BTN415 Lab 6

Transmitting Arrays, Structs, and Structs with Bitfields

In this lab, you will learn how to transmit different types of data over sockets.

# LEARNING OUTCOMES

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

1. Transmit and receive arrays of floats with a defined size over sockets
2. Transmit and receive regular structs over sockets
3. Transmit and receive structs containing bitfields

# SPECIFICATIONS

In this lab, we will create methods to enhance our **oop\_winsock** library, so that it can send data with different types of formats. Currently, as our starting point (available on Github), this library can only be used to send **char arrays**. The methods that will need to be created are described in what follows.

oop\_winsock\_client::send\_float\_array, oop\_winsock\_server::receive\_float\_array

The **send\_float\_array** method should take as an argument an array of type **float**, copy it into a local **char** array buffer, and send it over a tcp socket. The array of type **float** should have a constant size of **4** (four) elements. This method should not return any values.

The **receive\_float\_array** method should take as an argument an array of type **float**. It expects an array of type **float** with size **4** (four) elements to be received through a socket connection. The received data should be first stored into a **char** array buffer. Following this method should copy the received data into the float array provided as an argument[[1]](#footnote-1). This method should not return any values.

oop\_winsock\_client::send\_packet, oop\_winsock\_server::receive\_packet

struct packet {

int student\_number;

char[32] student\_name;

float student\_gpa;

};

The **send\_packet** method should take as an argument a **packet** struct (defined above). The input packet should be used directly as an argument to the **send() winsock** method. This method should not return any values. *Hint casting*.

The **receive\_packet** method should take as an argument a **packet** struct (defined above). This argument should be passed by reference. The method expects a **packet** struct to be received through a socket connection. The **recv() winsock** should take as an argument the packet passed as an argument In other words, do not use an intermediate **char array buffer**. This method should not return any values. *Hint casting*.

oop\_winsock\_client::send\_bitpacket, oop\_winsock\_server::receive\_bitpacket

struct bitpacket {

int student\_number;

char[32] student\_name;

unsigned char current : 1;

unsigned char doing\_coop : 1;

unsigned char academic\_violations: 1;

unsigned char : 5; //padding

float student\_gpa;

};

The **send\_bitpacket** method should take as an argument a **bitpacket** struct (defined above). The input bitpacket should be used directly as an argument to the **send() winsock** method. This method should not return any values. *Hint casting*. *Hint2: This struct has a total size of 44 bytes.*

The **receive\_bitpacket** method should take as an argument a **bitpacket** struct (defined above). This argument should be passed by reference. This method should first use a **char array buffer** to collect a **bitpacket** transmitted over a **winsock** connection. Then, this method should use pointer arithmetics, together with bitwise operations, to paste the contents of the receiving **char array buffer** into the **bitpacket** passed as an argument. If you simply use **memcpy** for your **receive\_bitpacket** method, instead of pointer arithmetics, you will lose marks.

**Take Home**

Update the following functions to display the contents of the data packets using the built in **Print()** method.

* Winsock\_server
  + void send\_float\_array(float \*);
  + void receive\_float\_array(float \*);
  + void receive\_packet(packet &); *HINT: Thinking timing*
  + void receive\_bitpacket(bitpacket &);
* winsock\_client
  + void send\_float\_array(float \*);
  + void receive\_float\_array(float \*);
  + void send\_packet(packet);
  + void send\_bitpacket(bitpacket);

# SUBMISSION REQUIREMENTS

Once you have completed your lab create and upload the following files:

* Create a single ZIP file that contains all your source code files (\*.h and \*.cpp)
* The output.txt files generated by the lab
* Any additional information you feel necessary for me to mark your lab

1. Remember that arrays are, by default, passed by reference in C++. Hence, if you modify an array passed as an argument to a function, the modification will continue to affect the array after the function finishes execution. [↑](#footnote-ref-1)