Peer to Peer Chatting System
Final Report

CPE - 555
Real Time and Embedded Systems
Stevens Institute of Technology

Prof. Hong Man

Prerakkumar Doshi (CWID 10411489)
Namita Mall (CWID 10417064)
Parth Patel (CWID 10411938)
Shivani Laad (CWID 10418701)

"I pledge my honor that I have abided by the Stevens Honor System."

Project Description

This project aims at implementing an efficient communication between two or more hosts. Where in there one client is on a server host.There should be several other clients from different hosts so that can effectively communicate with each other using this Peer-to-Peer chatting system application. The IP address of all the client hosts involved can also be viewed. So this project is based on multithreading concept using C++ programming language. There are a variety of such systems available in the market, our project aims at solving the existing complexities in such systems and thereby solely focuses on enhancing the quality of communication between systems within a network.

Objective

The objective of this project is to successfully setup a peer to peer chatting system application, that is capable of facilitating effective encrypted and safe communications between the two or more hosts involved in the network.

Software and approach

The whole project based on object oriented C++ using its libraries like regular expression, STL and Multi-threading and concept like Exception Processing, Dynamic Memory allocation, inheritance, polymorphism const and mutable qualifiers, class orientation . All source files and Header files have been executed and performed on linux operating system and have been compiled using g++ compiler. We used Multi-thread concept so we don’t need socket the bind for all individual time, whenever the client is bind the thread socket invoked automatically. So need to lock and unlock the thread (mutex function)

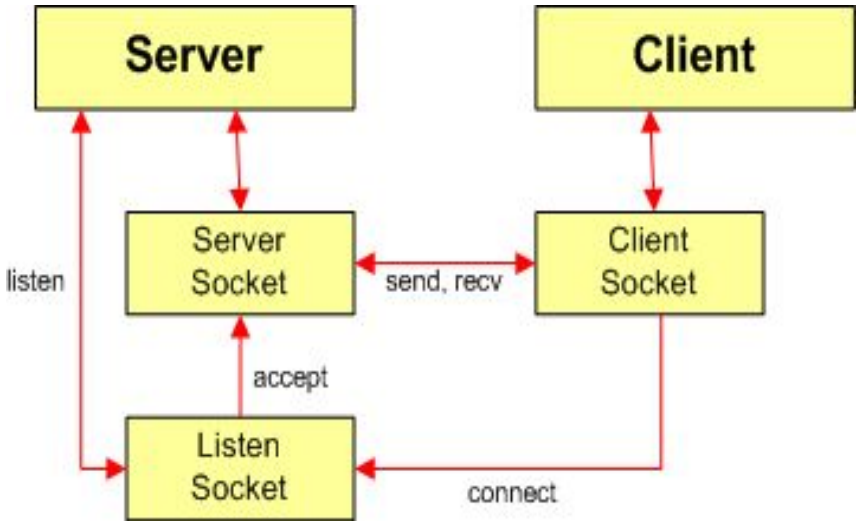
Limitations:

As it is Linux OS based application, every client hosts will execute only on terminal or MAC OS. And every client hosts should have to compile client header as well as source file using g++ compiler with multi-thread concept. Also all client hosts should be on the same Server.

Results and outcome

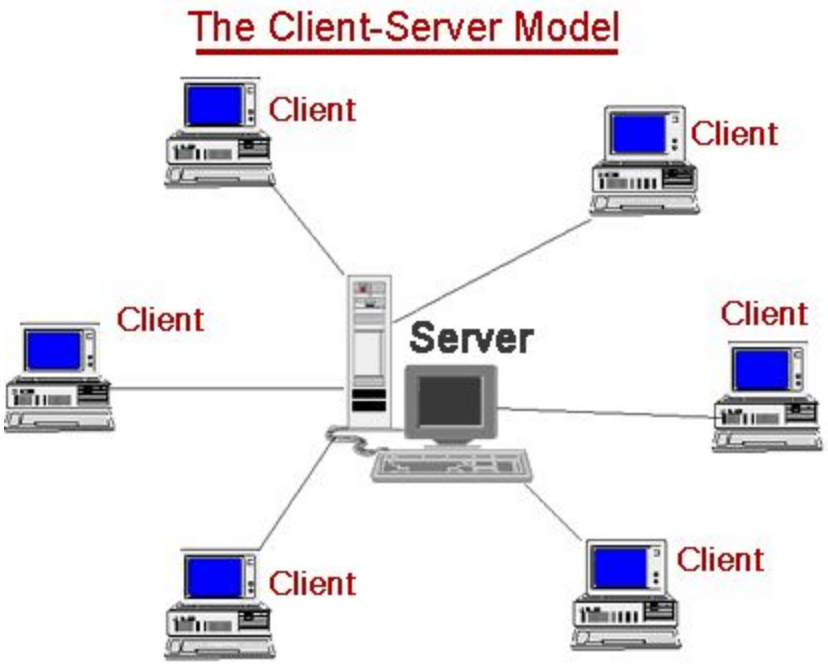
We have come up with successful Peer-to-Peer chatting system Application, where we could do safe chat with other clients by establishing a server between several clients. Also we could show the ip address of each client hosts system involved in this application on server host. As we chat on terminal as stdin or through file, this system is completely encrypted.

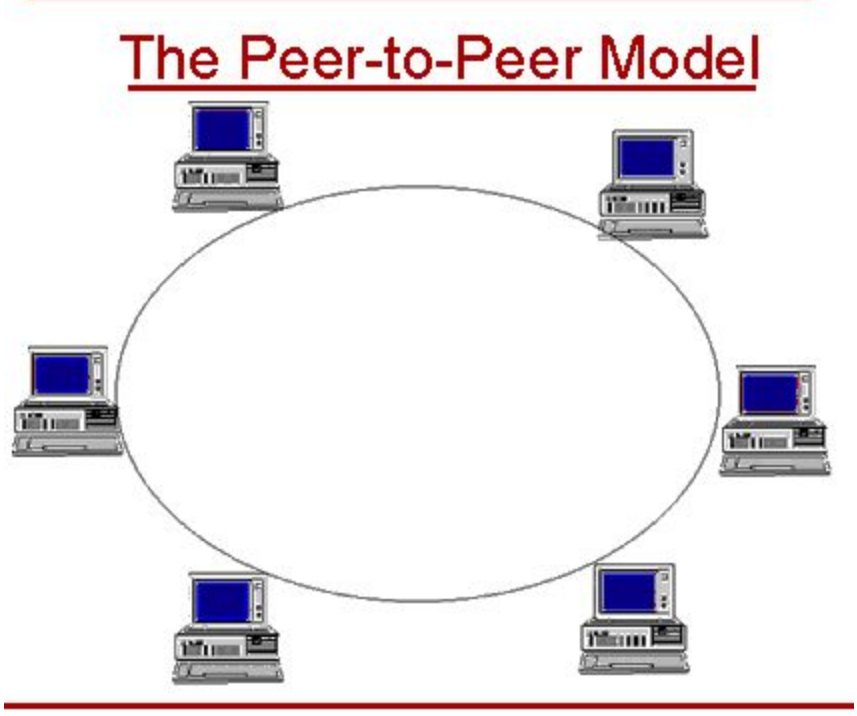
Client- server Architecture



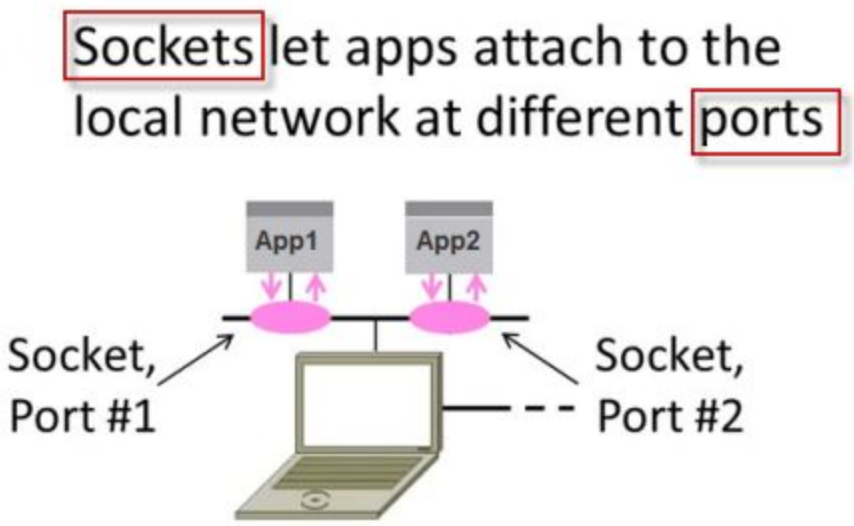
Sockets are a mechanism for exchanging data between processes. These processes can either be on the same machine, or on different machines connected via a network. Once a socket connection is established, data can be sent in both directions until one of the endpoints closes the connection.
We needed to use sockets for a project we were working on, so we developed and refined a few C++ classes to encapsulate the raw socket API calls. Generally, the application requesting the data is called the client, and the application servicing the request is called the server. we created two primary classes, Client and Server, that the client and server could use to exchange data.

WorkSpace


The Client-Server Model
Client
Client
Client
Server
Client
Client
Client


The Peer-to-Peer Model

Socket



Server Socket

1. create a **socket**
2. **bind** to an address (6666 which is client port)
3. **listen** on a port, and wait for a connection to be established.
4. **accept** the connection from a client.
5. **send/recv** - the same way we read and write for a file.
6. **shutdown** to end read/write.
7. **close** to releases data.

Client Socket

1. create a **socket**.
2. **bind*** - (Unnecessary).
3. **connect** to a server.
4. **send/recv** - repeat until we have or receive data
5. **shutdown** to end read/write.
6. **close** to releases data.

Program

We define three port here for binding of Socket

#define SERVERPORT "8888" // the port users will be connecting to
#define CLIENTPORT "6666" // port on which client listens for other peers' connections
#define PEERPORT "7777" //port which a client uses to connect with other clients' listening port.

For IP address of online clients we have built below member function, which prints IP Address of online clients

```
bool getOnlineClients(int sockfd)
{
	int rv;
	if(send(sockfd, "LIST", 5, 0) <0)
	{
		fprintf(stderr,"Error in sending LIST to server\n");
	}

	char buf[MAXDATASIZE];
	memset(buf, '\0', MAXDATASIZE);
	if((rv = recv(sockfd, buf, MAXDATASIZE-1, 0))>0)
	{
		cout<<"Online clients :-\n";
		cout<<string(buf);
	}
	else
	{
		cout<<"Server Down! Application will exit after current chat is over...\n";
		close(sockfd);
		return false;
	}
	return true;

}
```

This below *for* loop bind the Client to Server using Socket

```
for(p = clientInfo; p != NULL; p = p->ai_next)
        {
                if ((clientSocket = socket(p->ai_family, p->ai_socktype,
                        p->ai_protocol)) == -1)
                {
                        perror("client: socket");
                        continue;
                }

                if (setsockopt(clientSocket, SOL_SOCKET, SO_REUSEADDR, &yes,
                                sizeof(int)) == -1) {
                        perror("setsockopt");
                        exit(1);
                }

                if (bind(clientSocket, p->ai_addr, p->ai_addrlen) == -1) {
                        close(clientSocket);
                        perror("server: bind");
                        continue;
                }

                break;
        }
```

Steps to compile the Program

###To compile and run server###

g++ -std=c++11 server.cpp server.h -lpthread -o server

###To run server###

./server

###To compile client###

g++ -std=c++11 -fpermissive -w client.cpp client.h -lpthread -o client

###To run client###

./client 155.246.163.104 stdin

or ***./client 155.246.163.104 file***

Important note: the argument to ./client is the IP address of the server for all client hosts

Also -lpthread is for MultiThread concept compilation

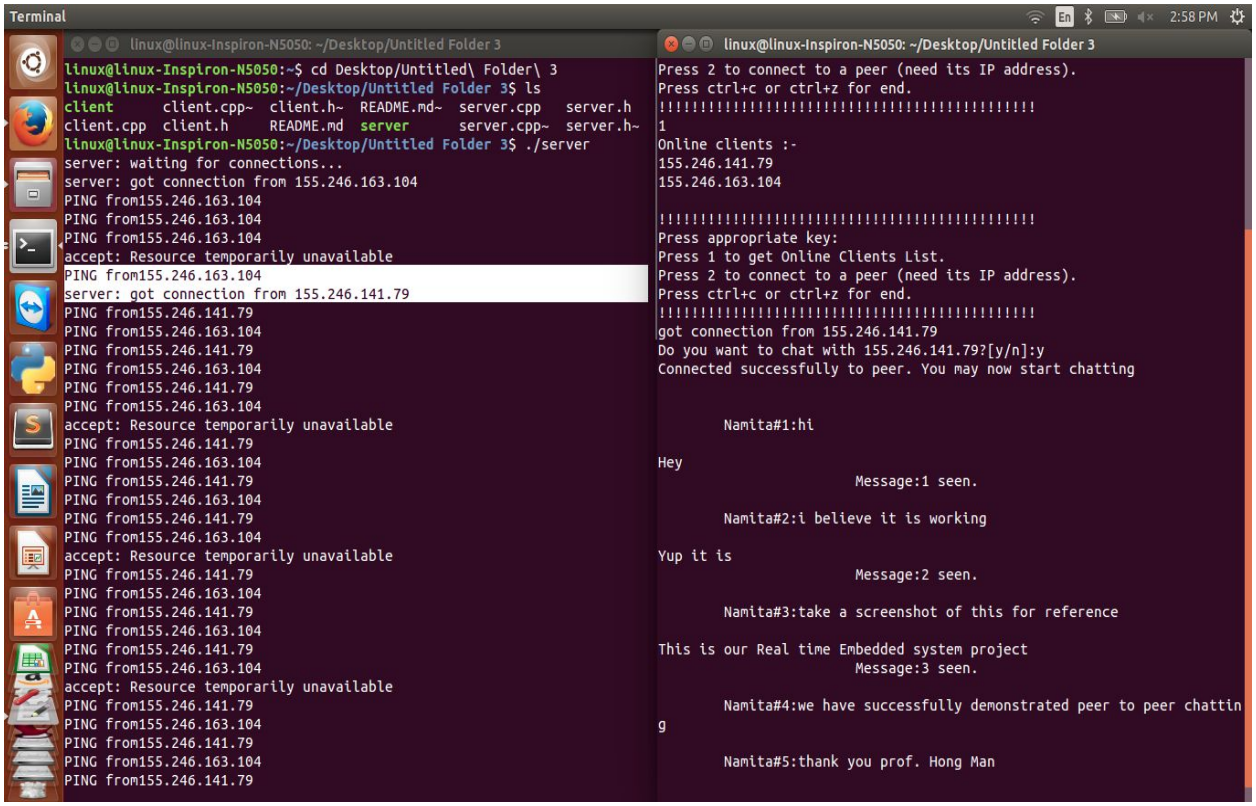
Contribution

Most of the time we all were worked mutually, as we all wanted to stick with applications Design. But client side has been designed by Prerak and Namita, and Server side is designed by Shivani and Parth.

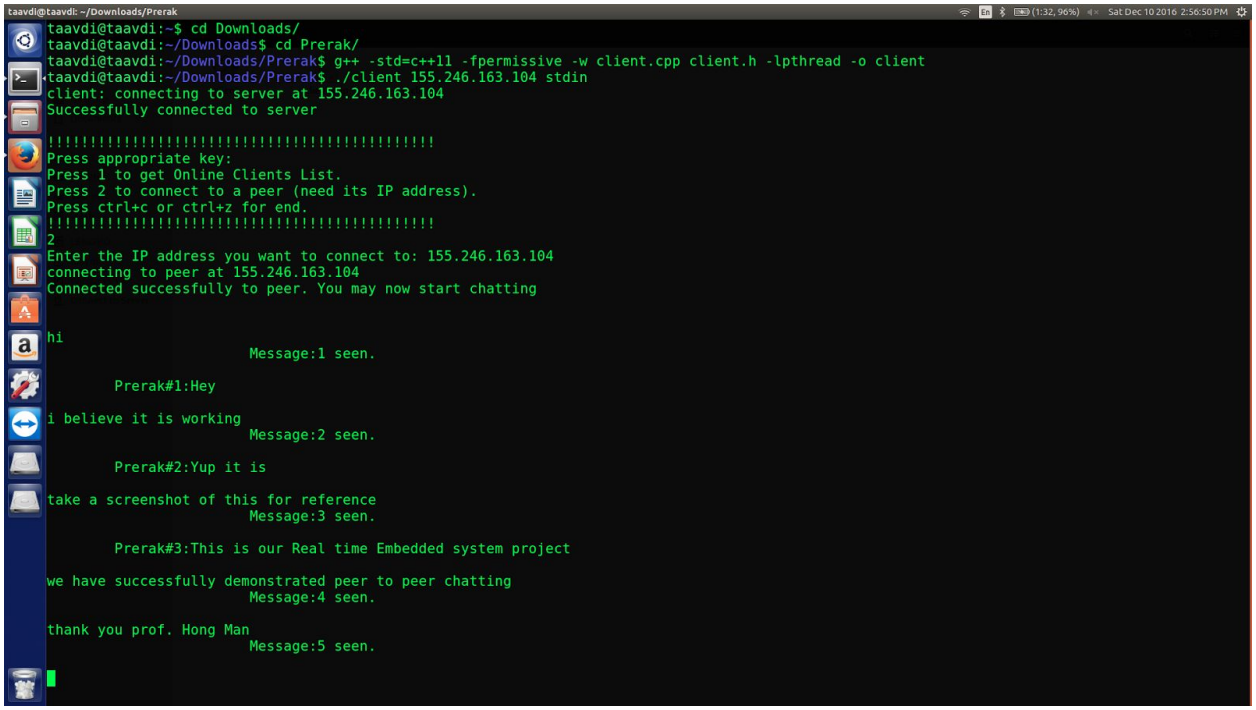
ScreenShots

Server Host

Client 1



Client 2 from different Host (Machine)



Conclusion

Sockets are a simple and efficient way to send data between processes. In this application we've gone over socket communications, and developed this Peer to Peer chatting system. One should now be able to run this socket communication application!