**Chat Application Using Socket Programming**

**14BCE024,14BCE029**

**Abstract:**

One of the most basic forms of human communication is chatting. Chat applications have been in existence almost since the inception of Internet itself. For any two programs running on a network, a communication link needs to be established between them. A socket is one such endpoint which enables contact via the link.

Perl language belongs to the family of general purpose, interpreted , dynamic programming languages. It includes Perl 5 and Perl 6. In the project itself, Perl has been used for two purposes 1) To create the Server and Client programs. 2) To design the GUI . Basic socket programming includes the server creating the link and listening for any clients wanting to establish the connection. the three phases of development involved 1) Create sockets on Client and Server end and allow the server to send a message across the link 2) Upgrade the one way communication from Server to Client to two way communication including Client and Server 3) Development of basic GUI for the purposes of demonstration and hiding the implementation details.

**Flow of the programs:**

1.Server

Start the Server program

Set the port to be communicated on

Set the protocol to communicate with

Set the server as local machine

Bind that socket to the port

If requests to connect, accept them and start the GUI

To receive any message click 'Receive' button

To send the message enter into the textbox and click the 'Send' button

2.Client

Start the Client program

Set the port to be communicated on

Set the protocol to communicate with

Set the server as local machine

Bind that socket to the port

Communication interface

To receive any message click 'Receive' button

To send the message enter into the textbox and click the 'Send' button

**Libraries used:**

1. strict

To enforce the standardized conventions used for the variable names , procedure calls used in the program. If unconventional names are used then the program does not run and issues an error.

2. Socket

To create the socket and used methods such as bind(), listen(), receive(), special print, gethostbyaddr(), autoflush etc that is all the methods required to establish and maintain the connection and communicate.

3. Win32::GUI

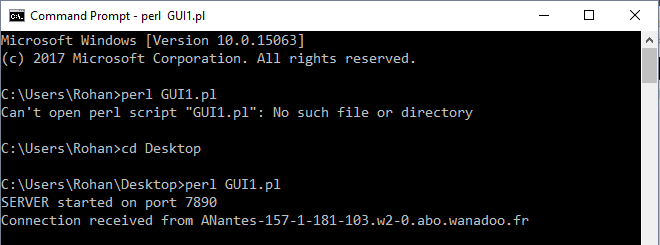
To create an easy to use Graphical Interface for the user to interact with instead of the bare bones command line utility to send and receive the messages amongst the Client and Server.

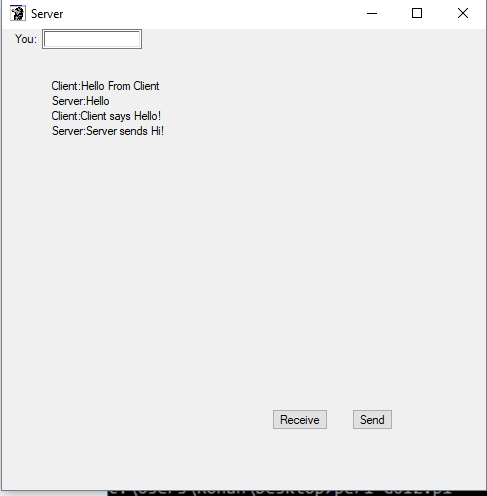
4. feature 'state'

To use the 'autoflush' command to clear out the socket handle before establishing the connection

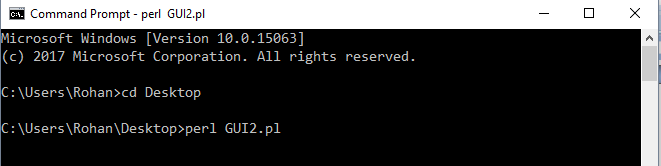
**The Programs :**

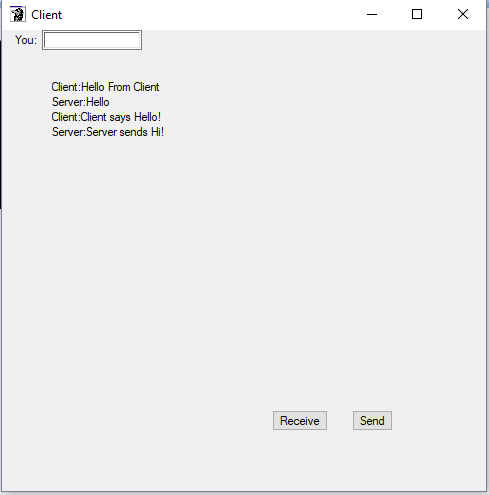
Server:





Client:





**Limitations:**

1. The Perl GUI is tough to implement and uneasy on the eyes to look upon and even packages like Tkinter provide just basic implementations.

2. The programs implemented are able to send and receive only one line at a time.

3. Separate button for 'SEND' had to be included because there is no event for textbox which can handle the pressing of ENTER key

4. A separate 'RECEIVE' button had to be integrated as automatic receiving would display 2 messages a time , one sent by the user and the previous message which was received in earlier cycle.

**Conclusion:**

Hence we have used the concepts of Socket Programming and GUI implementation. These have been used successfully to implement Chatting program which demonstrates the understanding of the concepts. Also using 'strict' enables us to use the accepted coding conventions. Win32::GUI although being unattractive still augments the Perl language as many other languages do not even have GUI implementation. The different packages available help us to implement and add various different functionalities.