BTN415 Lab UDP & Objects

UDP/IP and Objects using Dynamic Memory, Bits and CR

In this lab, you will apply your knowledge of UDP/IP Communication, Objects with Dynamic and Bitwise memory operations and CRC calculations.

LEARNING OUTCOMES

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

- Create a client & server application using UDP/IP Communications
- Successfully transmitted and received objects via a Socket
- Successfully calculate and transmit a Cyclic Redundancy Check algorithm

IN CLASS

In this lab we will complete the capabilities of the packet object, and finish the transmission of the packet over a UDP/IP socket connection

Download/Clone the Visual Studio solution from the class Github and perform the following:

PART A – Complete the Packet Class

```
Packet(char* src)
```

Complete the overloaded Packet constructor. A raw buffer of data just received over the socket interface is provided. Using the definition in packet.h, parse out the header, body and CRC information and populate the objects internal state.

```
void SetData(char* srcData, int Size)
```

Complete the SetData function in packet.h. This function receives two arguments. The first is a pointer to the address where the data to be transmitted is stored. The second is the size (in bytes) to be copied into the packet objects data buffer. Implement the allocation and copy of the data from the srcData location to the internal data location. Remember to update and set the header information accordingly.

```
char* SerializeData(int &TotalSize)
```

As we have learned in class, anytime an object has dynamic memory involved, it is important to serialize the data before transmitting it. SerializeData is to allocate enough space in the local TxBuffer of the Packet class, copy of the information/data to be transmitted into the buffer and return the TxBuffer address for the calling function to use in the transmission. This function also provides a reference to an integer, which you need to set to the total number of bytes in the TxBuffer.

```
unsigned int CalculateCRC()
```

Set the CRC data field to a value of 0xFF00FF00;

PART B – Complete the Server/Client CPP files

Add the UDP/IP code to send and receive the data packets of the socket connections defined. Look for the "TODO:" comment.

Take Home

```
unsigned int CalculateCRC()
```

Using your solution from the Cyclic Redundancy Check lab as a starting point. Implement a 32-bit CRC using the data in the TxBuffer of the Packet class. The polynomial to be used is as follows:

$X^{32} + x^{20} + x^{8} + x$

The answer should be stored in both the CRC member variable as well as included in the TxBuffer for transmission.

NOTE: The CRC should not use the CRC bytes as part of the calculation.

SUBMISSION REQUIREMENTS

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

- Packet.h
- Client.cpp
- Server.cpp
- Screenshot showing part of the execution of the code