**课程设计：基于MFC的即时通信工具的设计与实现**

学 院：信息与电子工程学院

专 业：软件工程

班 级：152、151

学 号：5151594017 、1150299131

姓 名：姚一城 、陈郑强

2018 年 1月2日-2018年 1月8日

**目录**

[一、课程设计目的 2](#_Toc10860)

[二、课程设计任务 2](#_Toc27563)

[三、课程设计内容与实现 2](#_Toc21241)

[3.1分工情况： 2](#_Toc10019)

[3.2停等协议流程 3](#_Toc10967)

[3.3协议格式： 3](#_Toc981)

[3.4发送线程 3](#_Toc24183)

[3.5接受报文处理线程 4](#_Toc10219)

[3.6消息加密解密 4](#_Toc24819)

[3.7服务端记录客户端信息 5](#_Toc20834)

[3.8信号量机制 5](#_Toc23154)

[3.9初始化 5](#_Toc26973)

[3.10发送信息 7](#_Toc21180)

[3.11接收信息 8](#_Toc22252)

[3.12实现自动滚屏 8](#_Toc16739)

[3.13实现回车快捷回复 8](#_Toc13591)

[四、分析与讨论 8](#_Toc14373)

# 一、课程设计目的

通过课程设计，加深对计算机网络的理解，了解如何通过调用传输层（或者更底层）接口实现应用层通信。

# **二、课程设计任务**

### 2.1应用层协议设计：主要设计应用层协议（头）；下层的承载协议（UDP）；

### 2.2服务器端设计：实现用户的区分与鉴别，客户端登录状态记录，登录IP的记录与消息的广播；

### 2.3客户端实现：实现登录，实现通信界面，发送数据进行交互。

### 2.4附加功能：数据传递的加密。

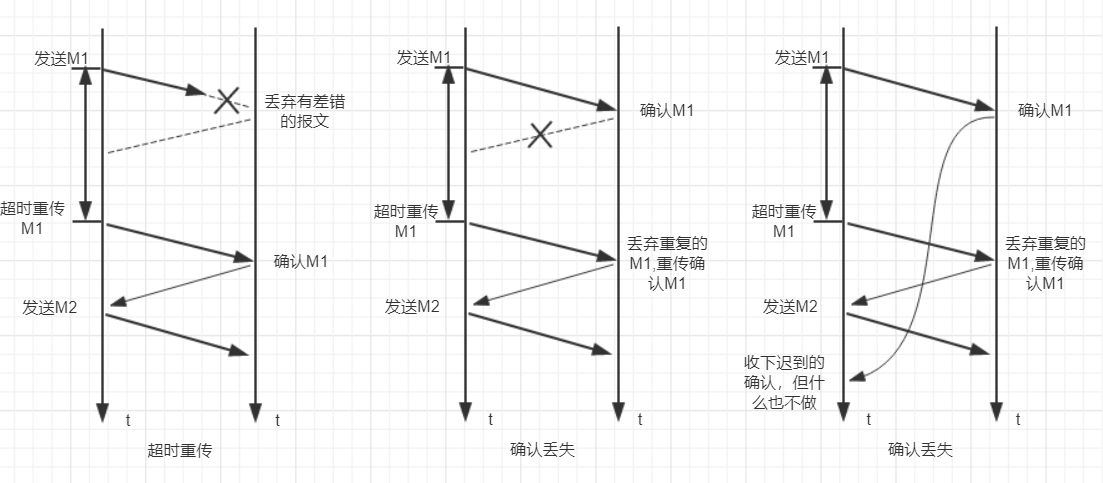
# **三、课程设计内容与实现**

## 3.1分工情况：

姚一城：主要实现了1～7

陈郑强：主要实现了8～12

## 3.2停等协议流程



## 3.3协议格式：

union DataPack {

char buf[BUF\_LEN + 128];

struct DataDetail {

unsigned int seq;

int type;

int sender;

int reciver;

char str[BUF\_LEN];

}data;

};

|  |
| --- |
| 序列号 |
| 类型 |
| 发送者 |
| 接收者 |
| 内容（长度可变） |
| 内容 |

注:其中每一行占32位。

## 3.4发送线程

DWORD WINAPI sendThread(LPVOID pParam) {

DataPack data; //缓存这个发送数据

while (true) {

Sleep(2000);//超时等待

if (curSeq != curACKSeq) {//如果上一个数据还没被确认

::sendto(nativeSocket, encrypt(data.buf), sizeof(data.buf), 0, (sockaddr\*)&addrSrv, sizeof(addrSrv));//超时重传

printf("超时重传了seq = %d\n\n",curSeq);

continue;

}

WaitForSingleObject(sendBufferCount, INFINITE); //相当于P操作

if(!sendBuffer.empty()) {//发送新数据

curSeq++;//发送序号+1

data = sendBuffer.front();

sendBuffer.pop();

data.data.seq = curSeq;

::sendto(nativeSocket, encrypt(data.buf), sizeof(data.buf), 0, (sockaddr\*)&addrSrv, sizeof(addrSrv));//发送这个数据，并缓存

}else {

//发送缓存为空，暂时没有数据需要发送

Sleep(1000);

}

}

}

根据停等协议，按序发送，每次发送完毕设置超时计时器，在规定时间内没有收到对应序列报文的序列号，则重新发送。

## 3.5接受报文处理线程

详见源码

收到新的ACK则开始传送下一个序号的报文，收到迟到的ACK则收下，什么也不做，收到普通消息则打印，收到重复消息则丢弃这个报文，并发送确认ACK。

## 3.6消息加密解密

char\* encrypt(char\* data) {

for (int i = 0; i < BUF\_LEN + 128; i++) {

if (data[i] < 0) {

data[i] += 128; // ==> >=0

}

else {

data[i] -= 128; // ==> <0

}

}

return data;

}

char\* decrypt(char\* data) {

for (int i = 0; i < BUF\_LEN + 128; i++) {

if (data[i] >= 0) { //原来 <0

data[i] -= 128;

}

else {

data[i] += 128; //原来 >=0

}

}

return data;

}

这里仅仅做了一个极为简单的加密处理，对于负的char，将其+128，对于正的char，将其-128（由于char的取值范围为-128～127）

## 3.7服务端记录客户端信息

每一个客户端首先需要“连接”客户端，在服务端生成一个客户ID,并记录对应的Ip，将ID传回客户端在客户端保留。以实现“有状态的连接”。

## 3.8信号量机制

//对接收缓存的互斥信号量

HANDLE reciveBufferMutex;

//用户是否链接成功的同步信号量，初值为0

HANDLE userMutex;

//用于发送线程继续执行的同步信号量，初值为0

HANDLE sendBufferCount;

//用于处理接受数据线程的同步信号量，初值为0

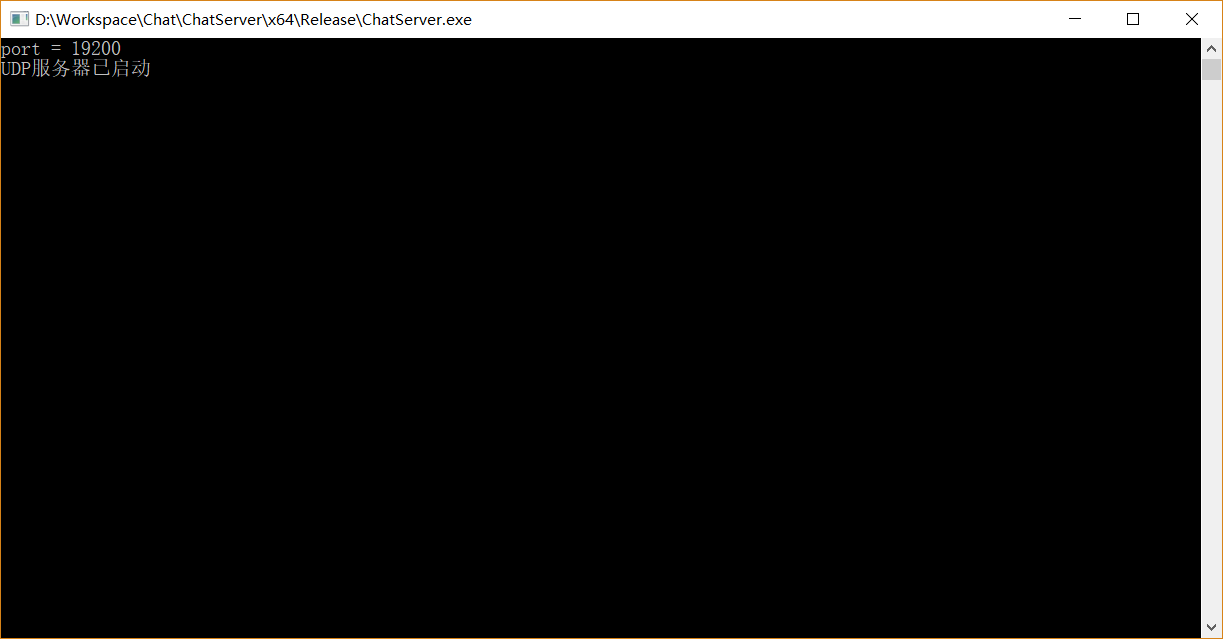
HANDLE reciveBufferCount ;

通过WaitForSingleObject和ReleaseSemaphore实现P,V操作

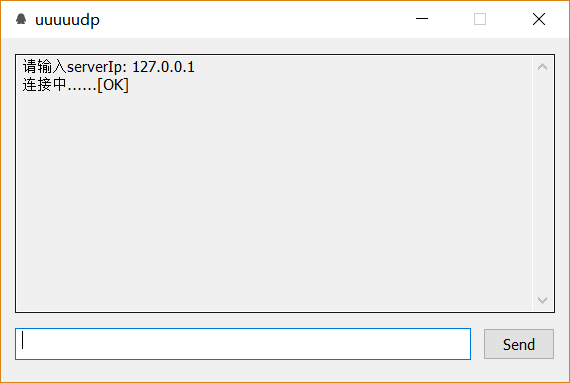
## 3.9初始化

本客户端采用类似匿名聊天室的方案，用户打开本客户端之后只需要根据提示在输入框中输入本次想使用的名字以及服务器的Ip地址，即可进入实时聊天室与当前的其他用户进行交流。

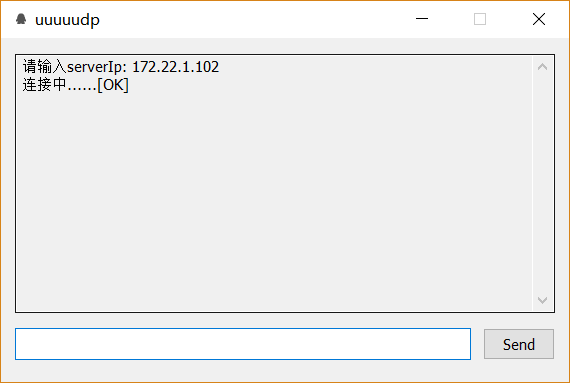
初始化服务端：



初始化客户端1：

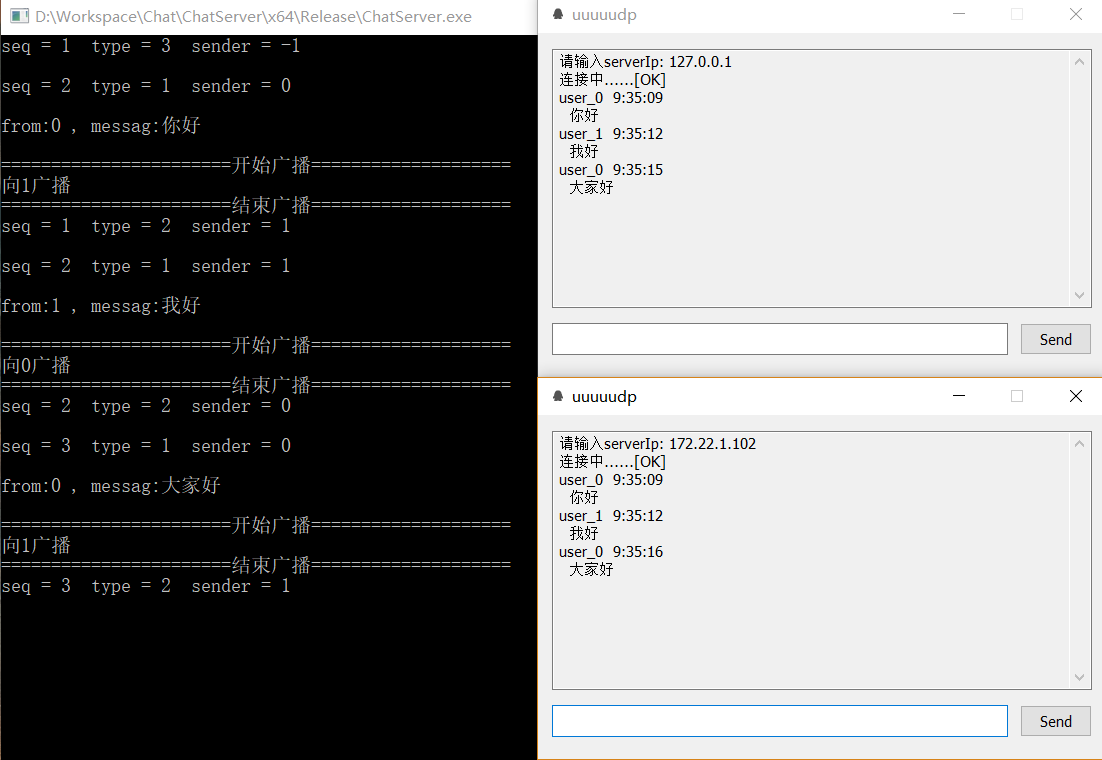


初始化客户端2：



## 3.10发送信息

用户输入一段信息，然后按下回车或者点击send按钮即可触发发送事件，程序会记录当前时间戳，把用户的名字，时间戳，内容封装并加密，形成一份报文，通过udp传输给服务器。服务器接收到报文后进行解码校验，确认无误之后重新加密，然后对所有用户进行广播。



## 3.11接收信息

程序收到服务器传输过来的报文，对报文进行解密，经过校验确认无误之后将内容显示在界面之中。

## 3.12实现自动滚屏

程序在用户进行输入之后会自动滚屏到信息记录最下方，增添了用户的使用友好性。

## 3.13实现回车快捷回复

通过对按键事件信息的监控，检测到触发WM消息事件之时便触发发送信息的方法。增添了用户的使用友好性。

# 四、分析与讨论

姚一城：通过对底层原始udp的进一步控制，在应用层实现了可靠的udp传输，主要基于停等协议，通过设计报文的数据格式，进一步加深了协议的概念。在实际编码中，涉及到c++多线程控制，这里用了大量的信号量机制来实现线程的同步和互斥，学习到了很多线程相关的知识。

陈郑强：通过这次的课程设计，我初步了解了UDP协议的用途，他更适合于非点对点的情况，虽然是不可靠传输，但是经过自行校验之后这个缺点还是可以克服的。在这次课程设计中，我负责制作图形界面，虽然一开始我以为只是一些简单的组件拖拽，但是随着需求的明确，对用户友好性的要求随之提升，我才明白，其实并没有什么事情是简单的。