实 验 报 告

组号：

姓名：　　学号：　　班级：

姓名：　　学号：　　班级：

一、 实验名称

Socket 网络编程实验，实现一个简单的聊天程序

二、 实验原理

使用 C#进行编写，采用了 Client/Server 模式的工作原理。

1. 验证用户登录时，客户端发送用户输入的账户和密码，服务器端对账户的密码进行比较，若密码正确则服务器允许该客户进行登录。
2. 在发送文字消息时，客户端向服务器端发送包含用户消息的报文，由服务器端将报文转到目标用户。
3. 发送离线文件时，客户端发送的文件暂存在服务器上，当接收方顺利接收文件后服务器端删除该文件。
4. 发送在线文件时，采用 NAT 穿透技术将发送端与接收端直接连接，发送完成后将 NAT 穿透连接拆除。
5. 进行语音通话时，双方的客户端先进行 NAT 穿透，完成 NAT 穿透后双方直接进行通信。

三、 实验目的

1. 通过 socket 编程，进一步理解计算机网络的工作原理，掌握基本的 socket 程序的编写方法，尝试设计和实现自己的应用层协议。
2. 掌握多线程编程的基本方法，理解多线程程序的工作原理和调试方法。

四、 实验内容

1. 基本功能

1) 验证用户登录

用户通过之前注册过的账户以及密码进行登录操作，若密码错误服务器不允许该用户登录。

2) 用户注册

用户输入密码后，服务器将会返回该用户的唯一账户号码。每个用户的账户号码由 10 位的数字组成。

3) 用户之间的文字聊天

用户之间可以进行文字聊天，且实时性较好，可以在发送文件或语音通话的同时进行文字聊天，实现了用户的离线文字消息的存储，用户可以收到未登录时其他用户发送的消息。

2. 高级功能

1) 用户之间传输离线文件

用户之间可以传输 100M 以上的大文件，实现了客户端的文件上传与下载功能。在用户发送或接收离线文件时依然可以使用消息发送功能。在发送过程和接收过程中，用户都可以取消发送或暂停发送。

2) 用户之间通过 NAT 穿透传输在线文件

通过 NAT 穿透技术，用户之间传输文件可以不经过服务器暂存，直接将文件发送到对方的客户端。在用户发送或接收在线文件时依然可以使用消息发送功能。在发送过程和接收过程中，双方都可以取消发送或暂停发送。

3) 断点续传功能

用户在发送离线文件的过程中，取消文件发送后，服务器会暂存未发送完全的文件，当客户端再次发送文件时服务器可以继续接收该文件。

4) 用户之间的语音聊天

用户之间进行质量较好的语音聊天，聊天语音清晰，延迟较小，聊天效果接近当前市面上的聊天软件效果。在语音通话的同时，用户依然可以正常使用消息传输功能。

5) 用户网络异常后的处理

客户端和服务器端都有很好的鲁棒性，在网络发生各种异常时，客户

端和服务器端都会及时做出相应，不会出现无响应等异常状态。用户可以得到良好的使用体验。

五、 实验实现

1. 人员分工

1) XXX：客户端结构的设计与编写，消息收发系统的设计与编写，应用层协议的设计与实现，文件传输协议的设计，NAT 穿透结构设计，在线文件传输部分设计与实现，语音通话部分设计与实现，实验报告客户端部分、协议部分和消息收发系统部分的编写。

2) XXX：编写服务器部分的程序，服务器转发消息机制的设计与实现，参与程序的调试，参与编写报告。服务器环境部署。

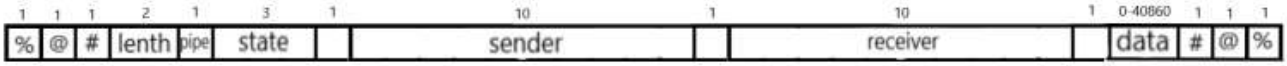
2. 实验设计

（一）协议

协议全部采用 TCP 传输协议进行实现

1）报文格式

协议报文格式如下。



Lenth：长度为 2 字节，报文除去报头的“%@#”部分和 lenth 部分后剩余部分的长度。

Pipe：长度为 1 字节，该报文使用的管道（管道：参考了多路复用技术的思想，将 TCP 报文流划分为了 3 个管道，各个线程之间可以通过不同的管道进行无干扰的消息收发）

State：长度为 3 字节，该报文的状态码，不同的状态码有不同的使用场景。（状态码表附在实验报告附件中）

Sender 和 receiver：长度为 10 字节，发送方和接收方的 10 位的数字账号。

Data：报文中的数据，数据长度可变，从 0-40860 字节。

判断报文边界：

在接收消息时，接收方通过报头的“%@#”来识别报头，然后接收两个字节的 lenth，再从 TCP 流上接收 lenth 个字节后，接收消息过程完成

2）登录与系统消息传输协议

协议步骤：

a） 客户端发送“100”报文请求登录服务器，报文数据部分包含了用户输入的密码。

b） 若密码正确，服务器端返回“400”报文，允许客户端登录，若密码错误，服务器端返回“401”号报文，拒绝用户登录。

c） 用户收到“400”报文后，发送“101”“103”“104”号报文，请求服务器发送当前已注册用户个数，当前账户的离线文件目录，当前账户的离线文字消息。

d） 服务器返回“402”“406”“410”报文，分别返回当前已注册用户个数，当前账户的离线文件目录和当前账户的离线文字消息。若当前账户无离线文件或无离线文字消息，则不回复“406”报文或“410”报文。

3）文字消息传输协议

协议步骤:

a） 客户端发送“200”报文，报文中 data 部分携带发送方的文字消息

b） 服务器收到“200”报文后，判断接收方是否在线，若在线则立刻发送“410”报文给接收方客户端，若不在线，则先将该离线消息暂存于服务器。

c） 接收方客户端收到“410”报文后，将消息显示于客户端上

4）离线文件传输协议

协议步骤：

发送离线文件部分：

a） 客户端向服务器发送“301”报文，报文中包括客户端将要发送的文件名和文件大小。

b） 服务器端向客户端返回“421”报文，报文中包括客户端已经发送的文件的长度。

c） 客户端根据服务器“421”报文中的长度偏移发送文件的文件指针，并读出 40860 字节的内容放入“302”报文中发送给服务器。重复该步骤直到

文件全部发送。若在该过程中，客户端取消发送则发送一个“306”报文，表示客户端取消发送文件。

d) 客户端发送“303”报文，表示所有文件包均发送。

e) 服务器端返回“427”报文，表示所有文件数据包均收到。

接收离线文件部分：

a) 客户端向服务器发送“304”报文，报文中包含客户端请求的文件名。

b) 服务器向客户端返回“423”报文，表示该文件在服务器端存在。

c) 客户端向服务器发送“305”报文，报文中包含客户端当前已经接收的文件大小。

d) 服务器端将客户端请求的文件的文件指针按照客户端以接收的文件大小进行偏移，从文件中读出 40860 字节放入“424”报文中，发送给客户端。重复该过程直到文件全部发送。在该过程中，若收到了客户端发送的“316”报文，则表示客户端取消接收文件，服务器停止发送文件包。

e) 服务器发送“425”报文，表示该文件已经全部发送完毕。

f) 客户端向服务器返回“315”报文，表示客户端已经接收到了全部文件。

5) NAT 穿透协议

NAT 穿透有多种方式，在比较了各种 NAT 穿透方式的优劣后，我们选择了成功率高的服务器代理方式。

协议步骤：

假设客户端 A 请求与客户端 B 进行 NAT 穿透

a) 客户端向服务器发送“300”报文，请求与客户端 B 进行 NAT 穿透。

b) 若客户端 B 不在线，服务器向客户端 A 返回“310”报文，表示对方无法进行 NAT 穿透。若对方在线，则将“300”报文发送给客户端 B。

c) 客户端 B 收到“300”报文后，若同意进行 NAT 通信则返回“307”报文，若不允许进行 NAT 通信则返回“310”报文。

d) 服务器收到客户端 B 发送的“307”报文后，将在 50000-60000 端口之间选择一个端口用于连接两个客户端，并通过“420”报文将该端口号返回给客户端 A 和 B，并且开始在该端口上监听 TCP 连接。若服务器收到的客户端 B 的“310”报文，则直接将该报文转发给客户端 A。

e) 客户端 A 与 B 收到了服务器的“420”报文后，提取出报文中端口号，并且尝试与服务器的该端口进行连接。

f) 两个客户端连接上该端口后，服务器将两个 TCP 流连接，从而实现了两个客户端的 NAT 穿透。

6）在线文件传输协议

假设客户端 A 请求向客户端 B 发送在线文件

协议步骤：

a) 首先通过服务器进行客户端 A 到客户端 B 的 NAT 穿透。

b) 建立 NAT 连接后，客户端 A 发送“500”报文，包含即将发送的文件名和文件大小。

c) 客户端 B 返回“501”报文，表示可以开始接收文件。

d) 客户端 A 开始将文件放入“502”报文中，发送给客户端 B。在此过程中，若收到客户端 B 发送的“512”报文后，表示需要暂停发送，客户端 A 将会暂停发送，直到收到客户端 B 的“513”报文。若收到了客户端 B 的“504”报文，表示取消文件发送。若此时，客户端 A 取消了发送，客户端 A 将发送“504”报文，表示取消文件发送。发送完毕后，客户端 A 发送“503”报文，表示所有文件已经发送完毕。

e) 客户端 B 收到全部报文后，返回“515”包，表示所有文件都已经收到。

7）语音通话传输协议

假设客户端 A 向客户端 B 发起语音通话。

协议步骤：

a) 首先通过服务器进行客户端 A 到客户端 B 的 NAT 穿透。

b) 客户端 A 向客户端 B 发送“500”报文，询问客户端 B 是否做好语音通话准备。

c) 客户端 B 返回“501”报文，表示可以开始进行语音通话。

d) 客户端 A 与 B 通过“502”包进行语音数据传输。

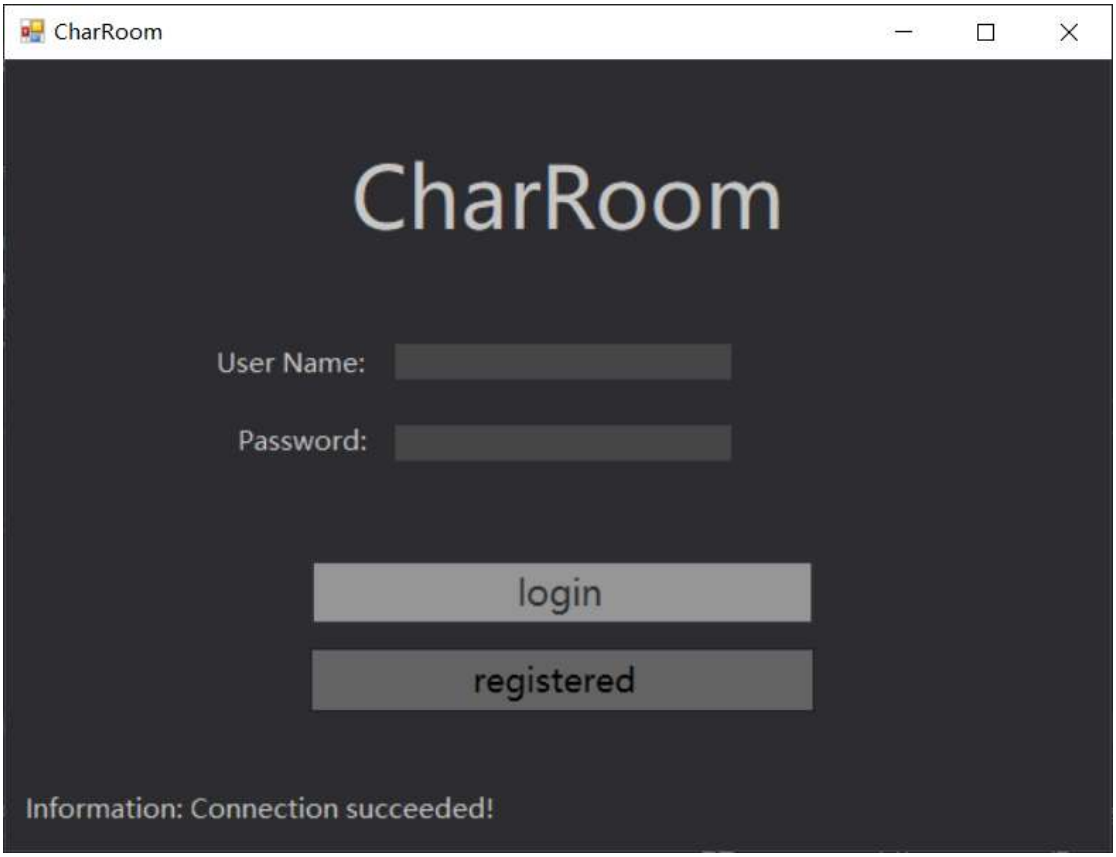
e) 当客户端 A 或客户端 B 任意一方需要结束通话时，发送“504”包。

8）心跳协议

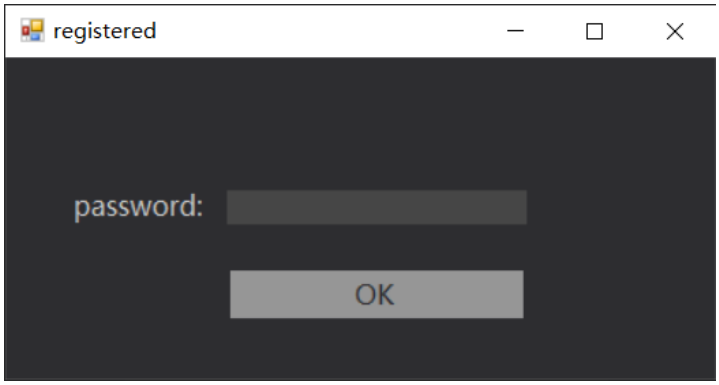
为了使得客户端和服务器能够及时的察觉到 TCP 连接的异常，在协议中加入了心跳功能。客户端每 3 秒给服务器发送一个“999”报文，服务器若在 10 秒内没有收到客户端的任何报文则判断 TCP 连接终止。

（二）UI 设计

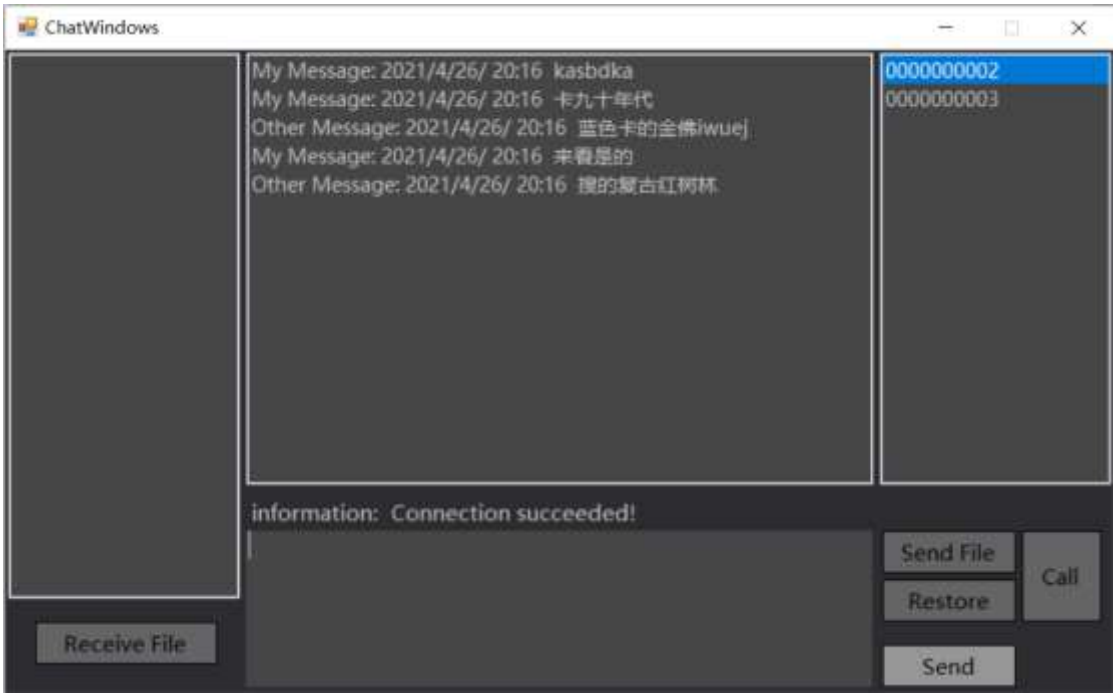
登录界面：



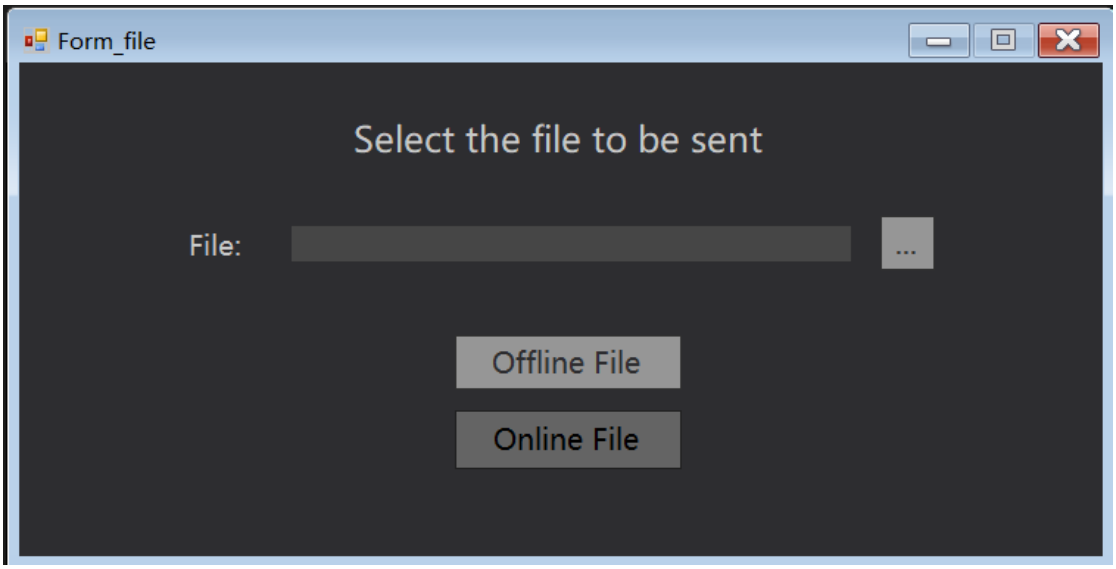
注册账户：



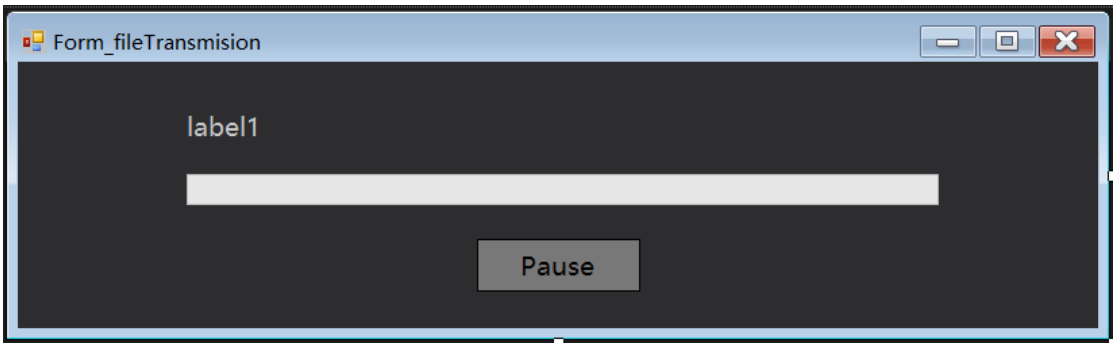
用户界面：



发送文件选择窗口：



文件发送：



（三）框架结构

1. 应用层报文收发系统（Postbox 类）

a) Massage 类

功能：负责消息的编码和译码，所有发送到 TCP 流上的信息都必须通过 message 类的实例进行发送。

函数：

Coding（）：数据编码函数，将 message 的实例转换为比特数组

Encoding（Byte[] newData）：数据译码函数，将收到的比特数组转换为 message 的实例

b) Postbox 类

功能：负责将 message 的实例发送到 TCP 流中

结构：该类中主要有以下几种结构

i. 自带发送线程 sendThread 和 receiveThread，用于在 TCP 流中接收和发送 message 实例

ii. 接收队列 receivequeue，用于暂存接收到的 message 实例

iii. 发送队列 sendqueue，用于暂存需要发送的 message 实例

iv. 连接状态 counnctState，表示当前的 TCP 连接状态，0 代表未连接，1 代表连接正常。

v. TCP 流 Stream，用于向 TCP 流中写入和取出数据。

工作方式：

i. 其他的线程有需要发送的数据时，首先需要新建一个 message 的实例，通过 postbox 类中的 sendmessage 函数将该消息放入发送队列中，同时唤醒 sendThread 线程，该线程会从队列中取出 message 的实例，将该实例进行编码后发送到 TCP 流中。

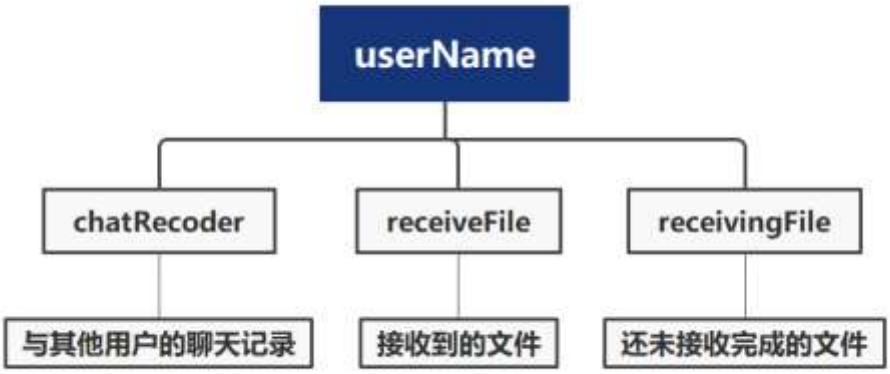
ii. receivThread 线程接收 message 实例的过程参考了自动机理论的方法。当 receivThread 线程从 TCP 流中接收到了比特流后，首先识别该流中的消息头部，找到头部后将协议中的长度域取出，按照长度域中的长度取出整个协议报文，再通过 mesaage 函数的 Encoding 函数将该报文转换成 message 的实例。然后 receiveThread 线程将该

message 实例放入接收队列中后，唤醒在相应管道上等待的线程。

iii. 为了能够让多个线程能够同时在一条 TCP 流中进行接收和发送操作，在 postbox 中加入了管道的概念，每个线程都可以使用单独的管道进行发送。每一个管道有自己的发送队列和接收队列，同时给每个管道加上了优先级，优先级高的管道总时优先进行发送。

2. 客户端框架结构

客户端文件结构：



为了保证用户的相应度，客户端采用了多线程技术，主要的线程及功能如下：

a) 窗体线程

产生客户端界面，响应用户产生的事件。该线程不执行任何的耗时指令，只负责接收用户的指令并将用户的指令交付给其他后台线程进行执行。

b) 消息接收后台线程

负责接收全部的服务器指令，需要及时响应服务器的指令，并将运行的结果反馈给窗体线程进行显示，有需要时建立其他的线程以实现程序的功能。该线程一直在 0 号管道上进行监听，接收全部的 0 号管道的消息。

c) 离线文件传输线程

离线文件传输线程负责离线文件的收发工作，该线程与消息接收后台线程共享 postbox 的实例，采用 1 号管道进行消息的收发工作，由于文件的收发耗时很长，所以该线程总是处于阻滞的状态，但不会影响到其他线程的工作，客户端依然可以在离线文件传输的过程中保持对用户的高响应度。该线程仅在有文件传输任务时被建立，文件传输结束后，该线

程结束。

d) 在线文件传输线程

在线文件传输线程首先会通过 NAT 穿透建立自己的 postbox 实例，在进行通信时不会影响到其他线程的工作。

e) 语音通话线程

语音通话线程需要在发送本地客户端的录音数据的同时接收来自对方客户端的录音数据，所以语音通话线程有自己独立的 postbox 进行收发工作，由于语音通话的过程中，通信双方需要同时进行大流量的交互，所以该线程中自带两个子线程，接收线程负责接收对方的声音信号然后播放，发送线程负责录制当前客户端声音然后发送给对方。由于语音通话线程有自己的窗体程序，所以在进行语音通话时不会对正常的消息传输产生影响，客户端依然可以保证对用户的高响应度。

客户端功能实现方式：

a) 消息接收功能

消息接收后台线程在后台监听 0 号管道，一旦接收到了服务器发送的“410”消息，消息接收后台线程会更新本地的聊天记录，然后更新当前显示的聊天记录，达到实时向用户显示消息的效果。该线程除了需要监听“410”消息，还需要在收到服务器的 NAT 穿透请求时快速响应，及时的向用户反馈服务器的消息。

b) 离线文件传输功能

用户点击了发送或接收文件按钮后，离线文件传输线程被建立，然后按照离线文件传输协议的过程进行文件传输，文件传输完成后关闭该线程。

c) 在线文件传输功能（采用了 NAT 穿透技术）

发送文件时，在线文件传输线程被建立，然后按照离线文件传输协议的过程进行文件传输，文件传输完成后关闭该线程。

接收文件时，当后台线程接收到了在线文件传输请求后，先显示用户窗口，让用户选择是否接收该文件，若用户选择接收文件后，后台线程建立在线文件传输线程，执行在线文件传输协议后，在线文件传输线程

关闭。

d) 语音通话功能（采用了 NAT 穿透技术）

在比较了多个 C#音频库后，我们选择 NAudio 库进行麦克风声音的采集以及录音播放。仔细阅读了 NAudio 的 readme 文档以及该库的源码后，我们理解了库中函数的使用方法。

客户端请求通话时，语音通话线程被建立，然后按照语音通话协议的过程进行语音通话。

当后台线程接收到了其他客户端语音通话的请求后，先显示用户窗口，让用户选择是否接听，若用户选择接听后，后台线程建立语音通话线程，执行语音通话协议后，语音通话线程关闭。

e) 心跳功能

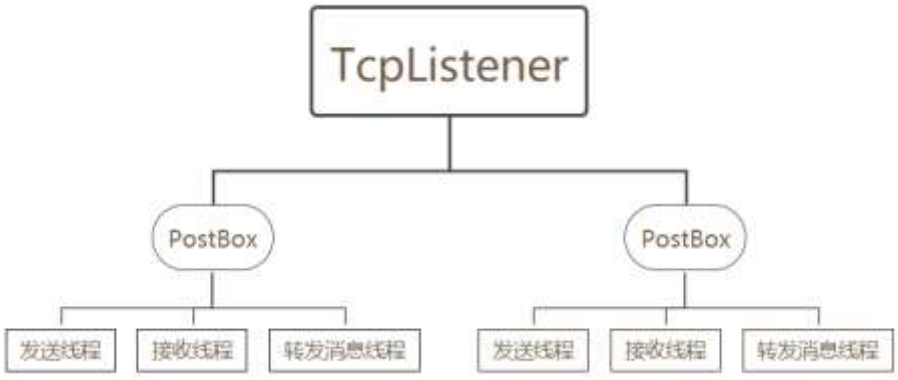
在客户端中加入了计时器，每隔固定的时间后会触发一次心跳消息发送。

f) 恢复功能

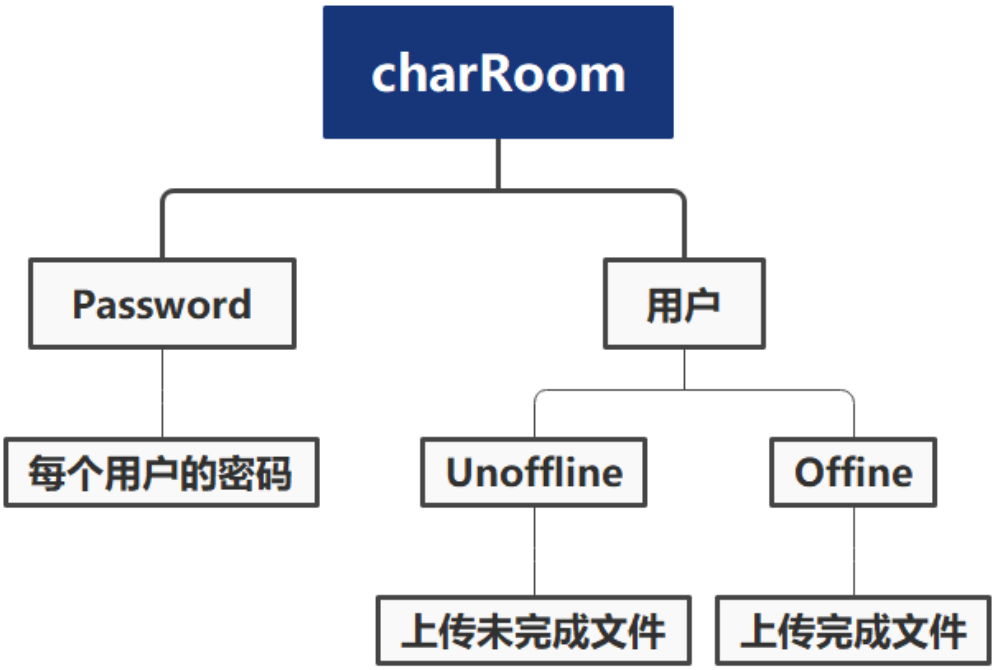
当由于网络连接中断或其他异常时，用户可以点击 restore 键进行网络恢复。恢复函数会重新尝试连接服务器，并更新当前的客户端状态。

3. 服务器框架结构

服务器线程说明：



服务器文件结构



服务器通过多线程来实现多个用户。一个用户对应一个线程，线程中包含该用户的账号,线程为该用户唯一创建一个用于接收、发送消息的 PostBox。主线程使用 TcpListener 来侦听来自 TCP 网络客户端的连接，当有用户打开客户端时，TcpListener 监听到连接请求后，为该用户开启线程并创建 PostBox。PostBox 收到客户端发送的消息后，通过报文状态码来判断该报文的类型，并进入相应的 if 语句执行相应代码。

用户打开客户端后，客户端每隔 3s 会向服务器发送心跳报文，服务器若 5s 内未收到任何报文，则判定客户端意外关闭，释放 Postbox，从 List 中移除该用户对应的项(若存在)，关闭 TCP 连接，并关闭该用户对应的所有线程。

主线程中声明了一个 int 型的全局变量 Count，用于记录当前服务器已注册的用户个数，并使用 Mutex 为其加锁，防止两个线程同时对该变量进行操作导致错误。Count 的值从相对路径下的 number.txt 文件中取出，Count 被修改后也将相应写入文件中，防止服务器因意外关闭导致已有账号信息丢失。当有用户需要注册时，该用户对应的线程通过 Count 计算并生成分配给该用户的账号，并将账号返回给客户端，同时对 Count 的值进行修改。

主线程中使用 List<T>维护了 SendSignal 类的实例，该 List 通过 Autoresetevent 进行加锁。SendSignal 类的声明如下：

```
public int No;  //该值与已登录的用户账户唯一对应
public AutoResetEvent signal;   //该信号用于唤醒用户对应的转发消息线程
public bool close;  //该信号指示用户对应的转发消息线程是否需要关闭
```

用户申请登陆时，判断输入密码与用户注册密码是否一致。一致时，则允许登录，向 List 中增加 No 为该用户账号的 SendSignal 的实例，并创建属于该用户的负责转发消息的线程。

用户退出登录时，服务器释放 PostBox，将 List 中对应的实例移除，关闭 TCP 连接，并关闭该用户对应的所有线程。

当用户 A 向用户 B 发送消息时，将消息信息保存在本地，并检索 List，若 List 中包含 B 对应的实例，则表示用户 B 在线，此时用户 A 通过设置 B 对应的 signal 信号来唤醒 B 的转发消息线程，使用户 B 的客户端能够及时显示消息。

当用户 A 向用户 B 发送离线文件时，服务器新建线程，该新线程将文件接收并存入本地。收到文件发送完成的确认包后，服务器向客户端回复确认包，后关闭线程。用户 B 登陆时检索本地是否有离线文件并显示文件名，同时也可通过点击 Restore 按钮获取最新的离线文件列表。

当用户需要下载离线文件时，服务器新建线程，该新线程将文件分解转换为报文进行发送，收到用户下载完成的确认报文后，线程关闭。

当用户 A 向用户 B 请求 NAT 穿透时，服务器建立线程为用户 A 与用户 B 进行连通，使其可以直接进行沟通交互。

4. 关键代码的描述

本次实验的代码均为小组成员自主编写

（一）关键代码 1：服务器端 TcpListener。

```
server = new TcpListener(IPAddress.Any, ServerNum.port);
server.Start(); //开始监听
while (true)
{
        Console.Write("Waiting for a connection... \n");
        TcpClient client = server.AcceptTcpClient();       //监听到 TCP 连接
        Server serv = new Server();
        Thread BreProgram = new Thread(new ParameterizedThreadStart(serv.Communicate));
```

```
//为用户开启新线程
            BreProgram.Start(client);
}
```

自主编写：XXX

（二）关键代码 2：服务器端为登录用户开启转发消息线程。

```
if (!MyTcpListener.sendsignal.Exists(x => x.No == account))
//该用户无转发消息线程
    {
        MyTcpListener.sendsignal.Add(new SendSignal(account));   //创建消息线程
        Thread newthread = new Thread(new ThreadStart(Ifmessage));
        newthread.Start();
    }
```

自主编写：XXX

（三）关键代码 3：服务器端有消息需要转发时激活转发线程

```
if (MyTcpListener.sendsignal.Exists(x => x.No == rec))
    {
        //保存 message 信息
        string path = Path.Combine(MyTcpListener.dataPath, receiveMessage.receiver,
"message.txt");
        string text3 = receiveMessage.state + " " + receiveMessage.sender + " " +
receiveMessage.receiver;
        File.AppendAllText(path, text3);
        MyTcpListener.sendsignal.Find(x => x.No == rec).signal.Set();
//唤醒接收用户转发线程
}
```

自主编写：XXX

（四）关键代码 4：客户端发送文字消息

```
private void Button_SendMessage_Click(object sender, EventArgs e)          //发送按钮
        {
            if(postbox.connectState!=1)
            {
                return;
            }
            if (listBox_friend.SelectedItem == null)              //还未选择发送的用户
            {
                label_inform.Text = "Please select a user.";
                return;
            }
            if (textBox_inputMessage.Text == "")              //还未输入发送内容
            {
                label_inform.Text = "Please input message.";
                return;
```

```
            }
            //向服务器发送消息
            string userdata = textBox_inputMessage.Text.Replace("\r\n", "");
            string data_string = DateTime.Now.Year.ToString() + "/" +
DateTime.Now.Month.ToString() + "/" + DateTime.Now.Day.ToString() + "/ "
                + DateTime.Now.Hour.ToString() + ":" + DateTime.Now.Minute.ToString() +
"  " + userdata;
            Byte[] data_byte = System.Text.Encoding.UTF8.GetBytes(data_string);
            Message message_Send = new Message(0,"200", userName,
(string)listBox_friend.SelectedItem, data_byte);
            try
            {
                postbox.SendMessage(message_Send); //发送数据包
            }
            catch
            {
                label_inform.Text = "Failed to send. ";
                return;
            }

            event_fileOpen.WaitOne();
            string path = Path.Combine(userName, "ChatRecoder", message_Send.receiver +
".txt");
            StreamWriter streamWriter = new StreamWriter(path, true);
//将消息记录入聊天文件
            streamWriter.WriteLine("0" + data_string);
            streamWriter.Close();
            //1：别人发送的，0：自己发送的
            event_fileOpen.Set();
            PrintChat(); ;
            textBox_inputMessage.Text = "";
        }
```

自主编写：XXX

（五）关键代码 5：客户端发送文件

```
private void button_file_Click(object sender, EventArgs e)              //文件按钮
        {
            if(postbox.connectState!=1)             //邮箱是否可用
            {
                return;
            }
            if (file_sending != null || file_receiveing != null || calling)
//是否有其他文件在发送
            {
```

```
                label_inform.Text = "Can't send now ";
                return;
            }
            if (name_nowChoose==null)            //是否选择了接收方
            {
                label_inform.Text = "please choose receiver. ";
                return;
            }
            Form_file form_File = new Form_file();          //显示文件选择窗口
            form_File.ShowDialog();

            if(form_File.path_file==null)             //用户没有选择文件
            {
                return;
            }
            file_sending = form_File.path_file;
            form_fileSending = new Form_fileTransmision();
            form_fileSending.SetLable(Path.GetFileName(form_File.path_file), true,
name_nowChoose);
            form_fileSending.SetProgressBar(0);
            form_fileSending.Show();                                    //开启进度
条窗口
            if (form_File.flag_OfflineFile)                         //发送离线文件
            {
                Thread thread_SendThread = new Thread(new ThreadStart(OfflineSendFile))
                {
                    IsBackground = true
                };          //启动发送线程
                thread_SendThread.Start();
            }
            else
            {
                Thread thread_SendThread = new Thread(new ThreadStart(NetFileSend))
                {
                    IsBackground = true
                };          //启动发送线程
                thread_SendThread.Start();
            }
        }
```

自主编写：XXX

（六）关键代码 6：客户端接收离线文件

```
private void button_receivefile_Click(object sender, EventArgs e)              //接收
按钮
        {
```

```
            if (postbox.connectState != 1)              //判断是否正常连接
                return;
            if (file_receiveing != null||file_sending != null||calling)            //判
断是否已经有文件正在接收
            {
                label_inform.Text = "Can't receive now ";
                return;
            }
            if (file_nowChoose == null)             //判断是否未选择文件
            {
                label_inform.Text = "please choose file. ";
                return;
            }
            file_receiveing = (string)listBox_file.SelectedItem;           //更新正在接
收的文件
            /*if (File.Exists(file_receiveing))               //已经有文件了就需要删除
原来的文件
                File.Delete(file_receiveing);*/
            form_fileReceive = new Form_fileTransmision();
            form_fileReceive.SetLable(file_receiveing, false, name_nowChoose);
            form_fileReceive.SetProgressBar(0);
            form_fileReceive.Show();
            Thread thread_SendThread = new Thread(new ThreadStart(OfflineReceiveFile))
            {
                IsBackground = true
            };                                                  //启动接收线程
            thread_SendThread.Start();
        }
```

自主编写：XXX

（七）关键代码 7：客户端恢复连接函数

```
void Initialization()                               //类的初始化（建立连接，绘制界面）
        {
            string folderPath = userName;
            timer_heart.Enabled = true;
            Message message_send;
            try
            {
                if (postbox.connectState != 1)
                    postbox.CreateConnect(ServerNum.IPAddress, ServerNum.port);
                postbox.Restore();
                message_send = new Message(0, "999", userName, ServerNum.name, null);
//心跳包发送
                postbox.SendMessage(message_send);
                if (Directory.Exists(folderPath) == false)
```

```
                Directory.CreateDirectory(folderPath);//创建文件夹
            folderPath = Path.Combine(userName, "ChatRecoder");
            if (Directory.Exists(folderPath) == false)
                Directory.CreateDirectory(folderPath);
            folderPath = Path.Combine(userName, "ReceiveFile");
            if (Directory.Exists(folderPath) == false)
                Directory.CreateDirectory(folderPath);
            folderPath = Path.Combine(userName, "ReceivingFile");
            if (Directory.Exists(folderPath) == false)
                Directory.CreateDirectory(folderPath);
            if (backgroundThread == null || !backgroundThread.IsAlive)
            {
                backgroundThread = new Thread(new ThreadStart(BackgroundThread))
//建立后台线程
                {
                    IsBackground = true
                };
                backgroundThread.Start();
            }
            //listBox_file.BeginUpdate();
            listBox_file.Items.Clear();
            message_send = new Message(0, "101", userName, ServerNum.name, null);
//请求好友列表
            postbox.SendMessage(message_send);
            message_send = new Message(0, "103", userName, ServerNum.name, null);
//传输离线文件个数
            postbox.SendMessage(message_send);
            message_send = new Message(0, "104", userName, ServerNum.name, null);
//传输离线文字消息
            postbox.SendMessage(message_send);
            label_inform.Text = "Connection succeeded!";
        }
        catch (Exception e)
        {
            label_inform.Text = e.Message + " at Initialization";       //无法与服
务器建立连接
            return;
        }
    }
```

自主编写：XXX

（八）关键代码 8：客户端消息接收后台线程

```
void BackgroundThread()                                    //后台线程
    {
        while (postbox.connectState == 1)
```

```
        {
            Message message_receive;
            Message message_send;
            try
            {
                postbox.ReceiveMessage(0, out message_receive);
            }
            catch (Exception e)
            {
                label_inform.Text = e.Message+" at BackgroundThread";       //无法
与服务器建立连接
                return;
            }
            if (message_receive.state == "410")                //接收到服务器发送的
普通消息
            {
                event_fileOpen.WaitOne();
                string path = Path.Combine(userName, "ChatRecoder",
message_receive.sender + ".txt");
                StreamWriter streamWriter = new StreamWriter(path, true);
                streamWriter.WriteLine("1" +
Encoding.UTF8.GetString(message_receive.data));
                //1：别人发送的，0：自己发送的
                streamWriter.Close();
                event_fileOpen.Set();
                if (message_receive.sender == name_nowChoose)
                    PrintChat();
            }
            else if(message_receive.state == "402")
            {
                int userSum =
Convert.ToInt32(System.Text.Encoding.UTF8.GetString(message_receive.data));
                listBox_friend.BeginUpdate();                       //显示好友列表
                listBox_friend.Items.Clear();
                for (int x = 1; x <= userSum; x++)
                {
                    if (x == Convert.ToInt32(userName)) continue;
                    listBox_friend.Items.Add(x.ToString("D10"));
                }
                listBox_friend.EndUpdate();
            }
            else if (message_receive.state == "406")
            {
                listBox_file.BeginUpdate();                         //更新文件列表
```

```
                string name = Encoding.UTF8.GetString(message_receive.data);
                int findResult = listBox_file.FindStringExact(name);
                if (findResult < 0)
                    listBox_file.Items.Add(name);
                listBox_file.EndUpdate();
            }
            else if (message_receive.state == "300")
            {
                if (file_sending != null||file_receiveing != null||calling)
//是否有其他文件在发送
                {                       //判断是否已经有文件正在接收
                    message_send = new Message(0, "310", userName,
message_receive.sender, null);
                    postbox.SendMessage(message_send);
                    return;
                }
                bool natIsFile = Encoding.UTF8.GetString(message_receive.data) ==
"file";       //判断是否时NAT文件发送
                if (natIsFile)
                    form_Netfile = new Form_NetFile(true);
                else
                {
                    if (form_Calling != null && form_Calling.calling)
                    {
                        message_send = new Message(0, "310", userName,
message_receive.sender, null);
                        postbox.SendMessage(message_send);
                        return;
                    }
                    else
                        form_Netfile = new Form_NetFile(false);
                }
                form_Netfile.ShowDialog();
                if(form_Netfile.chooseYes)
                {
                    if(natIsFile)
                    {
                        form_fileReceive = new Form_fileTransmision();
//开启进度条

form_fileReceive.SetLable(Path.GetFileName(file_receiveing), false, name_nowChoose);
                        form_fileReceive.SetProgressBar(0);
                        MethodInvoker methodInvoker_file = new
MethodInvoker(ShowForm_fileReceive);
```

```
                        BeginInvoke(methodInvoker_file);
                        Thread thread_NetFile = new Thread(new
ParameterizedThreadStart(NetFileReceive))
                        {
                            IsBackground = true
                        };
                        thread_NetFile.Start(message_receive.sender);
                    }
                    else
                    {
                        form_Calling = new Form_Calling(postbox);
                        form_Calling.SetForm(userName, message_receive.sender,
false); //开启通话窗口
                        MethodInvoker methodInvoker_call = new
MethodInvoker(ShowForm_Calling);
                        BeginInvoke(methodInvoker_call);
                    }
                }
                else
                {
                    message_send = new Message(0, "310", userName,
message_receive.sender, null);
                    postbox.SendMessage(message_send);
                }
                form_Netfile = null;
            }
        }
    }
```

自主编写：XXX

（九）关键代码 9：客户端文件接收线程

```
void OfflineSendFile()                    //发送离线文件
    {
        FileStream streamReader = new FileStream(file_sending, FileMode.Open);
        postbox.OpenPipe(1);                    //打开 1 号管道
        Message message_Receive;
        Message message_Send;
        long lenth_HaveReceive;
        Byte[] file_Byte;
        int lastpoint = 0;
        bool succeed = true;
        try
        {
            message_Send = new Message(0,
```

```
                    "301",
                    userName,
                    name_nowChoose,
                    Encoding.UTF8.GetBytes(Path.GetFileName(file_sending) + "*" +
Convert.ToString(streamReader.Length))); //请求与对方建立文件传输连接
                postbox.SendMessage(message_Send);
                postbox.ReceiveMessage(1,out message_Receive);
                if (message_Receive.state != "421")
                    throw new Exception("The server prohibits sending files .");
                lenth_HaveReceive =
Convert.ToInt64(Encoding.UTF8.GetString(message_Receive.data));
                streamReader.Seek(lenth_HaveReceive, SeekOrigin.Begin);
//设置文件指针
                file_Byte = new Byte[Message.lenth_DateByte];
                int lenth = 0;
                while ((lenth = streamReader.Read(file_Byte, 0, Message.lenth_DateByte)) !=
0)
                {
                    form_fileSending.waitHandle_pause.WaitOne();
                    if (form_fileSending.state == 3)
                    {
                        succeed = false;
                        break;
                    }
                    message_Send = new Message(1, "302", userName, name_nowChoose,
file_Byte.Skip(0).Take(lenth).ToArray());
                    postbox.SendMessage(message_Send);
                    if (lastpoint != 100 * streamReader.Position / streamReader.Length)
                    {
                        lastpoint = (int)(100 * streamReader.Position / streamReader.Length);
                        form_fileSending.SetProgressBar(lastpoint);
                    }
                    //Thread.Sleep(500);
                }
                if (succeed)
                    message_Send = new Message(1, "303", userName, name_nowChoose,
null);      //发送文件确认包
                else
                    message_Send = new Message(1, "306", userName, name_nowChoose,
null);      //发送文件取消包
                postbox.SendMessage(message_Send);
                if (message_Send.state == "303")
                {
                    do
```

```
                {
                    postbox.ReceiveMessage(1, out message_Receive);
                } while (message_Receive.state != "427");
                throw new Exception("Send file succeed");
            }
            else
                throw new Exception("Cancel sending ");
        }
        catch (Exception e)
        {
            label_inform.Text = e.Message;
            postbox.ClosePipe(1);
            streamReader.Close();
            file_sending = null;
            form_fileSending.Close();
            form_fileSending = null;
        }
    }
    void OfflineReceiveFile()                                   //接收离线文件
    {
        /*if (!File.Exists(file_receiveing))
            File.Create(file_receiveing);*/
        file_receiveing = Path.Combine(userName, "ReceivingFile", file_receiveing);
        FileStream fileStream = new FileStream(file_receiveing, FileMode.Append);
        Message message_Receive;
        Message message_Send;
        postbox.OpenPipe(1);                                    //打开 1 号管道
        long size_file;
        long size_HaveReceive = fileStream.Length;
        int lastpoint = 0;
        try
        {
            message_Send = new Message(0,
                "304",
                userName,
                ServerNum.name,
                Encoding.UTF8.GetBytes(file_nowChoose)); //请求与对方建立文件传
输连接
            postbox.SendMessage(message_Send);
            postbox.ReceiveMessage(1, out message_Receive);
            if (message_Receive.state != "423")
                throw new Exception("The server prohibits receiving files .");
            size_file = Convert.ToInt32(Encoding.UTF8.GetString(message_Receive.data));
            message_Send = new Message(1, "305", userName, ServerNum.name,
```

```
Encoding.UTF8.GetBytes(Convert.ToString(size_HaveReceive)));
                    postbox.SendMessage(message_Send);
                    while (true)
                    {
                        //Thread.Sleep(500);
                        //form_fileReceive.waitHandle_pause.WaitOne();
                        if (form_fileReceive.state == 3)
                        {
                            message_Send = new Message(1, "316", userName,
ServerNum.name, null); //请求取消发送
                            postbox.SendMessage(message_Send);
                            throw new Exception("cancel receive");
                        }
                        else if (form_fileReceive.state == 0)
                        {
                            message_Send = new Message(1, "313", userName,
ServerNum.name, null); //请求暂停发送
                            postbox.SendMessage(message_Send);
                            form_fileReceive.waitHandle_pause.WaitOne();
                            message_Send = new Message(1, "314", userName,
ServerNum.name, null); //请求继续发送
                            postbox.SendMessage(message_Send);
                        }
                        postbox.ReceiveMessage(1,out message_Receive);
                        if (message_Receive.state == "424")
                        {
                            fileStream.Write(message_Receive.data, 0,
message_Receive.data.Length);
                            if (lastpoint != 100 * fileStream.Position / size_file)
                            {
                                lastpoint = (int)(100 * fileStream.Position / size_file);
                                form_fileReceive.SetProgressBar(lastpoint);
                            }
                        }
                        if (message_Receive.state == "425")
                        {
                            message_Send = new Message(1, "315", userName,
ServerNum.name, null); //确认收到全部文件
                            postbox.SendMessage(message_Send);
                            int index =
listBox_file.FindStringExact(Path.GetFileName(file_receiveing));
                            listBox_file.Items.RemoveAt(index);
                            throw new Exception("Receive succeed");
                        }
```

```
                }
            }
            catch(Exception e)
            {
                label_inform.Text = e.Message;
                postbox.ClosePipe(1);                    //关闭 1 号管道
                fileStream.Close();
                if (e.Message == "Receive succeed")
                {
                    string newPath = Path.Combine(userName, "ReceiveFile",
Path.GetFileName(file_receiveing));
                    if (File.Exists(newPath))
                        File.Delete(newPath);
                    File.Move(file_receiveing, newPath);
                }
                file_receiveing = null;
                form_fileReceive.Close();
                form_fileReceive = null;
                //listBox_file.BeginUpdate();
            }
        }
        void NetFileSend()
        {
            Message message_Receive;
            Message message_Send;
            int net_Port;
            Postbox postbox_Net = new Postbox();
            FileStream fileStream = new FileStream(file_sending, FileMode.Open);
            Byte[] sendData = new Byte[Message.lenth_DateByte];
            bool succeed = true;
            int lastpoint = 0;
            postbox.OpenPipe(1);
            try
            {
                message_Send = new Message(0, "300", userName, name_nowChoose,
Encoding.UTF8.GetBytes("file")); //发送 300 包  请求 NET 穿透地址
                postbox.SendMessage(message_Send);
                while (!postbox.ReceiveMessage(1, out message_Receive, 1000))
                {
                    if (form_fileSending.state == 3)
                        throw new Exception("Cancel sending");
                }
                if(message_Receive.state=="310")
                    throw new Exception("Opposite refused the connection ");
```

```
                if (message_Receive.state != "420")
                    throw new Exception("server error ");
                net_Port = Convert.ToInt32(Encoding.UTF8.GetString(message_Receive.data));
                postbox_Net.CreateConnect(ServerNum.IPAddress, net_Port);
                message_Send = new Message(0, "500", userName, name_nowChoose,
                    Encoding.UTF8.GetBytes(Path.GetFileName(file_sending) + "*" +
Convert.ToString(fileStream.Length)));
                postbox_Net.SendMessage(message_Send);
                postbox_Net.ReceiveMessage(0, out message_Receive);
                if (message_Receive.state != "501")
                    throw new Exception("Opposite refused the connection ");
                int lenth = 0;
                while ((lenth = fileStream.Read(sendData, 0, sendData.Length)) != 0)
                {
                    form_fileSending.waitHandle_pause.WaitOne();
                    if (form_fileSending.state == 3)
                    {
                        succeed = false;
                        break;
                    }
                    if(postbox_Net.ReceiveMessage(0, out message_Receive, 0))
                    {
                        if (message_Receive.state == "512")
                        {
                            do
                            {
                                postbox_Net.ReceiveMessage(0, out message_Receive);
                            } while (message_Receive.state != "513" &&
message_Receive.state != "504");
                        }
                        if (message_Receive.state == "504")
                            throw new Exception("Opposite refused the connection ");
//接收方请求关闭文件传输
                    }
                    message_Send = new Message(0, "502", userName, name_nowChoose,
sendData.Skip(0).Take(lenth).ToArray());
                    postbox_Net.SendMessage(message_Send);
                    if (lastpoint != 100 * fileStream.Position / fileStream.Length)
                    {
                        lastpoint = (int)(100 * fileStream.Position / fileStream.Length);
                        form_fileSending.SetProgressBar(lastpoint);
                    }
                    //Thread.Sleep(500);
                }
```

```
                if (succeed)
                    message_Send = new Message(0, "503", userName, name_nowChoose,
null);        //发送文件确认包
                else
                    message_Send = new Message(0, "504", userName, name_nowChoose,
null);        //发送文件取消包
                postbox_Net.SendMessage(message_Send);
                if (message_Send.state == "503")
                {
                    do
                    {
                        postbox_Net.ReceiveMessage(0, out message_Receive);
                    } while (message_Receive.state != "515" && message_Receive.state !=
"504");
                    if (message_Receive.state == "504")
                        throw new Exception("Opposite refused the connection ");
//接收方请求关闭文件传输
                    else
                        throw new Exception("Net send succeed");
                }
                else
                    throw new Exception("Cancel sending ");
            }
            catch (Exception e)
            {
                label_inform.Text = e.Message;
                postbox.ClosePipe(1);
                postbox_Net.ClosePostbox();
                file_sending = null;
                fileStream.Close();
                form_fileSending.Close();
                form_fileSending = null;
            }
        }
        void NetFileReceive(object input_receiver)                    //NET 接收方
        {
            string receiver = (string)input_receiver;
            Message message_Send;
            Message message_Receive;
            Postbox postbox_Net = new Postbox();
            int size_file;
            int net_Port;
            try
            {
```

```
                postbox.OpenPipe(1);
                message_Send = new Message(0, "307", userName, receiver, null);
//发送确认接收报文
                postbox.SendMessage(message_Send);
                postbox.ReceiveMessage(1, out message_Receive);
                postbox.ClosePipe(1);
                if (message_Receive.state != "420")
                    throw new Exception("server error");
                net_Port = Convert.ToInt32(Encoding.UTF8.GetString(message_Receive.data));
                postbox_Net.CreateConnect(ServerNum.IPAddress, net_Port);
                if (!postbox_Net.ReceiveMessage(0, out message_Receive, 5000))          //
接收文件名和文件大小
                    throw new Exception("Opposite refused the connection ");
                string[] receiveData =
Encoding.UTF8.GetString(message_Receive.data).Split('*');
                string file_name = receiveData[0];
                size_file = Convert.ToInt32(receiveData[1]);
                file_receiveing = Path.Combine(userName, "ReceiveFile", file_name);
            }
            catch (Exception exception)
            {
                label_inform.Text = exception.Message;
                file_receiveing = null;
                postbox_Net.ClosePostbox();
                return;
            }
            if (File.Exists(file_receiveing))                    //删除已有文件
                File.Delete(file_receiveing);
            FileStream fileStream = new FileStream(file_receiveing, FileMode.OpenOrCreate);
            try
            {
                message_Send = new Message(0, "501", userName, receiver, null);
                postbox_Net.SendMessage(message_Send);
                int lastpoint = 0;
                while (true)
                {
                    //form_fileReceive.waitHandle_pause.WaitOne();
                    if (form_fileReceive.state == 3)
                    {
                        message_Send = new Message(0, "504", userName, receiver, null);
                        postbox_Net.SendMessage(message_Send);
                        throw new Exception("cancel receive");
                    }
                    else if (form_fileReceive.state == 0)
```

```
                {
                    message_Send = new Message(0, "512", userName,
ServerNum.name, null); //请求暂停发送
                    postbox_Net.SendMessage(message_Send);
                    form_fileReceive.waitHandle_pause.WaitOne();
                    message_Send = new Message(0, "513", userName,
ServerNum.name, null); //请求继续发送
                    postbox_Net.SendMessage(message_Send);
                }
                postbox_Net.ReceiveMessage(0, out message_Receive);
                if (message_Receive.state == "502")                    //接收文件包
                {
                    fileStream.Write(message_Receive.data,                    0,
message_Receive.data.Length);
                    if (lastpoint != 100 * fileStream.Position / size_file)
                    {
                        lastpoint = (int)(100 * fileStream.Position / size_file);
                        form_fileReceive.SetProgressBar(lastpoint);
                    }
                }
                if (message_Receive.state == "503")                    //接收到文件
确认包
                {
                    message_Send = new Message(0, "515", userName,
ServerNum.name, null);      //确认全部接收
                    postbox_Net.SendMessage(message_Send);
                    throw new Exception("Receive succeed");
                }
                if (message_Receive.state == "504")                    //接收到
文件取消包
                    throw new Exception("Opposite refused the connection. ");
            }
        }
        catch (Exception exception)
        {
            label_inform.Text = exception.Message;
            fileStream.Close();
            if (exception.Message == "Opposite refused the connection. ")
                File.Delete(file_receiveing);
            file_receiveing = null;
            postbox_Net.ClosePostbox();
            form_fileReceive.Close();
            form_fileReceive = null;
        }
```

```
        }
```
自主编写：XXX

（十）关键代码 10：客户端语言通话线程

```
void Call()
        {
            if (ifSender)
            {
                if (!SenderPrepare())
                    return;
            }
            else
            {
                if (!ReceiverPrepare())
                    return;
            }
            Thread thread_Send = new Thread(new ThreadStart(SendThread))
            {
                IsBackground = true
            };
            Thread thread_Receive = new Thread(new ThreadStart(RecieveThread))
            {
                IsBackground = true
            };
            thread_Send.Start();
            thread_Receive.Start();
        }
        void SendThread()
        {
            Message message_Send;
            Message message_Receive;
            Byte[] sendData = new Byte[Message.lenth_DateByte];
            var waveIn = new WaveInEvent();
            BufferedWaveProvider bufferedWaveProvider = new
BufferedWaveProvider(waveIn.WaveFormat)
            {
                BufferLength = Message.lenth_DateByte
            };
            waveIn.DataAvailable += (s, a) =>
            {
                bufferedWaveProvider.AddSamples(a.Buffer, 0, a.Buffer.Length);
                if (bufferedWaveProvider.BufferLength > 70000)
                {
                    waveIn.StopRecording();
```

```
                }
            };
            try
            {
                waveIn.StartRecording();
                while (true)      //从麦克风接收数据
                {
                    Thread.Sleep(1000);
                    bufferedWaveProvider.Read(sendData, 0,
bufferedWaveProvider.BufferedBytes);
                    //bufferedWaveProvider.ClearBuffer();
                    if (!calling)
                    {
                        message_Send = new Message(0, "504", user_my, user_other,
null);      //发送文件取消包
                        postbox_Call.SendMessage(message_Send);
                        throw new Exception("call over");
                    }
                    if (postbox_Call.ReceiveMessage(0, out message_Receive, 0) &&
message_Receive.state == "504")
                        throw new Exception("call over");        //接收方请求关闭文件传
输
                    message_Send = new Message(0, "502", user_my, user_other,
sendData);
                    postbox_Call.SendMessage(message_Send);
                }
            }
            catch (Exception exception)
            {
                string result = exception.Message;
                label_inform.Text = "call over";
                calling = false;
                waveIn.StopRecording();
                waveIn.Dispose();
                postbox_Call.ClosePostbox();
            }
        }
        void RecieveThread()
        {
            Message message_Send;
            Message message_Receive;
            var waveIn = new WaveInEvent();
            WaveOutEvent outputDevice = new WaveOutEvent();
            BufferedWaveProvider bufferedWaveProvider = new
```

```
BufferedWaveProvider(waveIn.WaveFormat)
            {
                BufferLength = Message.lenth_DateByte
            };
            bool flag = true;
            try
            {
                while (true)
                {
                    postbox_Call.ReceiveMessage(0, out message_Receive);
                    if (!calling)
                    {
                        message_Send = new Message(0, "504", user_my, user_other,
null);
                        postbox_Call.SendMessage(message_Send);
                        throw new Exception("cancel receive");
                    }
                    if (message_Receive.state == "504")                    //接收到文
件取消包
                        throw new Exception("call over");
                    if (message_Receive.state == "502")                  //接收文件包
                    {
                        //Byte[] data = new Byte[message_Receive.data.Length];     //将
收到的字节数据播放
                        //Array.Copy(message_Receive.data, 0, data, 0,
message_Receive.data.Length);
                        bufferedWaveProvider.ClearBuffer();
                        bufferedWaveProvider.AddSamples(message_Receive.data, 0,
bufferedWaveProvider.BufferLength);
                        IWaveProvider provider = (IWaveProvider)bufferedWaveProvider;
                        if(flag)
                        {
                            label_inform.Text = "connect succeed";
                            outputDevice.Init(provider);
                            outputDevice.Play();
                            flag = false;
                        }
                    }
                    Thread.Sleep(600);
                }
            }
            catch (Exception exception)
            {
                string result = exception.Message;
```

```
                label_inform.Text = "call over";
                outputDevice.Stop();
                calling = false;
                outputDevice.Dispose();
                outputDevice = null;
                postbox_Call.ClosePostbox();
            }
        }
        bool SenderPrepare()
        {
            Message message_Send;
            Message message_Receive;
            int net_Port;
            label_inform.Text = "connecting...";
            postbox.OpenPipe(1);
            try
            {
                message_Send = new Message(0, "300", user_my, user_other,
Encoding.UTF8.GetBytes("call")); //发送 300 包  请求NET穿透地址
                postbox.SendMessage(message_Send);
                while (!postbox.ReceiveMessage(1, out message_Receive, 1000))
                {
                    if (!calling)
                        throw new Exception("Cancel sending");
                }
                if (message_Receive.state == "310")
                    throw new Exception("Opposite refused the connection ");
                if (message_Receive.state != "420")
                    throw new Exception("server error ");
                net_Port =
Convert.ToInt32(Encoding.UTF8.GetString(message_Receive.data));
                postbox_Call.CreateConnect(ServerNum.IPAddress, net_Port);
                message_Send = new Message(0, "500", user_my, user_other, null);
                postbox_Call.SendMessage(message_Send);
                postbox_Call.ReceiveMessage(0, out message_Receive);
                if (message_Receive.state != "501")
                    throw new Exception("Opposite refused the connection ");
                return true;
            }
            catch (Exception exception)
            {
                label_inform.Text = exception.Message;
                postbox_Call.ClosePipe(1);
                postbox_Call.ClosePostbox();
```

```
                return false;
            }
        }
        bool ReceiverPrepare()
        {
            Message message_Send;
            Message message_Receive;
            int net_Port;
            try
            {
                postbox.OpenPipe(1);
                message_Send = new Message(0, "307", user_my, user_other, null);
//发送确认接收报文
                postbox.SendMessage(message_Send);
                postbox.ReceiveMessage(1, out message_Receive);
                postbox.ClosePipe(1);
                if (message_Receive.state != "420")
                    throw new Exception("server error");
                net_Port =
Convert.ToInt32(Encoding.UTF8.GetString(message_Receive.data));
                postbox_Call.CreateConnect(ServerNum.IPAddress, net_Port);
                postbox_Call.ReceiveMessage(0, out message_Receive);          //接收对方
的确认信号
                message_Send = new Message(0, "501", user_my, user_other, null);
                postbox_Call.SendMessage(message_Send);
                return true;
            }
            catch (Exception exception)
            {
                label_inform.Text = exception.Message;
                postbox_Call.ClosePostbox();
                return false;
            }
        }
        private void Form_Calling_FormClosing(object sender, FormClosingEventArgs e)
        {
            calling = false;
        }
```

自主编写：XXX

六、 测试及结果分析

1. 消息转发

测试过程：使用计算机 A 登录账号 0000000001（以下简称账号 1），计算机 B 登录 0000000002（以下简称账号 2）后，A 向 B 发送消息。

结果：B 可以正常收到消息。

2. 离线消息

测试过程：使用计算机 A 登录账号 1，A 向账号 2 发送消息。

结果：计算机 B 登录 2 后，可以正常收到消息。

3. 离线文件

测试过程：使用计算机 A 登录账号 1，A 向账号 2 发送文件。

结果：计算机 B 登录 2 后，可以在离线文件列表中查看自己的离线文件，点击下载按钮后可以开始下载离线文件。

说明：在计算机 A 发送离线文件时，计算机 A 依然可以使用消息发送功能。计算机 B 接收离线文件时，计算机 B 也可以使用消息发送功能。在发送过程和接收过程中，用户都可以取消发送或暂停发送。

4. 在线 NAT 穿透传输文件

测试过程：使用计算机 A 登录账号 1，计算机 B 登录账号 2，账号 1 向账号 2 发送在线文件。

结果：计算机 B 会弹出选择窗口，用户可以选择是否接收该文件。选择了同意接收后，在线文件传输开始，计算机 B 能够正确收到文件。

5. 语音通话

测试过程：使用计算机 A 登录账号 1，计算机 B 登录账号 2，账号 1 向账号 2 发起语音通话。

结果：计算机 B 会弹出选择窗口，用户可以选择是否通话。选择了同意后，语音通话开始。任何一个用户需要结束通话时可以直接关闭通话窗口，即可结束语音通话。

说明：语音通话效果很好，语音清晰，延迟较小，效果可以与市面上的聊天软件相提并论

6. 异常恢复

测试过程：分别在传输在线文件时，语音通话时，传输离线文件时断开网络。

结果：服务器和客户端均可正常感知到网络中断，服务器端会将该用户所占用线程清除，客户端会结束消息发送线程，并提醒用户当前网络已中断。

七、 实验结论

本次实验很好的完成了实验的基本要求和高级要求，其中文件传输功能和语音通话功能效果良好。文件传输速度快、完整、无错。语音通话效果良好，语音清晰，延迟较小。而且在高级要求的基础上我们加入了心跳等功能进一步提高了系统的鲁棒性和稳定性，保证了软件的高可用性。本次实验中，我们尽可能的提高用户的使用体验，尽可能的使系统更加的稳定。

八、 总结及心得体会

在实验 8 中，程序总耗时将近 3 个星期，在其中我们遇到了很多困难，在此挑选较为经典的问题进行阐述。

1. 最初编写 PostBox 时，只有一个线程可以在 PostBox 上进行发送与接收。受到了多路复用技术的启发，在 PostBox 的设计中加入了管道的概念，使得多个线程可以使用同一个 PostBox 进行无干扰的发送与接收。

2. 最初的 PostBox 用于收发文件时，出现接收到的文件乱码的问题。通过不断调试，发现是 C#内部进行参数传递时采用了地址传递的方法，导致在发送队列外修改数据内容时，导致已加入发送队列的数据被修改，最终导致文件接收方收到的文件是乱码。解决方法是每次接受文件时，会为接收到的数据创建一片新的内存空间。

3. 协议设计时，数据中不能包含“%@#”。最后的解决方法是在报文中加入了报文长度的部分，使得接收线程在接收消息时，能够通过报文长度识别出消息边界。

4. PostBox 中有时发送消息时，发送线程无法唤醒。经过调试后发现是多线程问题。解决方法是修改 SendMessage 函数内部的语句顺序，最终问题得到解决。

5. 最初时 PostBox 无法支持多线程同时访问，每个线程必须拥有自己独立的 PostBox。解决方法是通过给 PostBox 内部的属性上锁，并采用了线程安全队列，从而解决了该问题。

6. 服务器最初通过数组队列来存放唤醒各个用户转发消息线程的信号量，

但 C#的数组并不会为其分配空间，导致程序报错。通过不断修改与调试，最终新建 SendSignal 类，并通过 List<T>容器解决内存分配问题。

7. 信号量 AutoResetEvent 声名时，需要为其分配新的内存空间，但 List<T>容器的添加方法同样会为实例化的类分配新空间，导致信号量空间被重复分配。解决方法是在类中编写构造函数，使得 List<T>实例化类时，通过构造函数进行实例化并分配空间。

8. 客户端非正常关闭时，服务器无法感知，导致为该用户创建的线程无法关闭。解决方法是为客户端增加了心跳。

9. 客户端显示文件传输进度窗口时，窗口处于假死状态。无法与用户进行互动。解决方法是采用 MethodInvoker 委托，将窗体显示函数使用该委托运行

10. 客户端进行文件传输时，接收方暂停后发送方依然继续发送文件，导致接收方内存超限。解决方法是更换了文件发送线程中指令的执行顺序。

11. Postbox 在最初设计时没有发送窗口大小，发送队列的大小可以无限增大最终导致程序内存超限，解决方法是给每个发送队列加上上限，超过该上限后发送线程会使调用 sendmessage 函数的线程阻塞，从而限制发送队列的大小。

经过这次实验，我们有如下的心得体会：

1. 掌握了 Socket 编程的基本方法，进一步了解了传输层与应用层的工作交互方式。

2. 在不断地调试中，不断地发现问题解决问题，不仅锻炼了我们的耐心，还让我们深刻体会到一个软件编写所经历的不易。

3. 为了实现实验目的，优化用户体验，提高程序响应度，我们采用了多线程的技术。这导致我们在实验中遇到许多与多线程相关的时序问题，无法使用调试进行查错修改。在这个过程当中，经过我们的不断体会与理解，我们学会了多线程程序的调试方法与解决方法，这让我们能够更好的在未来的实验与工作中获得成功。

4. 为了实现软件的更多功能，报文的状态码不断进行改变，由于备注不清晰，有时导致后续混淆报文的功能与信息。这让我们深刻体会到，在着手编写程序前，设计程序这一步骤的重要性，只有在对程序有了初步的设计基础，思路清晰后，才能简化程序的编写过程与后续的调试过程。

附件

参考资料：

.NET API 浏览器：.NET API 浏览器 | Microsoft Docs

Naudio Github 网站：GitHub - naudio/NAudio: Audio and MIDI library for .NET