合肥工業大學

《网络工程师综合实训》报告

设计题目：**1.1** 局域网实时通讯工具

学生姓名：任俊生

学　　号：**2019214608**

专业班级：计算机科学与技术 **19-1** 班

2021 年 **12** 月

一、设计要求

1. 熟悉开发工具（Visual Studio、C/C++、Qt 等）的基本操作；

2. 了解各平台基于对话框的应用程序的编写过程；

3. 对于 Socket 编程建立初步的概念。

4. 熟悉 Sock API 主要函数的使用；

5. 掌握相应开发工具对 Socket API 的封装；

6. 制作基于局域网的一对一网络即时通讯工具，实现基本数据的网络传输。

7. 支持一对一客户服务器双向通信的同时，支持多个客户端同时与服务器通信

8. 客户端服务端成对使用，发送接受的信息整合在一个程序中

二、开发环境与工具

系统环境：Windows 10

开发工具：QT Creater 4.11.1 （C++）

编译器：MinGW 5.7.2 64bit

三、设计原理

基于 Tcp 协议的 socket 通信，通过 TcpServer 类，服务端创建一个 socket 对进行监听,当客户端 socket 发送连接请求后进行连接,连接成功后进行 socket 通信，断开连接后，对 socket 进行关闭。

四、系统功能与模块划分

客户端：

创建一个 socket 对象，通过 socket 对象对指定 IP 地址和端口的服务端发送连接请求，服务端同意连接请求后与服务端进行双向通信。

以按钮为模块单位进行划分。

1. “连接”模块：

读取输入的 IP 地址与端口，进行发送连接请求，在 30s 内如果连接成功则连接成功，否则无响应则连接超时。连接成功后，对发送按键状态变为‘可用’。“连接”按键功能转为“断开连接”。断开连接为关闭 socket 与服务端建立的连

接。

2. “发送”模块：

连接建立后，获取当前系统时间，同时获取输入文本框中的内容，将文本框中内容提取转换为 Qstring 对象，转换成 utf-8 格式通过 socket 发送到服务端。

3. “读取”模块

连接成功建立后，Socket 接收到服务端发送来的信息，通过提前绑定好的信号与槽函数，触发 socket_Read_Data 函数，读取 socket 缓冲区数据，然后输出到输出文本框中。

4. “断连”模块

连接成功建立后，socket 收到服务端发来的断连请求，通过提前绑定好的信号与槽函数，触发 socket_Disconnected 函数，使发送按键失效，socket 连接自动关闭。

服务端：

创建一个 TcpServer 对象，作为服务端的主要载体，创建一个 hashmap 为正在通信的客户端配置指定的序号和 TcpSocekt 指针。以区分正在通信的客户端和选择某一个客户端进行通信。服务端主要组件包括端口号文本框，用于输入侦听的端口号。输出文本框：将接受到的信息和发送的信息输出到文本框中。输入文本框：将发送的信息输入到该文本框中，点击发送按钮触发发送时间。客户端表：显示当前与该服务端连接的所有客户端的相关信息包括索引号、目标 ip 地址、目标端口号。Combox：选择指定的客户端进行发送信息。

以按钮为功能模块进行划分：

1. “侦听”按钮：

读取端口号，调用 Server 对象的 listen 函数对指定的端口进行监听，同时将按键文字改为“取消侦听”，再次按下后，Server 对象取消侦听并断开与所有客户端的连接。

2. 新连接模块：

当有新的客户端请求连接时，创建一个 socket，并分配一个索引值，将索引值及其对应的 socket 对象指针压入 hash 表，绑定接收数据和断开连接的信号槽。并将连接成功的信息发送到输出文本框中，更新 combox 和客户端表

3. 读取数据模块：

当有客户端 socket 发送给服务端消息时，提前绑定的槽会触发槽函数，槽函数读取缓冲区数据，并附上时间和 from 信息输出到输出文本框中。

4. 断开连接：

当有客户端 socekt 断开连接时，自动触发槽函数，将该 socket 在 hash 表中删除，并调用 deletelater，在下一个执行周期将其删除。更新 combox 和客户端表

5. “发送“按钮：

将输入文本框中的信息转为 utf-8 格式，然后获取 combox 中选中的客户端，将其发送给该客户端。

五、设计步骤

客户端：

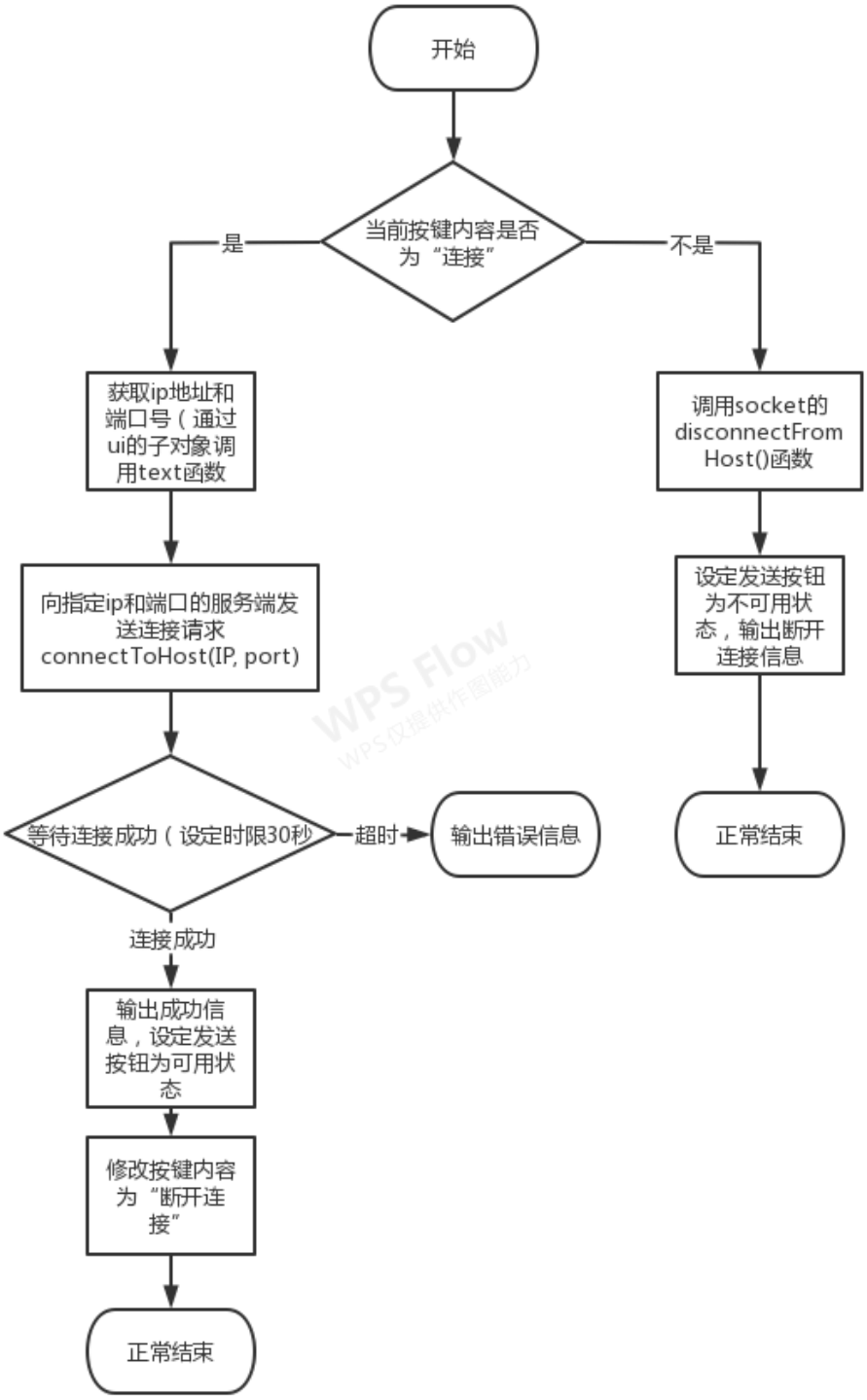
主要功能模块为 “连接”模块

核心代码

```
void MainWindow::on_connect_btn_clicked()
{
    if(ui->connect_btn->text() == tr("连接"))
    {
        QString IP;
        int port;

        //获取 IP 地址
        IP = ui->ip_addr->text();
        //获取端口号
        port = ui->port_num->text().toInt();

        //取消已有的连接
        socket->abort();
        //连接服务器
        socket->connectToHost(IP, port);

        //等待连接成功
        if(!socket->waitForConnected(30000))
        {
            QDateTime current_date_time =QDateTime::currentDateTime();
            QString current_date =current_date_time.toString("yyyy.MM.dd hh:mm:ss.zzz ddd");
            current_date="<font color = gray>" + current_date + "</font>";
```

```
            ui->_output->append(current_date);
            ui->_output->append("连接失败");
            qDebug() << "Connection failed!";
            return;
        }
        QDateTime current_date_time =QDateTime::currentDateTime();
        QString current_date =current_date_time.toString("yyyy.MM.dd hh:mm:ss.zzz ddd");
        current_date="<font color = gray>" + current_date + "</font>";
        ui->_output->append(current_date);
        ui->_output->append("连接成功");
        qDebug() << "Connect successfully!";

        //发送按键使能
        ui->send_btn->setEnabled(true);
        //修改按键文字
        ui->connect_btn->setText("断开连接");
    }
    else
    {
        //断开连接
        socket->disconnectFromHost();
        //修改按键文字
        ui->connect_btn->setText("连接");
        ui->send_btn->setEnabled(false);
    }


    }
```

流程图：



服务端核心模块为“新连接”模块：

代码：

```
void MainWindow::server_New_Connect()
{
    //获取客户端连接
```

```
    QTcpSocket* temp_sock;
    //接受 server 发来的 socket
    socket = server->nextPendingConnection();
    temp_sock = socket;
    ++count;
    //将索引和 socket 指针压入 hash 表
    socket_list.insert(count,temp_sock);
    //连接 QTcpSocket 的信号槽，以读取新数据
    QObject::connect(socket_list[count], &QTcpSocket::readyRead, this,
&MainWindow::socket_Read_Data);
    QObject::connect(socket_list[count], &QTcpSocket::disconnected,
this, &MainWindow::socket_Disconnected);
    //发送按键使能
    ui->send_btn->setEnabled(true);
    //输出时间和连接成功信息
    QDateTime current_date_time =QDateTime::currentDateTime();
    QString current_date =current_date_time.toString("yyyy.MM.dd
hh:mm:ss.zzz ddd");
    current_date="<font color = gray>" + current_date + "</font>";
    ui->_output->append(current_date);
    ui->_output->append("一个客户端连接成功");
    //combobox 和客户端表中添加条目
    ui->comboBox->addItem(QString::number(count));
    ui->client_list->addItem("序号：  " + QString::number(count)+
                             "  ip 地址" +
socket_list[count]->peerAddress().toString()
                             + "  端口号：  "
+QString::number(socket_list[count]->peerPort()) +
                             "  主机名：  " +
socket_list[count]->peerName());


}
```

流程图：

六、关键问题及其解决方法

关键问题：

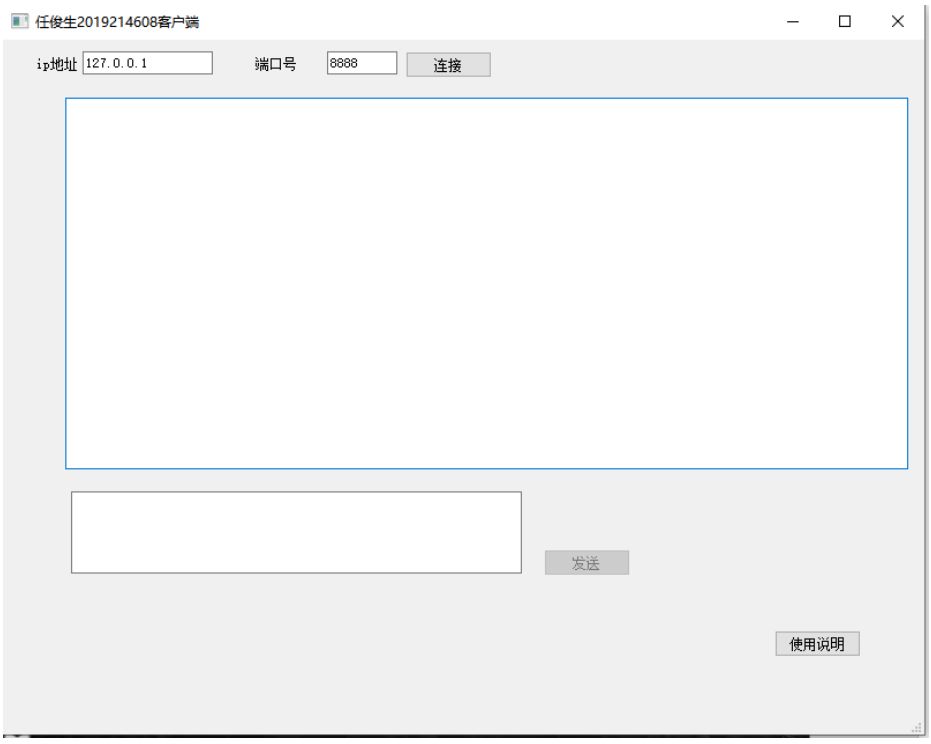
如何实现服务端可以同时与多个客户端进行通信，且为一对一通信而非广播。

解决方法如下

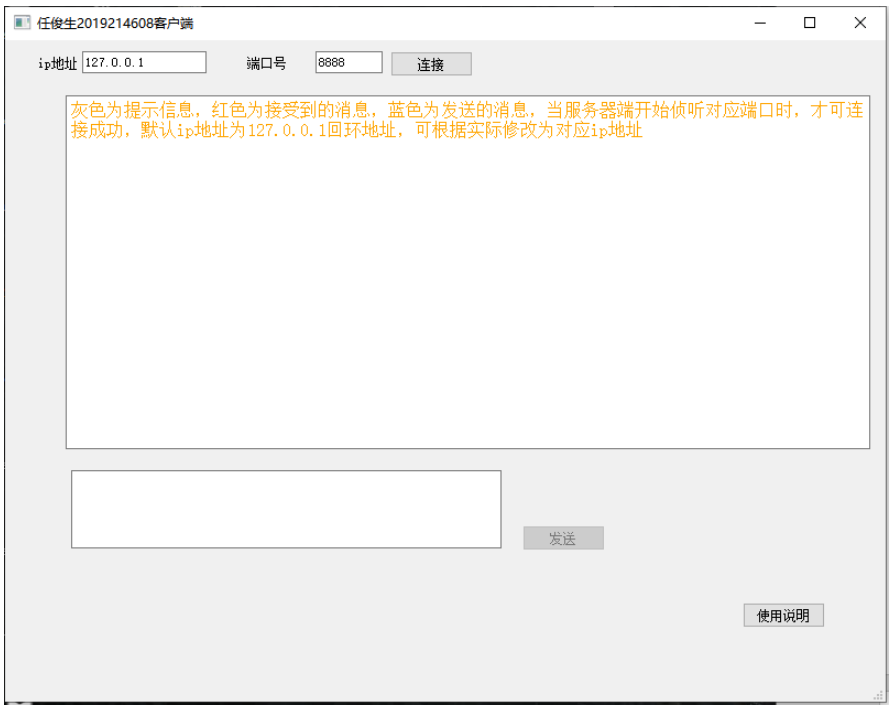
经网络查询，网上大多通过多线程解决该问题，即服务端作为主线程，每收到一个客户端连接请求就创建一个子线程来作为对应的 socket。但是当客户端连接数较大时，需要创建较多线程来处理，作为一个简单的通讯软件，这样处理减少了一定的代码难度但是可能会消耗过多的计算机资源，而且需要继承改写 Tcpsocket 类和 Tcpserver 类，增加了程序的不确定性，所以没有采用这种解决方法，采用了比较节省资源的解决方法，利用 QT 自带的 QHash 表和 combobox 组件进行一对一通信，将 QHash 的 pair 格式为<int,QTcpSocekt*>，int 为索引值用于快速寻找对应的 socket 指针，当有新连接时，将其压入 QHash，和 comboBox,通过在 comboBox 中选择指定的索引值,在 QHash 中获取对应的 socket 指针，从而得到对应的 socket 然后对其调用 write 函数发送信息。接受信息即将遍历所有 socket 读取缓冲区内容输出到文本框中即可。

七、设计结果

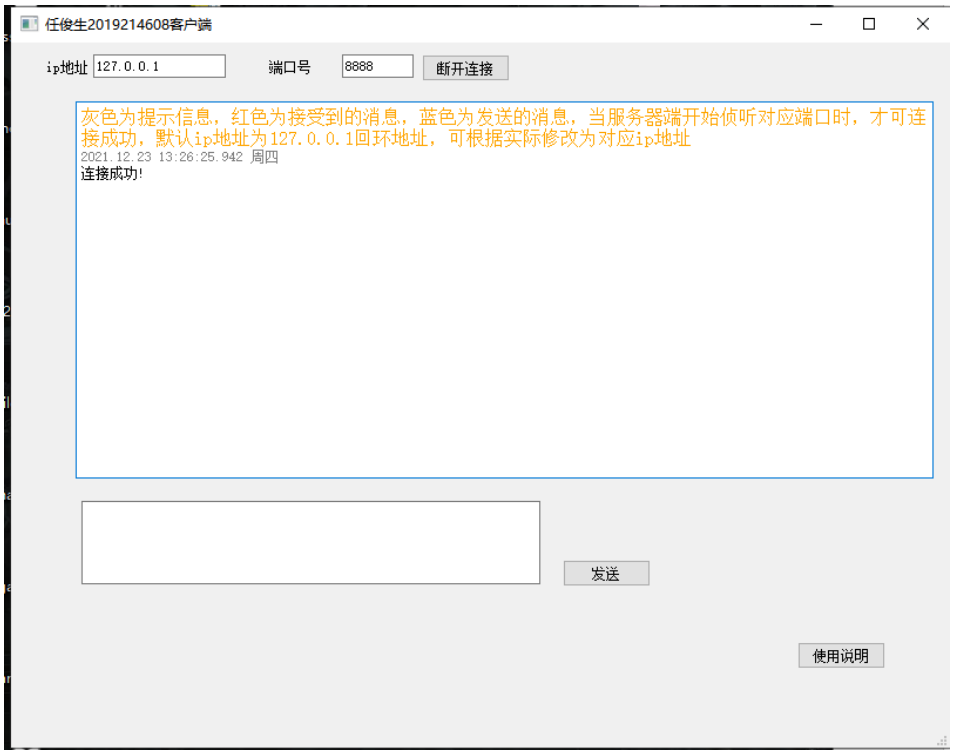
客户端：



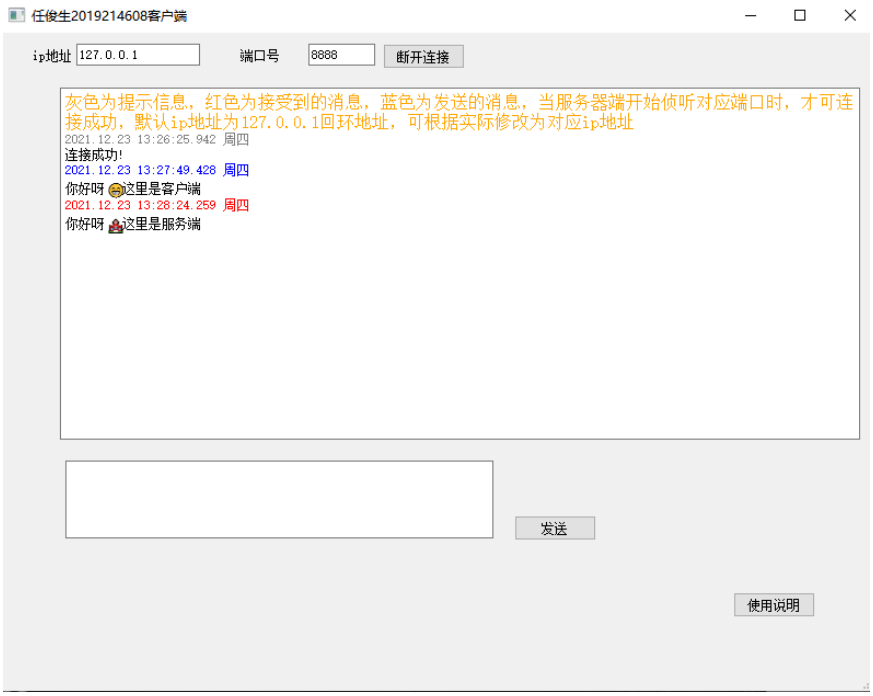
使用说明



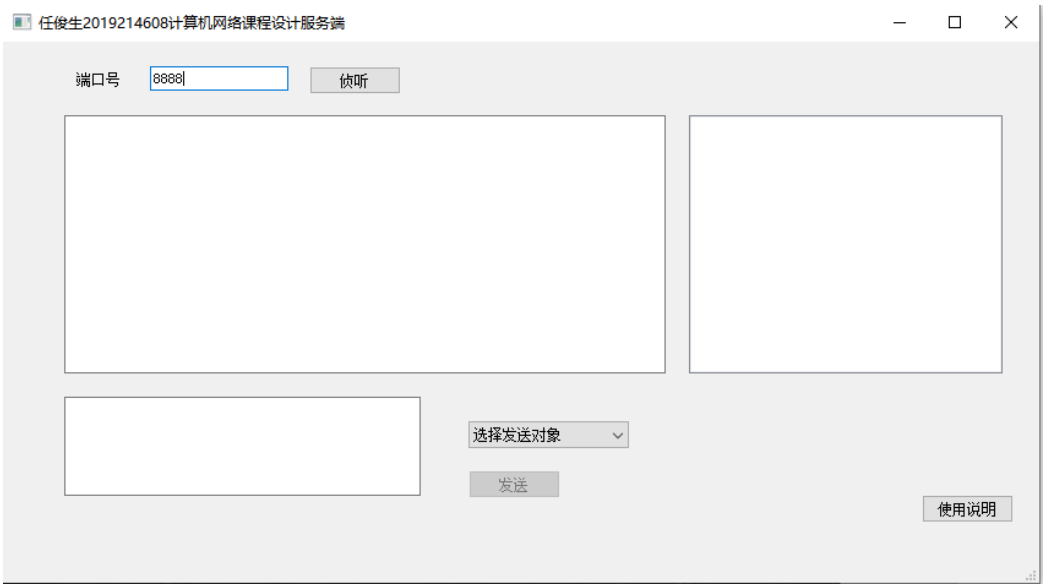
服务端侦听后点击 “连接”



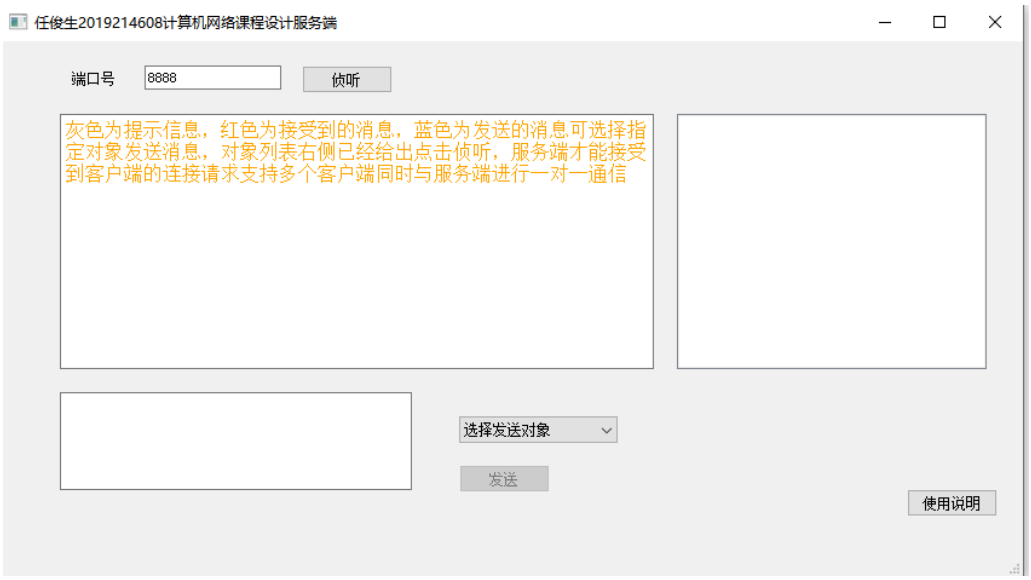
收发信息


任俊生2019214608客户端
ip地址
127.0.0.1
端口号
8888
断开连接
灰色为提示信息，红色为接受到的消息，蓝色为发送的消息，当服务器端开始侦听对应端口时，才可连接成功，默认ip地址为127.0.0.1回环地址，可根据实际修改为对应ip地址
2021.12.23 13:26:25.942 周四
连接成功!
2021.12.23 13:27:49.428 周四
你好呀 这里是客户端
2021.12.23 13:28:24.259 周四
你好呀 这里是服务端
发送
使用说明

服务端：

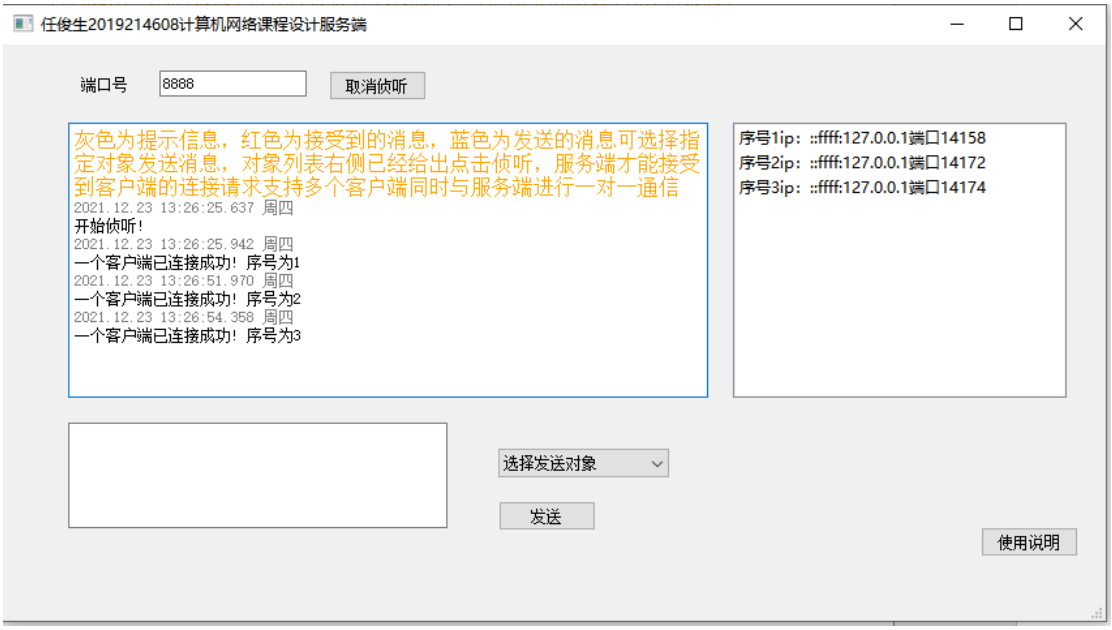

任俊生2019214608计算机网络课程设计服务端
端口号
8888
侦听
选择发送对象
发送
使用说明

点击使用说明



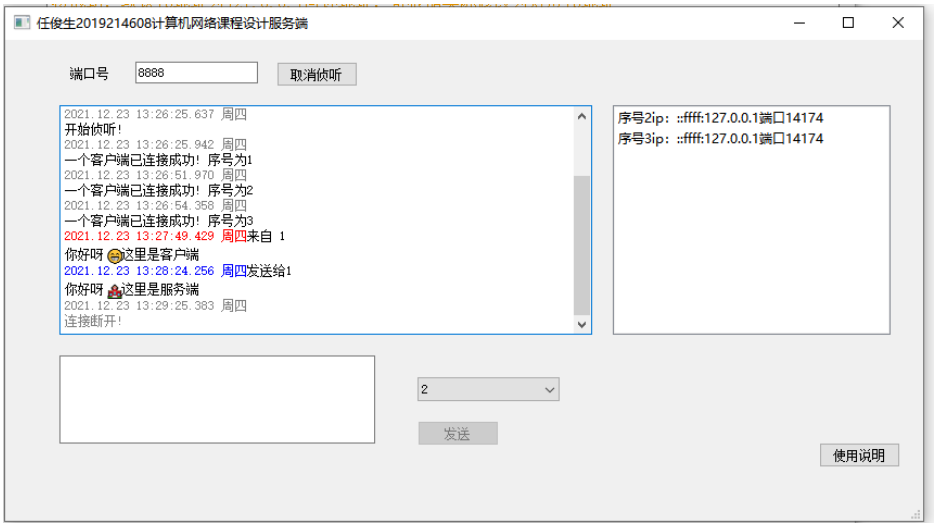
点击“侦听”

多个客户端申请连接后



收发信息

有客户端断开连接



思考题：

1. 实现服务端与多个客户端同时通信 已经实现

2. 实现将接受和发送的信息整合到同一程序中 已经实现

八、使用说明

客户端：

Ip 地址处输入对应的服务端 ip 地址，端口号处输入服务端侦听的端口号，默认缺省给出本机地址方便测试使用，点击“连接”按钮发送连接请求

连接成功后，“连接”按钮功能变为“断开连接”同时“发送”按钮可用

在“发送”按钮左侧文本框中输入内容，点击发送将发送给服务端，同时内容及其时间信息输出到上面的大文本框中。

点击“断开连接”，连接断开。

服务端：

端口号处输入侦听的端口号，点击“侦听”开始侦听来自该端口的连接请求，“侦听”按钮变为“取消侦听”

有连接后，发送按钮可用，选择 comboBox 中的索引（索引对应的客户端信息在客户端表中显示），在发送按钮左侧文本框中输入内容后点击发送，即可向指定客户端发送信息。接受和发送的信息综合显示到上面大文本框中。

点击“取消侦听”，断开所有连接，回到初始状态。

九、设计体会

1. 了解和熟悉了 Windows 环境下 QT 进行 socket 编程的基本步骤，完成了 socket 编程的简单实践

2. 熟悉了 hash 表等常用的重要数据结构，并将其综合到课程设计程序中

3. 学习了QT编译运行生成的.exe文件如何使用windeployqt工具进行程序打包，形成独立的应用程序，脱离编译环境下运行。

反思：通过验收，老师的指导发现，本课程设计缺少实际的应用场景，仍然需要改善，形成一个简单的具有初步应用价值的程序。