# 实验二 Socket通信实验

1. **题意**：

设计程序，分别构建通信的两端:服务器端和客户端应用程序,套接字类型为面向连接的Socket,自己构建双方的应答模式，实现双方的数据的发送和接收（S发给C，C发给S）。

1. **基本思路**：
   1. 服务器端：服务器要与多个客户端连接，为每一个连接的客户端创建一个通信Socket，自己还要有一个Socket用于监听客户端的连接请求；服务器要将连接进来的客户端显示出来，用户可以根据显示出来的用户列表来向指定的客户端发信息；接收所有的客户端的信息，并将信息转发给指定的客户端。
   2. 客户端：客户端要有接收服务器信息的功能，但客户端只向服务器发信息，客户端通过服务器的转发功能向其它的客户端发送信息。客户端要可以处理服务器发过来的信息，还要有数据结构用来保存所有客户端的名字，并将所有客户端名字列表显示出来。
2. **核心代码**：
   1. 连接服务器：

private void button1\_Click(object sender, EventArgs e)

{

button1.Enabled = false;

try

{

client = new TcpClient(Dns.GetHostName(), 56781);

toolStripStatusLabel1.Text = "连接成功";

}

catch

{

toolStripStatusLabel1.Text = "连接失败，请再次登录！";

button1.Enabled = true;

return;

}

NetworkStream networkStream = client.GetStream();

br = new BinaryReader(networkStream);

bw = new BinaryWriter(networkStream);

SendMessToServer("Login," + userName.Text);

Thread threadRece = new Thread(new ThreadStart(ReceMessFromServer));

threadRece.IsBackground = true;

threadRece.Start();

}

* 1. 接受服务器消息

private void ReceMessFromServer()

{

string receMess = null;

while(isExit == false)

{

try

{

receMess = br.ReadString();

}

catch

{

if (isExit == false)

MessageBox.Show("服务器断开");

break;

}

string[] splitString = receMess.Split(',');

string command = splitString[0].ToLower();

switch (command)

{

case "login": //先不做

break;

case "logout": //先不做

break;

case "talk": //格式：talk,用户名,对话信息

//AddTalkMessage(splitString[1] + "：\r\n");

//AddTalkMessage(receiveString.Substring(

// splitString[0].Length + splitString[1].Length+2));

AddTalkMessage(string.Format("[{0}]说：{1}",

splitString[1], receMess.Substring(

splitString[0].Length + splitString[1].Length + 2)));

break;

}

}

Application.Exit();

}

* 1. 向服务器发送信息

private void SendMessToServer(string message)

{

try

{

//将字符串写入网络流，此方法会自动附加字符串长度前缀

bw.Write(message);

bw.Flush();

}

catch

{

AddTalkMessage("发送失败!");

}

}

* 1. 接受客户端连接

private void ListeingClient()

{

TcpClient newClient = null;

while (true)

{

try

{

newClient = tcpListener.AcceptTcpClient();

}

catch

{

break;

}

User user = new User(newClient);

Thread newClientThread = new Thread(ReceMessage);

newClientThread.Start(user);

userlist.Add(user);

AddMessageToMonitor(string.Format("[{0}]进入。", newClient.Client.RemoteEndPoint));

AddMessageToMonitor(string.Format("当前连接数：{0}", userlist.Count));

}

}

* 1. 处理Client发来的消息

private void ReceMessage(object userData)

{

User user = (User)userData;

TcpClient client = user.client;

while(isExit == false)

{

string receMess = null;

try

{

receMess = user.br.ReadString();

}

catch

{

if(isExit == false)

{

AddMessageToMonitor(String.Format("与[{0}]失去联系，已终止接收该用户信息", client.Client.RemoteEndPoint));

RemoteUser(user);

}

break;

}

AddMessageToMonitor(string.Format("{0}:{1}", user.client.Client.RemoteEndPoint, receMess));

string[] splitString = receMess.Split(',');

switch (splitString[0])

{

case "Login":

user.userName = splitString[1];

Adduser(splitString[1]);

break;

case "Logout":

RemoteUser(user);

Remoteuserlist(splitString[1]);

return;

case "Talk":

string talkMess = receMess.Substring(splitString[0].Length + 1);

AddMessageToMonitor(string.Format("[{0}] said: {1}"

, user.userName

, talkMess));

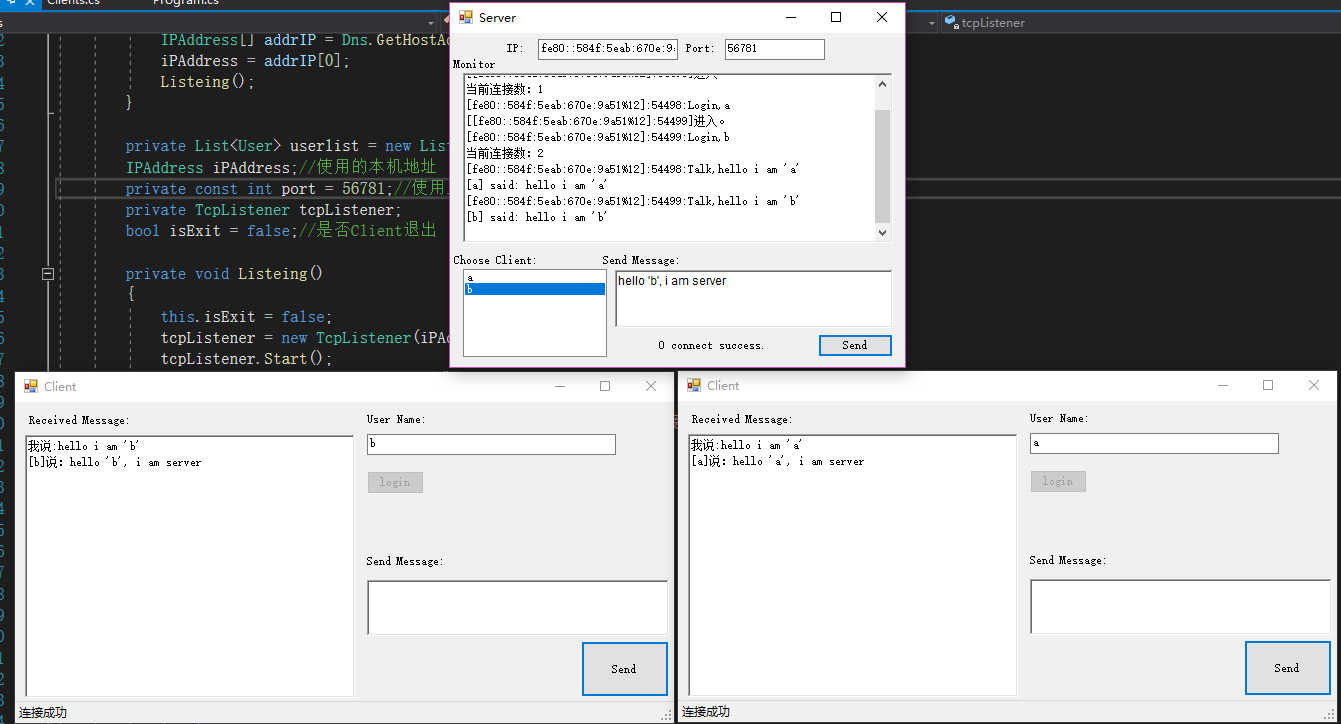
break;

}

}

}

1. **运行截图**：



1. **实验总结**：收获很大。