BTN415 Term Project, Winter 2017

Milestone #2

In this milestone you will create a **MySocket** class that defines and implements TCP and UDP socket communications. An object of type **MySocket** is responsible for configuration and communication over a single socket connection. It can be instantiated to operate as a Server or Client connection. Your class implementation can be tested using the ***milestone2_Client.cpp*** and ***milestone2_Server.cpp*** files provided.

Class MySocket Requirements

In the global namespace, you should have the following defined:

- Enumeration of type **SocketType** that contains *{CLIENT, SERVER}*
- Enumeration of type **ConnectionType** that contains *{TCP, UDP}*
- *Constant integer* that defines the **DEFAULT_SIZE** of the buffer space

Your class **MySocket** should contain, as a minimum, the following member variables:

- **char *Buffer** to dynamically allocate RAW buffer space for communication activities
- A **WelcomeSocket** used by a **MySocket** object configured as a TCP/IP Server
- A **ConnectionSocket** used for client/server communications (*both TCP and UDP*)
- **struct sockaddr_in SvrAddr** to store connection information
- **SocketType mySocket** to hold the type of socket the **MySocket** object is initialized to
- **std::string IPAddr** to hold the *IPv4* IP Address string
- **int Port** to hold the port number to be used
- **ConnectionType connectionType** to define the *Transport Layer* protocol being used (*TCP/UDP*)
- A **bool bTCPConnect** flag to determine if a connection has been established or not
- **int MaxSize** to store the maximum number of bytes the buffer is allocated to. This will help prevent overflows and synchronization issues.

Your class **MySocket** should contain, as a minimum, the following member functions:

- **MySocket(SocketType, std::string, unsigned int, ConnectionType, unsigned int)** – A constructor that configures the socket and connection types, sets the IP Address and Port Number and dynamically allocates memory for the Buffer. Note that the constructor should put servers in conditions to either accept connections (if TCP), or to receive messages (if UDP).
    - NOTE: If an invalid size is provided the **DEFAULT_SIZE** should be used.
- **~MySocket()** – A destructor that cleans up all dynamically allocated memory space
- **void ConnectTCP()** – Used to establish a TCP/IP socket connection (3-way handshake).

    - *HINT: This function should have logic to prevent a **UDP** configured **MySocket** object from trying to initiate a connection-oriented scenario.*
- **void DisconnectTCP()** – Used to disconnect an established TCP/IP socket connection (4-way handshake)
- **void SendData(const char*, int)** – Used to transmit a block of RAW data, specified by the starting memory address and number of bytes, over the socket. This function should work with both TCP and UDP.
- **int GetData(char*)** – Used to receive the last block of RAW data stored in the internal **MySocket Buffer**. After getting the received message into Buffer, this function will transfer its contents to the provided memory address and return the total number of bytes written. This function should work with both TCP and UDP.
- **std::string GetIPAddr()** – Returns the IP address configured within the MySocket object
- **void SetIPAddr(std::string)** – Changes the default IP address within the MySocket object
    - **This method should return an error message if a connection has already been established**.
- **void SetPort(int)** – Changes the default Port number within the MySocket object
- **This method should return an error if a connection has already been established**
- **int GetPort()** – Returns the Port number configured within the MySocket object
- **SocketType GetType()** – Returns the default SocketType the MySocket object is configured as
- **void SetType(SocketType)** – Changes the default SocketType within the MySocket object

HINT: Set functionality should contain logic to prevent the header information from being changed if a TCP/IP connection is established or Welcome socket is open

You can download the milestone2_Client.cpp and milestone2_Server.cpp files from the course blackboard (or instructor's website).

Milestone2_Client.cpp

```
#include <iostream>
#include "MySocket.h"

int main()
{
	MySocket ClientSocket(SocketType::CLIENT, "127.0.0.1", 5000, ConnectionType::TCP, 100);

	std::string Pkt = "I love BTN415";

	ClientSocket.ConnectTCP();
	ClientSocket.SendData(Pkt.c_str(), strlen(Pkt.c_str()));

	char buff[100];
	int RxSize = ClientSocket.GetData(buff);

	std::cout << "Msg = " << buff << ", Bytes = " << RxSize << std::endl;

	ClientSocket.DisconnectTCP();

	return 1;
}
```

Milestone2_Server.cpp

```
#include <iostream>
#include "MySocket.h"

int main()
{
	MySocket ServerSocket(SocketType::SERVER, "127.0.0.1", 5000, ConnectionType::TCP, 100);

	char buff[100];

	int RxSize = ServerSocket.GetData(buff);

	std::cout << "Msg = " << buff << ", Bytes = " << RxSize << std::endl;

	std::string Pkt = "I Love BTN415 too!";

	ServerSocket.SendData(Pkt.c_str(), strlen(Pkt.c_str()));

	ServerSocket.DisconnectTCP();

	return 1;
}
```