number of different scenarios.

LEARNING OUTCOMES

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

- Create a client application
- Implement a reliable data communication using the TCP/IP standards and protocolFor this lab, you should use as a starting point the code that was discussed during our lecture on TCP/IP Servers. This code can be downloaded from our course's Github repository using this link: https://github.com/marceljar/BTN415_Labs/blob/main/lab1/client.cpp). After downloading the source code, you should modify the code in order to achieve what is asked in what follows.

PART A – [0.5 marks]

Update the source code in order for the client to connect to a remote server with IP 159.203.26.94, listening on port 27010 (spare servers listening on ports 27020 and 27030 are also available, in case the first one is not accessible). If you are experiencing difficulties with these servers, you can try downloading the localhost server I created using this address: http://marceljar.ca/static/executables/server.exe . This localhost server uses the IP 127.0.0.1and port 27000. Note that this is an executable file. So your system might complain that it looks like a virus.

PART B – [1.0 marks]

Update the source code in order for the client to be able to transmit messages containing spaces. *Hint: use strings, c_str, and getline.*

PART C – [2.0 marks]

Update the source code in order for the client to be able to keep transmitting messages to the server (and receiving feedback messages).

PART D– [1.0 marks]

Update the source code in order for the client to be able to keep transmitting messages to the server (and receiving feedback messages) until the client sends a message containing only: "[q]". (i.e., the sequence of characters [, q, and]).

SUBMISSION INSTRUCTIONS

Note that these instructions are in an increasing order of difficulty. You only need to submit one **cpp** file containing your source code, which is the one that solves the most difficult part you have had success or got the closest to the solution. For example, if you finished part D, simply submit your source code for this part. If you got stuck on part C, then submit your source code for this part.