Homework #4 - Socket Communication

# Goals of this assignment

1. Learn, through the use of sockets, a practical application for Inter-Process Communication
2. Go outside of your coding comfort zone!
3. Learn GUI development

# Description

For this assignment you have a choice. You may create a command line application for full credit, or a Graphical User Interface (GUI) using Microsoft Foundation Classes (MFC) for extra credit. Either way, your application will act as a simple Local Area Network (LAN) chat application. It will allow a user to enter chat messages on the command line or into an edit box and send them to another process on the same computer (simulating a chat application over the Internet). For the GUI version, transmitted chat messages will appear in a larger, read-only, edit box (along with received messages, optionally).

# Requirements and Specification

1. Messages shall be transmitted using a User Datagram Protocol/Internet Protocol (UDP/IP) Socket. Use the following link for explanation and examples of how to use socket functions in a Windows environment:

<https://msdn.microsoft.com/en-us/library/windows/desktop/ms741394(v=vs.85).aspx>

The socket functions you will need to use are socket(), sendto() and closesocket(). You will only be required to use a single socket for sending (you may **optionally** use a receive socket as well, in order to receive chat messages). Use IP address 127.0.0.1 and port 3514 (CS351-Section 4) to address the other process. **If you decide** to use a receive socket, you must use a different port - 3515. You will use the function recvfrom(), which is a blocking call (i.e. it waits for I/O). This function must be performed in a separate Thread.

1. When the user enters a chat message in the message box and presses return (or a Send button in a GUI), the message shall be sent to the addressee. If you are using a GUI, the message shall appear in the “Received Message” window. The format of a Chat Message shall be as follows:

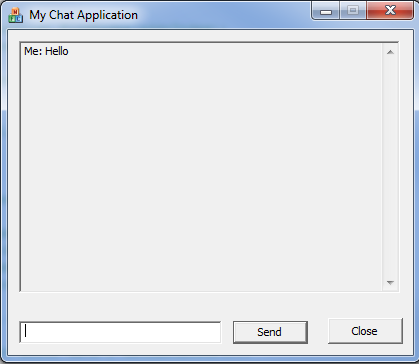
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | ….. | Byte n |
| A1 | A2 | A3 | A4 | A5 | A6 | ….. | An |

A1 - A(n-1): Alphanumeric characters representing the chat message entered by the user.

An : NULL character - ‘\0’

**If you decide** to use a receive socket, upon receiving a Chat Message, your application shall write the received message to the console, or add the message to the “Received Message” window if a GUI is used.

1. I will provide a companion application that will allow you to test yours. It will show Chat Messages in the Receive Message window. Also it will allow you to send Chat Messages to your application if you decide to use a receive socket. This is what my application window looks like. Feel free to model yours after mine, or come up with your own design.



# Scoring

If you complete the console version of the assignment (a console application that allows a user to send chat messages to 127.0.0.1:3514), you can earn up to 100 points for a homework assignment. This is a typical score for any other homework.

If you make your application a GUI, you will earn an additional 20 points to be added to the homework/quiz grade.

If you successfully implement a receive socket and display received messages (either on the console or GUI) you will earn an additional 10 points to be added to the midterm grade.