

《计算机网络》期末大作业



同濟大學
TONGJI UNIVERSITY

网络聊天工具设计

姓名： 黄靖斌 1552333

任康瑞 1552320

课程： 计算机网络

指导老师： 马小峰

一、设计要求：

1、题目

使用 Socket 做一个聊天工具

2、目的

通过实践例子掌握 UDP 或 TCP 协议实现（局域网）的简单在线聊天工具。

培养、锻炼自助分析和解决问题能力。

3、其他要求

两人一小组（自由组合）

小组自主决定使用 UDP 或 TCP 协议

编程语言自主选择

二、设计背景

随着计算机科学技术的飞速发展，网络越来越深刻的改变着人们生活方方面面。各种基于网络的应用技术极大的丰富了我们的日常生活，提高了我们之间的沟通质量。例如 QQ，微信等基于 Internet 的即时聊天工具。这些工具通过网络这个新兴的媒介进行信息交流相比其他传统媒介具有数据量大，实时性强，操作简单，成本低廉等优点。不仅如此即时聊工具还具备许多传统媒介不具备的强大功能，它们能传送文字、声音、影像和文档，而且能更加人性化的显示联络人的名单和通信状态。因而它们在现实生活中受到了广泛的欢迎，这是有目共睹的。

在本学期中，我们学习计算机网络课程，对局域网、TCP/IP 协议

的概念有了一定的了解,也初步具备了设计一个简单的局域网聊天工具的能力。因此,使用 Socket 做一个聊天工具是对自己是否理解这门课程的最好检验。

三、设计目的

主要目的在于检验这学期计算机网络课程的学习质量与学习效果,在设计这样一个用 Socket 实现的局域网聊天工具中,可以更好的了解当前软件的通讯形式,同时加深在计算机网络课程中马老师所讲的理论知识。第一,Socket 套接字。作为应用程序访问连网协议的接口,是聊天工具得以实现的第一步。通过程序创建套接字,绑定本地 IP 地址和端口号,即可以使得在同一局域网下的两个程序“寻找到”对方,建立连接,进行通信。第二,我们设计的客户端、服务端之间建立连接的互动过程,是模仿 TCP 传输过程。TCP 传输面向连接,通信的成功建立在可靠的连接上,且只能进行“点对点”连接。为实现用户上线且能被其他在线用户所知晓,服务器会在与新用户建立连接后,向所有在线用户发送某一新用户刚刚上线。

其次,设计这样一种局域网聊天工具,具有一定的实用性与适用性,在我们平常的生活中可以发挥一定作用。为了适应现代信息化,教学局域网应运而生。而教学局域网内的即时通信能力则显得尤为重要。所以对于学校和公司内部网络等机构,局域网即时聊天的应用还是相当重要的。一个好的局域网通信软件将对学校和公司的运作产生积极的影响。同时,在看腻了互联网聊天工具之后,更换一种新颖的

局域网聊天工具具有更丰富的娱乐性，在一个宿舍中，使用自己研究设计的聊天工具进行聊天也是非常有乐趣的一件事。

最后，这次设计实践也提升了我们的各项技能。第一：由于课题难度较大，我们花费了大量时间去查阅资料，因此，也提高在互联网查阅资料文献、整理有用信息的技能。第二：本次设计是在 Qt 的编程环境下进行，使用了 C++ 编程，因此提高了我们的 C++ 编程技能。

四、程序设计方案

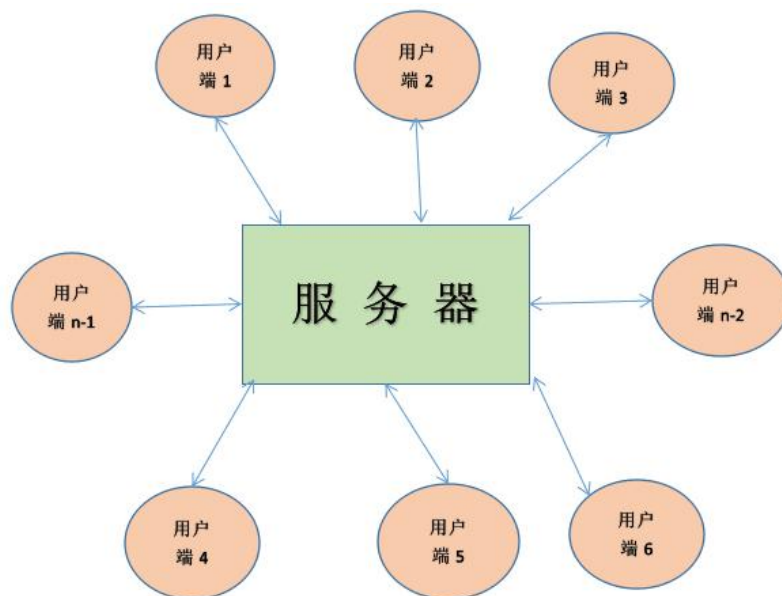
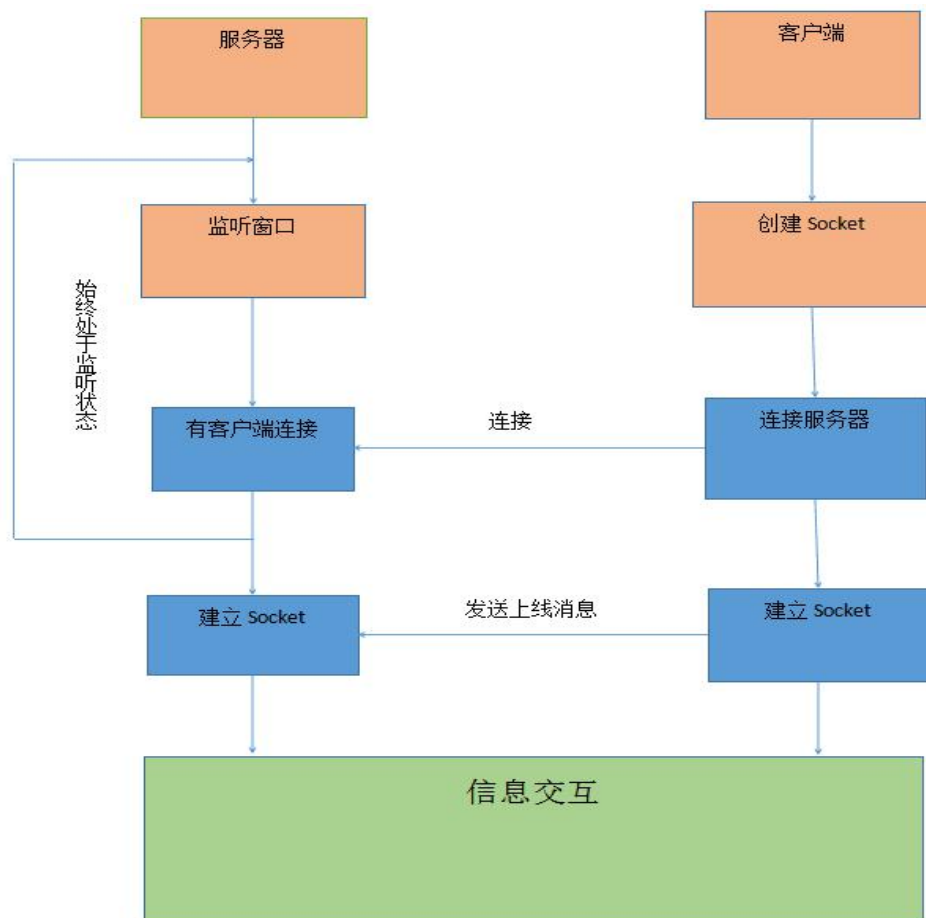
1、框架

我们讨论后决定使用 Qt 环境进行程序编写，因为 Qt 编写的程序界面良好，具有优秀的跨平台特性。

本设计方案采用 TCP 协议，不采用 UDP 协议的原因是其不可靠传输的特点使得数据传输存在一定风险。

在设计中我们编写了两个应用程序，一个服务器端，一个客户端，服务器端负责开启局域网监听行为，客户端具有连接服务器端功能，可以同时开启多个客户端，在用户端上线时，会给服务器发送上线信息，服务器在转发给所有在线用户，以此来提醒所有用户某一新用户已上线。

整体构架如下图：



用户端与服务器相互连接，服务器负责转发
客户端发来的消息

2、流程

该方案采用 TCP 传输控制协议，所以在收发消息前要通过程序和对方建立可靠连接。

服务器端程序流程如下：

创建套接字(socket)；

将套接字绑定到一个本地地址和端口上；

将套接字设为监听模式，准备接受客户请求；

等待客户请求到来：当客户请求到来后，接受连接请求，返回一个新的对应于此次连接的套接字；

用返回的套接字和客户端进行通信；

通信成功，将收到的文本消息转发给其余客户端；

返回，继续等待客户要求。

客户端程序流程如下：

创建套接字(socket)；

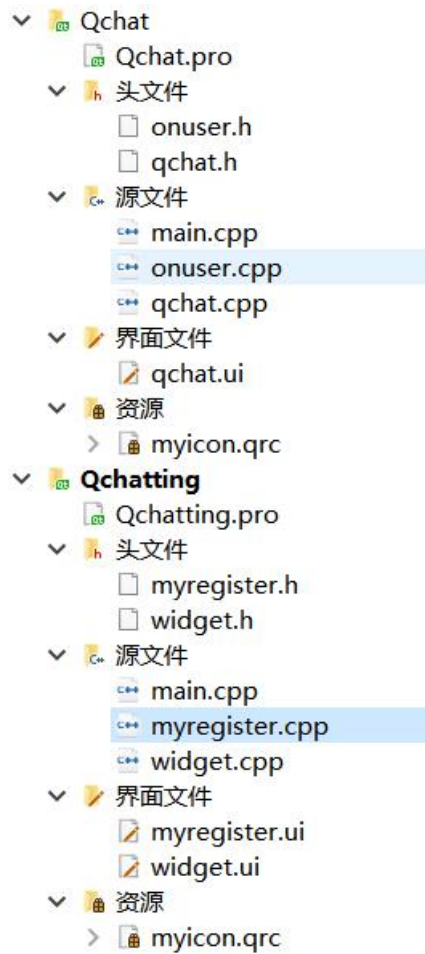
向服务器端发出连接请求；

和服务器端进行通信；

关闭套接字。

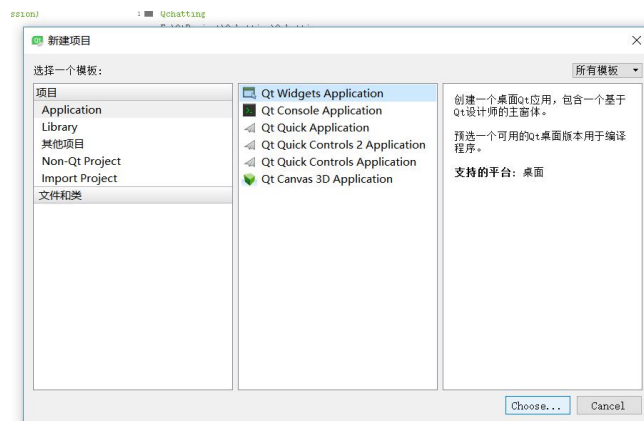
五、编程与实现


程序文件汇总



初始准备

在 Qt 中分别新建 Qt Widgets Application 工程,并命名



2.打开已有的界面文件或者新建一个界面文件.ui,在  中绘制客户端和服务端界面.

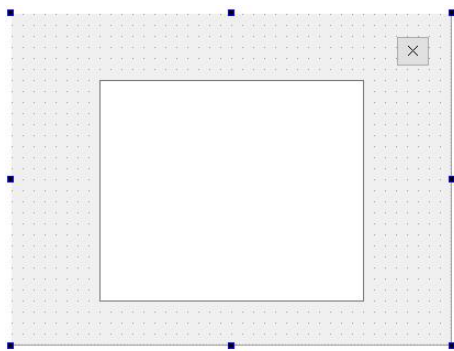


图 1 服务器



图 2 客户端登陆界面

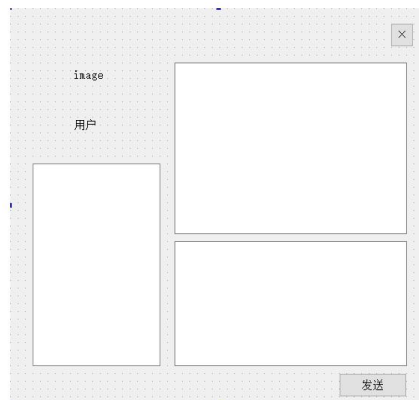


图 3 客户端聊天界面

3.在主函数中打开界面窗口

```
#include "myregister.h"

#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    //打开界面窗口
    myRegister Re;
    Re.show();
    return a.exec();
}
```


4.在.pro 文件和要使用的网络连接的头文件中引入 Qt 网络库

.pro 文件

```
QT += core gui network
```

头文件

```
#include <QtNetwork>
```

界面优化

5.进行字体设置,界面调整,自行编写或者导入 qss 样式表美化界面

```
//加载头像
//界面初始设置
QPixmap image;
image.load("./picture/1.jpg");
ui->setupUi(this);
image=image.scaled(ui->ShowPicture->size());
ui->ShowPicture->setPixmap(image);
ui->ShowPicture->show();//显示
this->setWindowTitle("QQ");
this->setWindowFlags(Qt::FramelessWindowHint);//去掉标题栏
this->setWindowOpacity(0.9);//设置透明1-全体透明
QFont myfont;
myfont.setFamily("幼圆");
myfont.setPointSize(12);
ui->label->setFont(myfont);
ui->showtalk->setFont(myfont);
ui->message->setFont(myfont);
ui->listWidget->setFont(myfont);
```

界面调整

6.编写和导入 qss 样式表

```
//qss样式表导入
ui->setupUi(this);
QFile file("./qss/mystyle.qss");
file.open(QFile::ReadOnly);
QString styleSheet = QLatin1String(file.readAll());
qApp->setStyleSheet(styleSheet);
file.close();
```

```

QFrame
{
    border-image: url(./picture/2.jpg);
}
.QWidget {
    background-color: beige;
}

/* Nice Windows-XP-style password character. */

/* We provide a min-width and min-height for push buttons
   so that they look elegant regardless of the width of the text. */
QPushButton {
    background-color: palegoldenrod;
    border-width: 2px;
    border-color: darkkhaki;
    border-style: solid;
    border-radius: 5;
    padding: 3px;
    min-width: 9ex;
    min-height: 2.5ex;
}

#pushButton {
    width: 2;
    height: 2px;
    background-color: rgba(0,0,0,0);
    border:0;
    color:red;
}

#pushButton:hover {
    background-color: rgba(255,255,0,0.3);
    border:0;
}

QPushButton:hover {
    background-color: khaki;
}

/* Increase the padding, so the text is shifted when the button is
   pressed. */
QPushButton:pressed {
    padding-left: 5px;
    padding-top: 5px;
    background-color: #d0d67c;
}

```

样式表部分展示

网络连接配置

1.服务端:

①设置服务器不断监听端口 2000

```
//TCP监听
server=new QTcpServer();
server->listen(QHostAddress::Any,2000);
connect(server, SIGNAL(newConnection()), this, SLOT(acceptConnection()));
```

②当有客户端连接进来时建立 socket 连接并储存,同时之前储存的用户发送新用户的名字,提示其上线的消息.

```
//监听连接
void Qchat::acceptConnection()
{
    OnUser *on=new OnUser;
    //接受新的连接
    on->socket= server->nextPendingConnection();
    on->socket->waitForReadyRead(150);
    //读取上线信息
    QByteArray dataread = on->socket->readAll();
    QString str = QString::fromLocal8Bit(dataread);
    ui->show->append(str+"上线!");
    on->uname=str;
    //发送之前各用户的名字
    for(int i = 0;i < online->length();i ++)
    {
        QString message=(QString)"online"+"|"+online->at(i)
        on->socket->write(message.toLocal8Bit());
        on->socket->waitForReadyRead(50);
    }
    //将socket加入列表
    online->append(on);
    //发送自己的名字
    for(int i = 0;i < online->length();i ++)
    {
        QString message=(QString)"online"+"|"+str+"|";
        online->at(i)->socket->write(message.toLocal8Bit());
    }
    ui->show->append("已发送上线消息");
    //收到消息反应
    connect(on->socket, SIGNAL(readyRead()), this, SLOT(readClient()));
}
```

Zoom:

③创建槽,将客户端发来信息的信号与槽连接

信号连接槽

```
connect(on->socket, SIGNAL(readyRead()), this, SLOT(readClient()));
```

槽函数编写,当收到信息时读取信息,并根据信息转发给相应用户

```

void Qchat::readClient()
{
    qDebug() << online->length();
    //转发消息
    for(int i = 0;i < online->length();i ++)
    {
        qDebug() << "QDataStream " ;
        QByteArray message=online->at(i)->socket->readAll();
        if(!message.isEmpty()){
            qDebug() << "break " ;
            sendMessage(message);
            break;
        }
    }
}

//发送信息
void Qchat::sendMessage(QByteArray infomation)
{
    QString str = QString::fromLocal8Bit(infomation);
    QStringList info = str.split("|");
    ui->show->append("已转发消息");
    for (int i = 0;i < online->length();i ++)
    {
        if(online->at(i)->uname==info[2]){
            online->at(i)->socket->write(str.toLocal8Bit());
            break;
        }
    }
}

```

2.客户端

①客户端聊天界面启动时与服务端建立 Socket 连接,同时与将自己名字发送给服务端,提示自身上线


```

void Widget::start(QString name,QString add)
{
    this->uname=name;
    this->touser=name;
    //欢迎信息
    ui->label->setText(name);
    ui->showtalk->append("你好,"+uname+"!");
    //socket连接
    client = new QTcpSocket(this);
    client->connectToHost(QHostAddress(add),2000);
    //将名字传给服务器,上线
    client->waitForBytesWritten(50);
    client->write(uname.toLocal8Bit());
    connect(client,SIGNAL(readyRead()),this,SLOT(readMessage()));
}

```

②将从服务读取到信息的信号绑定槽,并对服务端发来消息进行分割,判断其是其他用户的上线消息或者对方用户发给自己的消息。

若是用户上传消息则将用户加入在线用户列表。

若是对方用户的信息则将对方用户名字以及发送的消息显示在聊天

文本框上

```

//接受服务端信息
void Widget::readMessage()
{
    QString check1="message";
    QString check2="online";
    QByteArray dataread=client->readAll();
    //接受信号分离
    QString str = QString::fromLocal8Bit(dataread);
    QStringList info = str.split("|");
    //根据信号类型作出反应
    if(info[0]==check1){
        ui->showtalk->setTextColor(QColor(0,0,255));
        ui->showtalk->append(" "+info[1]);
        ui->showtalk->setTextColor(QColor(0,0,0));
        ui->showtalk->insertPlainText("对 ");
        ui->showtalk->setTextColor(QColor(255,0,0));
        ui->showtalk->insertPlainText("我");
        ui->showtalk->setTextColor(QColor(0,0,0));
        ui->showtalk->insertPlainText(" 说\n");
        ui->showtalk->setTextColor(QColor(0,255,0));
        ui->showtalk->insertPlainText(info[3]);
        ui->showtalk->setTextColor(QColor(0,0,0));
    }else if(info[0]==check2)
    {
        ui->listWidget->insertItem(0,info[1]);
    }
    info.clear();//这句话必须有,不然会出现客户端崩溃的情况,超出范围
}

connect(client,SIGNAL(readyRead()),this,SLOT(readMessage()));

```

③将点击发送按钮的信号绑定槽,发送按钮点击时读取输入文本框内的信息,并包装后向服务器发送.

```
void Widget::on_send_clicked()
{
    QString message;

    //给服务器发消息,发送自己名字+对方名字+要说的话
    message =(QString)"message"+"|"+uname+ "|"+touser+"|"+ui->message->toPlainText()+"|";
    client->write(message.toLocal8Bit());
    //屏幕显示自己要说的话
    ui->showtalk->setTextColor(QColor(255,0,0));
    ui->showtalk->append(" 我 ");
    ui->showtalk->setTextColor(QColor(0,0,0));
    ui->showtalk->insertPlainText("对 ");
    ui->showtalk->setTextColor(QColor(0,0,255));
    ui->showtalk->insertPlainText(touser);
    ui->showtalk->setTextColor(QColor(0,0,0));
    ui->showtalk->insertPlainText(" 说\n");
    ui->showtalk->setTextColor(QColor(0,255,0));
    ui->showtalk->insertPlainText(ui->message->toPlainText());
    //清空屏幕
    ui->showtalk->setTextColor(QColor(0,0,0));
    ui->message->clear();
}
```

④将用户列表选中项改变的信号绑定槽函数,当选中用户列表中的项目时时改变信息发送的用户。

```
//改变说话对象
void Widget::on_listWidget_itemSelectionChanged()
{
    touser=ui->listWidget->currentItem()->text();
}
```

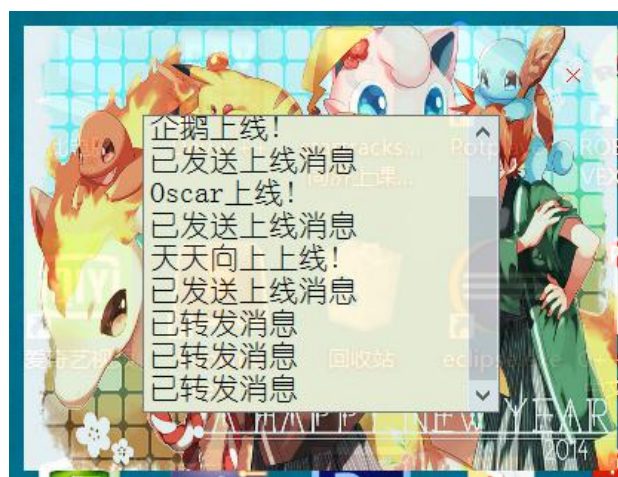
六、设计成果展示、



登陆界面

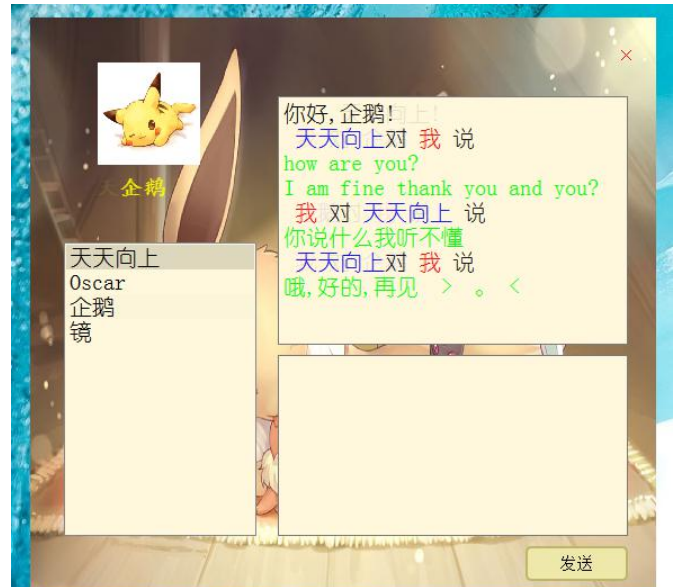


用户登陆



用户上线

上线后界面：



在线用户，互相发送信息，在界面的左下角为在线用户，左上角为用户头像。右半面为发送信息界面

七、设计总结和心得

对于这次我们的大作业设计，我们还是挺满意的。在拿到设计题目之后，我们在网上搜集了大量资料，决定要使用的网络库，起初我们使用的是 boost 的 `boost::asio` 库，但是在使用了一段时间后在与 qt 的连接上出现了问题，由于我们希望能够好好完成这次设计，制作精美的页面，我们放弃了 `boost::asio`，而使用的 Qt 自带的网络库进行编程。我们在网上查阅的大量资料，很长一段时间内陷入了多用户聊天的纠结中，也一直在纠结使用同步还是异步方式进行通讯，同时我们对于多线程的编程也懵懵懂懂，使用 boost 的时候一直在进行多线程的尝试，好在 Qt 的网络库 `socket` 的使用中封装了多线程的配置。最后我们找到了一篇能够较好解决这一问题的文章，我们编写的服务端将连接用户的 `socket` 以链表的形式储存了起来，这样就便于实现多用户的连接。

我们的设计有许多的优点

①使用了 TCP 连接，较为可靠

②界面制作精美，对用户友好

也有许多的不足

①中文输入存在着偶尔使服务端崩溃的状况

②无法对下线的用户发送信息，由于时间原因也没有做到下线用户的删除工作。

③密码输入只是形式，未建立后台的数据库

④ 未配置其他多方面功能，比如修改头像等。

总的来说这次设计是我们的一次努力的尝试，尽管不能做到尽善尽美，但是成功制作出了这样一个程序还是让我们非常有成就感的，这次的程序只是一个雏形，可留待我们后面去完善。我们在这次的设计中学会了网络 socket 编程的方法，也学会了去搜寻资料，向学长，同学，网友们求助，这让我们深刻地感觉到，知识就是通过不断地询问，查找才能够这样传送下来的。也让我们决定培养自己的能力，当理解足够深刻时，能够利用自己的知识，来帮助他人解决问题，能将方法和知识不断地传给需要的人。