



东南大学

信息通信网络概论 实验报告

实验题目： 基于 Qt 的聊天程序设计与实现

姓 名：

学 号：

一. 实验目的

1. 了解双机通信过程，并设计实现双机通信
2. 掌握如果利用 TCP 套接字创建并连接服务器
3. 了解在双机互联构建的局域网环境下如何进行数据交换

二. 实验条件和环境

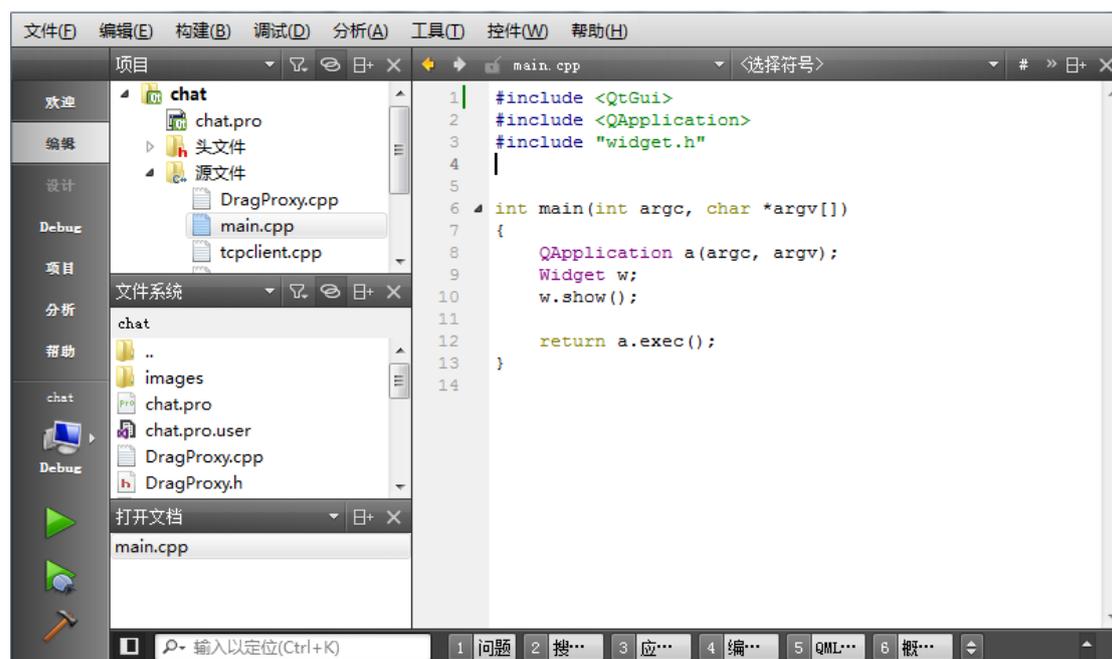
(一) 开发平台:

Qt 5.2.1 简介

Qt是一个1991年由奇趣科技开发的跨平台C++图形用户界面应用程序开发框架。它既可以开发GUI程序，也可用于开发非GUI程序，比如控制台工具和服务器。Qt是面向对象的框架，使用特殊的代码生成扩展（称为元对象编译器(Meta Object Compiler, moc)）以及一些宏，易于扩展，允许组件编程。2008年，奇趣科技被诺基亚公司收购，QT也因此成为诺基亚旗下的编程语言工具。2012年，Qt被Digia收购。2014年4月，跨平台集成开发环境Qt Creator 3.1.0正式发布，实现了对于iOS的完全支持，新增WinRT、Beautifier等插件，废弃了无Python接口的GDB调试支持，集成了基于Clang的C/C++代码模块，并对Android支持做出了调整，至此实现了全面支持iOS、Android、WP。



Qt 的logo以及理念



Qt 5.2.1 的开发环境

(二) 操作系统：Windows操作系统

三. 实验内容

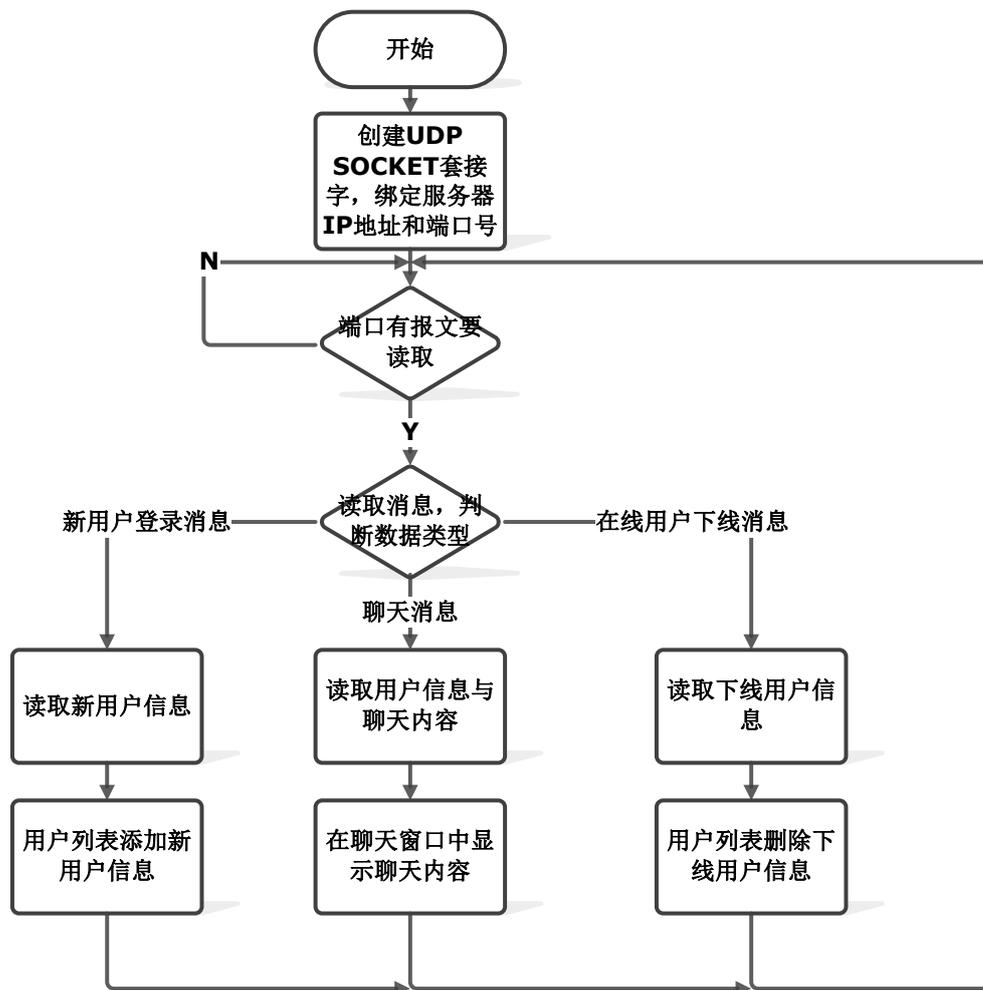
1. 编写一个双机通信程序，编程语言不限
2. UI设计可充分发挥想象，附加功能由自己设计。

注意：本次试验，我们在基础要求上，进一步发挥，完成了局域网聊天程序的设计，该程序实现了以下两个功能：

- 1、基于UDP协议的多人聊天功能；
- 2、基于TCP协议的文件的可靠传输。

四. 实验过程及程序

(一) 系统流程图



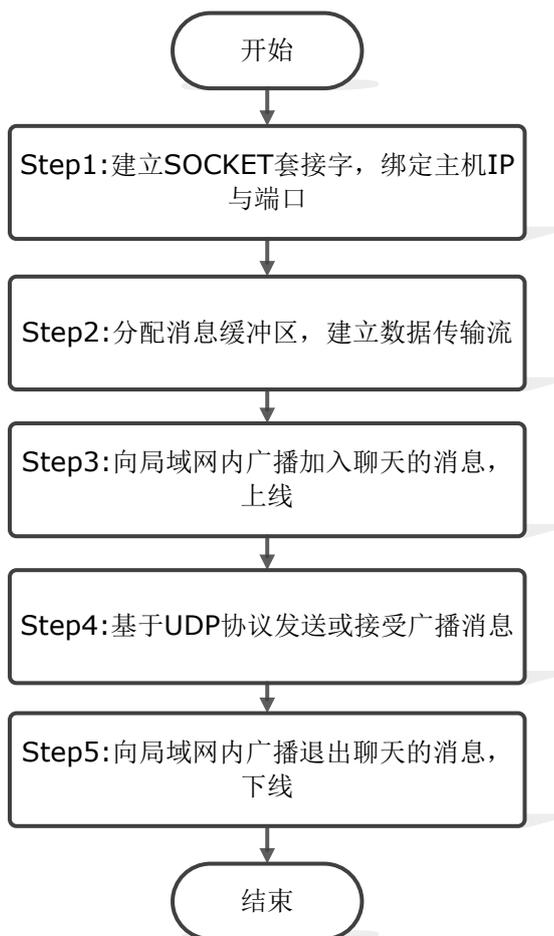
系统流程图

(二) 主要功能代码实现

1、聊天模块

原理：采用 UDP 协议实现局域网内聊天消息的广播。

原理示意图：



用户界面介绍:



主要功能及其具体代码实现:

功能一:

用户列表区

```
// 发送消息
```

```
void sendMessage(MessageType type, QString serverAddress="");
```

```
功能二:
```

```
// 读取并处理消息
```

```
void hasPendingFile(QString userName, QString serverAddress,  
                    QString clientAddress, QString fileName);
```

```
功能一具体代码实现:
```

```
// 使用 UDP 广播发送消息
```

```
// 参数说明: type: 发送内容的数据类型; serverAddress: 服务器 IP
```

```
void Widget::sendMessage(MessageType type, QString serverAddress)
```

```
{
```

```
    // 创建字节数组 data 保存数据, 创建数据流 out 写入数据
```

```
    QByteArray data;
```

```
    QDataStream out(&data, QIODevice::WriteOnly);
```

```
    // 获得本地主机名
```

```
    QString localHostName = QHostInfo::localHostName();
```

```
    // 获得本地 IP 地址
```

```
    QString address = getIP();
```

```
    // 发送消息: 用户名+本地主机名
```

```
    out << type << getUserName() << localHostName;
```

```
    // 处理不同类型的消息
```

```
    switch(type)
```

```
    {
```

```
        // 消息类型为聊天信息
```

```
        case Message :
```

```
            // 发送内容不能为空, 若发送内容为空, 弹出警告窗口
```

```
            if (ui->messageTextEdit->toPlainText() == "") {
```

```
                QMessageBox::warning(0,QStringLiteral("警告"),QStringLiteral("发送  
内容不能为空"),QMessageBox::Ok);
```

```
                return;
```

```
            }
```

```
            // 发送消息: IP 地址+Message
```

```
            out << address << getMessage();
```

```
            ui->messageBrowser->verticalScrollBar()
```

```
->setValue(ui->messageBrowser->verticalScrollBar()->maximum());
```

```
            break;
```

```
        // 消息类型为参与聊天的新请求
```

```
        case NewParticipant :
```

```
            // 发送消息: 新参与者的 IP 地址
```

```
            out << address;
```

```
            break;
```

```
        // 消息类型: 用户下线
```



```
case ParticipantLeft :
    break;
// 消息类型：传送文件
case FileName :
{
    int row = ui->userTableWidget->currentRow();
    QString clientAddress = ui->userTableWidget->item(row, 2)->text();
    // 发送消息：本地主机 IP 地址+客户端 IP 地址
    out << address << clientAddress << fileName;
    break;
}
// 消息类型：拒绝接受文件
case Refuse :
    // 发送消息：服务器地址
    out << serverAddress;
    break;
}
//广播发送字节数组 data 中的数据
udpSocket->writeDatagram(data,data.length(),QHostAddress::Broadcast, port);
}
```

功能二具体代码实现：

```
// 接收并处理 UDP 消息
void Widget::processPendingDatagrams()
{
    while(udpSocket->hasPendingDatagrams())
    {
        // 创建数据报
        QByteArray datagram;
        // 数据包整形?
        datagram.resize(udpSocket->pendingDatagramSize());
        // 读取数据报
        udpSocket->readDatagram(datagram.data(), datagram.size());
        // 创建数据流 in，接收消息
        QDataStream in(&datagram, QIODevice::ReadOnly);
        int messageType;
        in >> messageType;
        QString userName,localHostName,ipAddress,message;
        QString time = QDateTime::currentDateTime()
            .toString("yyyy-MM-dd hh:mm:ss");
        // 接收消息，接收端与发送端的数据类型以及操作一一对应
        switch(messageType)
        {
            case Message:
```



```
in >> userName >> localHostName >> ipAddress >> message;
ui->messageBrowser->setTextColor(Qt::darkBlue);
ui->messageBrowser->setCurrentFont(QFont("Times New
Roman",10));
ui->messageBrowser->append("[ " +userName+" ] "+ time);
ui->messageBrowser->append(message);
break;

case NewParticipant:
in >>userName >>localHostName >>ipAddress;
newParticipant(userName,localHostName,ipAddress);
break;

case ParticipantLeft:
in >>userName >>localHostName;
participantLeft(userName,localHostName,time);
break;

case FileName: {
in >> userName >> localHostName >> ipAddress;
QString clientAddress, fileName;
in >> clientAddress >> fileName;
hasPendingFile(userName, ipAddress, clientAddress, fileName);
break;
}

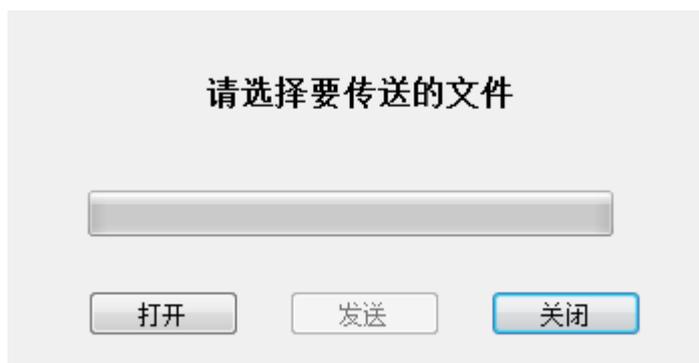
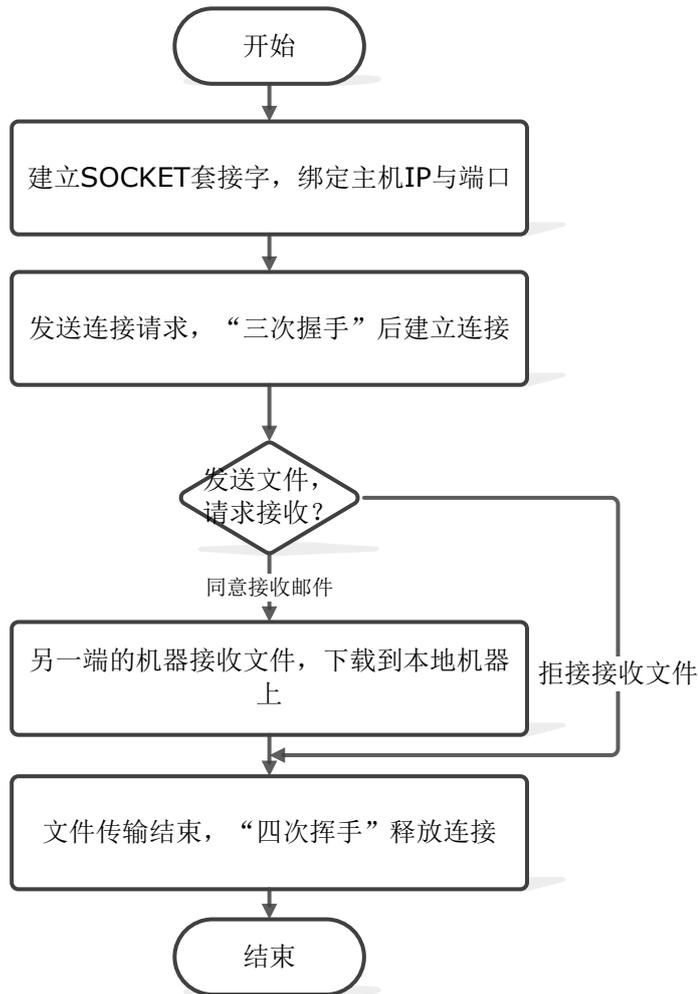
case Refuse: {
in >> userName >> localHostName;
QString serverAddress;
in >> serverAddress;
QString ipAddress = getIP();

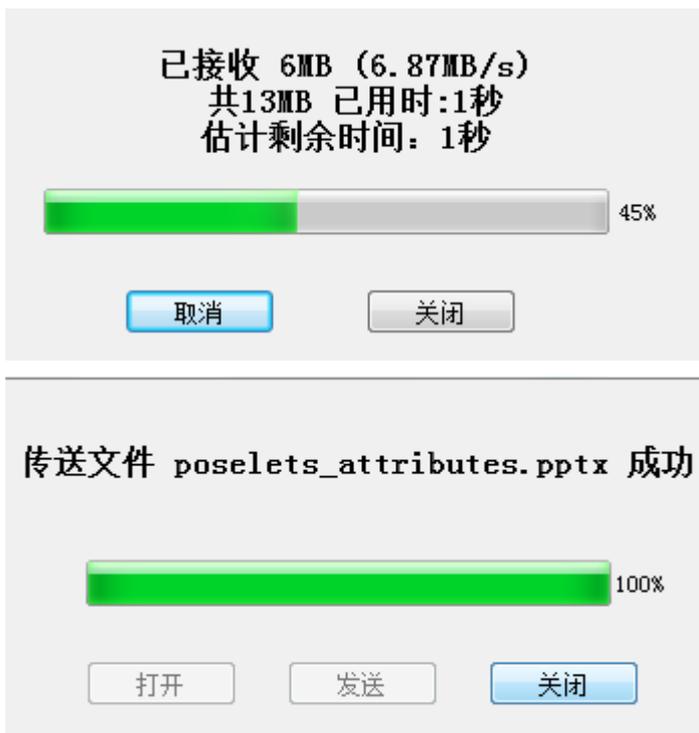
if(ipAddress == serverAddress)
{
server->refused();
}
break;
}
}
}
```

2、文件传输模块:

原理：采用 TCP 协议实现局域网内文件的可靠传输。

原理示意图：





主要功能及其具体代码实现:

功能一:

// 服务器端发送数据

```
void TcpServer::sendMessage();
```

功能二:

// 客户端创建新连接

```
void TcpClient::newConnect();
```

// 客户端读取数据

```
void TcpClient::readMessage();
```

功能一具体代码实现:

// 服务器端发送数据

```
void TcpServer::sendMessage()
```

```
{
```

```
    ui->serverSendBtn->setEnabled(false);
```

```
    clientConnection = tcpServer->nextPendingConnection();
```

```
    connect(clientConnection, SIGNAL(bytesWritten(qint64)),  
           this, SLOT(updateClientProgress(qint64)));
```

```
    ui->serverStatusLabel->setText(QStringLiteral("开始传送文件 %1 !  
").arg(theFileName));
```

```
    localFile = new QFile(fileName);
```

```
    if(!localFile->open((QFile::ReadOnly))){
```



```
        QMessageBox::warning(this, QStringLiteral("应用程序"), QStringLiteral("无法读取文件 %1:\n%2")
                                .arg(fileName).arg(localFile->errorString()));
        return;
    }
    TotalBytes = localFile->size();
    QDataStream sendOut(&outBlock, QIODevice::WriteOnly);
    sendOut.setVersion(QDataStream::Qt_4_7);

    time.start();// 开始计时
    QString currentFile = fileName.right(fileName.size()
                                        - fileName.lastIndexOf('/')-1);
    sendOut << qint64(0) << qint64(0) << currentFile;
    TotalBytes += outBlock.size();
    sendOut.device()->seek(0);
    sendOut << TotalBytes << qint64((outBlock.size() - sizeof(qint64)*2));
    bytesToWrite = TotalBytes - clientConnection->write(outBlock);
    outBlock.resize(0);
}
```

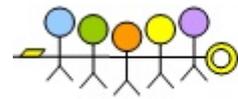
功能二具体代码实现:

```
// 客户端创建新连接, 接受文件
// 创建新连接
void TcpClient::newConnect()
{
    blockSize = 0;
    tcpClient->abort();
    tcpClient->connectToHost(hostAddress, tcpPort);
    time.start();
}

// 读取数据
void TcpClient::readMessage()
{
    QDataStream in(tcpClient);
    in.setVersion(QDataStream::Qt_4_7);

    float useTime = time.elapsed();

    if (bytesReceived <= sizeof(qint64)*2) {
        if ((tcpClient->bytesAvailable()
            >= sizeof(qint64)*2) && (fileNameSize == 0))
        {
            in>>TotalBytes>>fileNameSize;
        }
    }
}
```



```

        bytesReceived += sizeof(qint64)*2;
    }
    if((tcpClient->bytesAvailable() >= fileNameSize) && (fileNameSize != 0)){
        in>>fileName;
        bytesReceived +=fileNameSize;

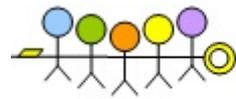
        if(!localFile->open(QFile::WriteOnly)){
            QMessageBox::warning(this,QStringLiteral("应用程序
            "),QStringLiteral("无法读取文件 %1:\n%2.")
            .arg(fileName).arg(localFile->errorString
            ());

            return;
        }
    } else {
        return;
    }
}
if (bytesReceived < TotalBytes) {
    bytesReceived += tcpClient->bytesAvailable();
    inBlock = tcpClient->readAll();
    localFile->write(inBlock);
    inBlock.resize(0);
}
ui->progressBar->setMaximum(TotalBytes);
ui->progressBar->setValue(bytesReceived);

double speed = bytesReceived / useTime;
ui->tcpClientStatusLabel->setText(QStringLiteral("已接收 %1MB (%2MB/s)\n
共%3MB 已用时:%4 秒\n 估计剩余时间: %5 秒")
    .arg(bytesReceived / (1024*1024))
    .arg(speed*1000/(1024*1024),0,'f',2)
    .arg(TotalBytes / (1024 * 1024))
    .arg(useTime/1000,0,'f',0)
    .arg(TotalBytes/speed/1000 -
useTime/1000,0,'f',0));

if(bytesReceived == TotalBytes)
{
    localFile->close();
    tcpClient->close();
    ui->tcpClientStatusLabel->setText(QStringLiteral("接收文件 %1 完毕")
    .arg(fileName));
}
}
}

```



3、附加功能模块

(字体、加粗、颜色等的修改，聊天记录的导出)

代码实现:

// 更改字体族

```
void Widget::on_fontComboBox_currentFontChanged(QFont f)
{
    ui->messageTextEdit->setCurrentFont(f);
    ui->messageTextEdit->setFocus();
}
```

// 更改字体大小

```
void Widget::on_sizeComboBox_currentIndexChanged(QString size)
{
    ui->messageTextEdit->setFontPointSize(size.toDouble());
    ui->messageTextEdit->setFocus();
}
```

// 加粗

```
void Widget::on_boldToolBtn_clicked(bool checked)
{
    if(checked)
        ui->messageTextEdit->setFontWeight(QFont::Bold);
    else
        ui->messageTextEdit->setFontWeight(QFont::Normal);
    ui->messageTextEdit->setFocus();
}
```

// 斜体

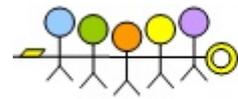
```
void Widget::on_italicToolBtn_clicked(bool checked)
{
    ui->messageTextEdit->setFontItalic(checked);
    ui->messageTextEdit->setFocus();
}
```

// 下划线

```
void Widget::on_underlineToolBtn_clicked(bool checked)
{
    ui->messageTextEdit->setFontUnderline(checked);
    ui->messageTextEdit->setFocus();
}
```

// 更改颜色

```
void Widget::on_colorToolBtn_clicked()
{
```



```
    color = QColorDialog::getColor(color, this);

    if (color.isValid()) {
        ui->messageTextEdit->setTextColor(color);
        ui->messageTextEdit->setFocus();
    }
}

// 处理字体格式的更改
void Widget::currentFormatChanged(const QTextCharFormat &format)
{
    ui->fontComboBox->setCurrentFont(format.font());

    // 如果字体大小出错(因为我们最小的字体为 9)，使用 12
    if (format.fontSize() < 9) {
        ui->sizeComboBox->setCurrentIndex(3);
    } else {
        ui->sizeComboBox->setCurrentIndex( ui->sizeComboBox

->findText(QString::number(format.fontSize())));
    }
    //界面调整
    ui->boldToolBtn->setChecked(format.font().bold());
    ui->italicToolBtn->setChecked(format.font().italic());
    ui->underlineToolBtn->setChecked(format.font().underline());
    color = format.foreground().color();
}

// 保存聊天记录按钮
void Widget::on_saveToolBtn_clicked()
{
    // 要保存的聊天记录不为空
    if (ui->messageBrowser->document()->isEmpty()) {
        QMessageBox::warning(0, QStringLiteral("警告"), QStringLiteral("聊天记录
为空白，无法保存！"), QMessageBox::Ok);
    } else {
        QString fileName = QFileDialog::getSaveFileName(this,
                                                    QStringLiteral("
保存聊天记录"), QStringLiteral("聊天记录"), QStringLiteral("文本(*.txt);All
File(*.*)"));
        if(!fileName.isEmpty())
            saveFile(fileName);
    }
}
}
```

```
// 保存聊天记录
bool Widget::saveFile(const QString &fileName)
{
    QFile file(fileName);
    if (!file.open(QFile::WriteOnly | QFile::Text)) {
        QMessageBox::warning(this, QStringLiteral("保存文件"),
            QStringLiteral("无法保存文
件 %1:\n %2").arg(fileName)
                .arg(file.errorString()));
        return false;
    }
    QTextStream out(&file);
    out << ui->messageBrowser->toPlainText();

    return true;
}

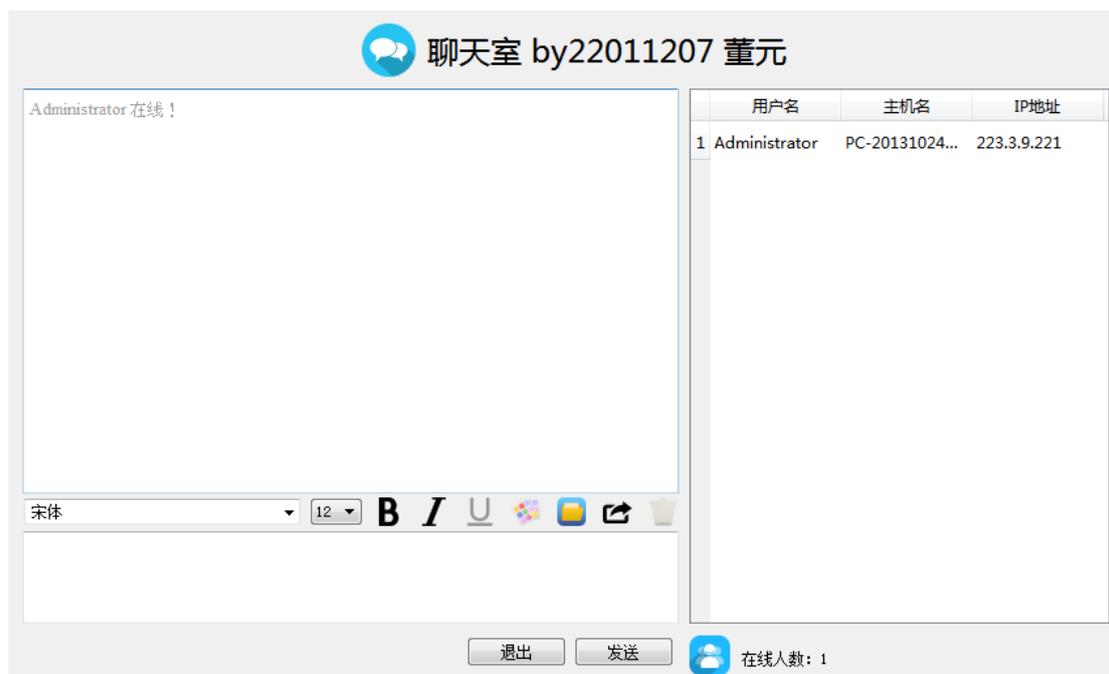
// 清空聊天记录
void Widget::on_clearToolBtn_clicked()
{
    ui->messageBrowser->clear();
}
```

五、实验结果

1.聊天模块：实现局域网内的多人聊天；

说明：由于本聊天程序实现的是局域网内的多人聊天，故在现有实验条件下不方便在多机上进行测试，故以下实验结果的截图均基于同一台计算机。

(1) 主界面

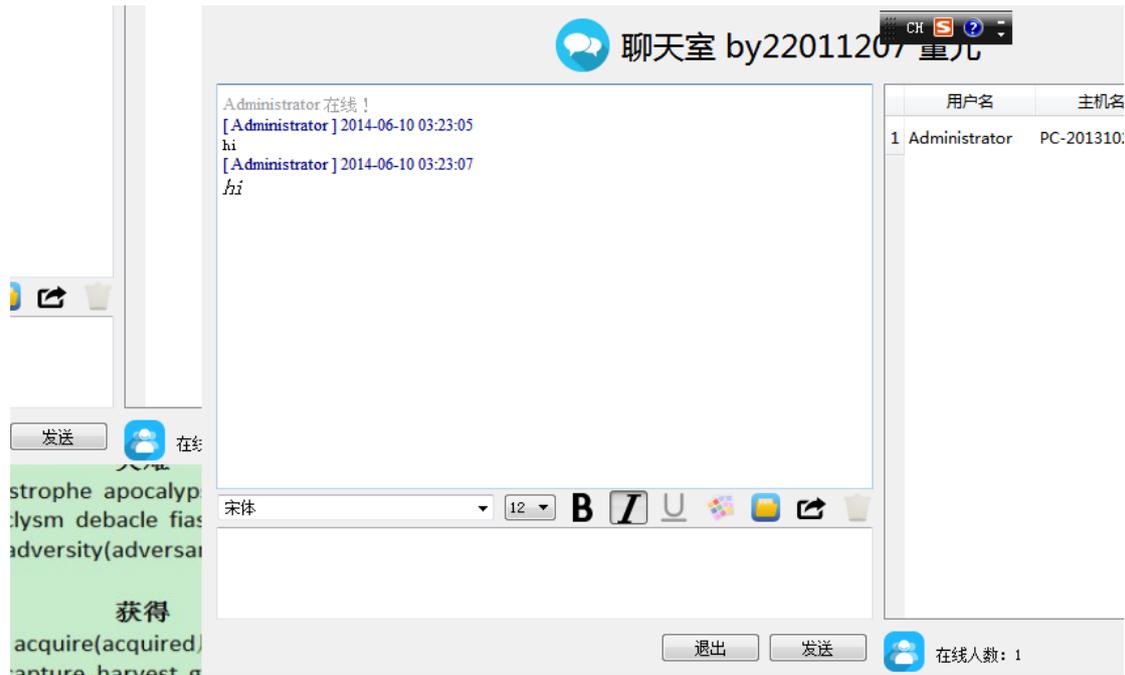


(2) 多机通讯

如下图所示，通过设置不同字体格式来区分通讯的程序，可看出两个程序通讯时的结果：



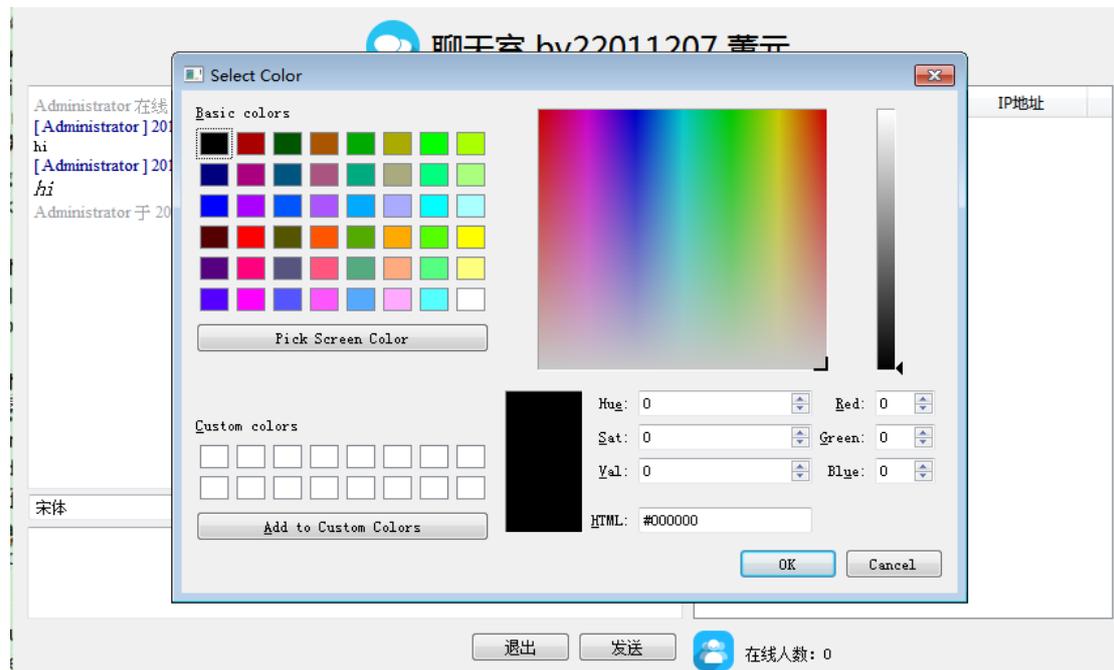
通信 A 采用加粗的字体格式

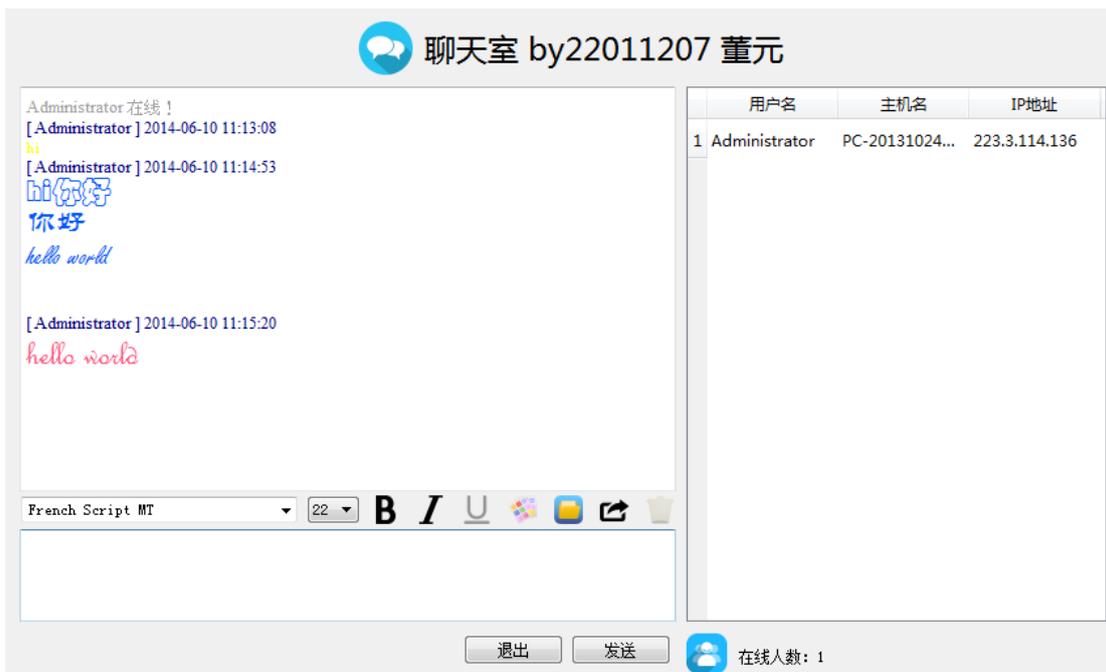


通信方 B 采用斜体的字体格式

2.聊天消息格式设置模块:

该功能可实现聊天消息字体，字号，颜色，加粗，斜体，下划线的设置。



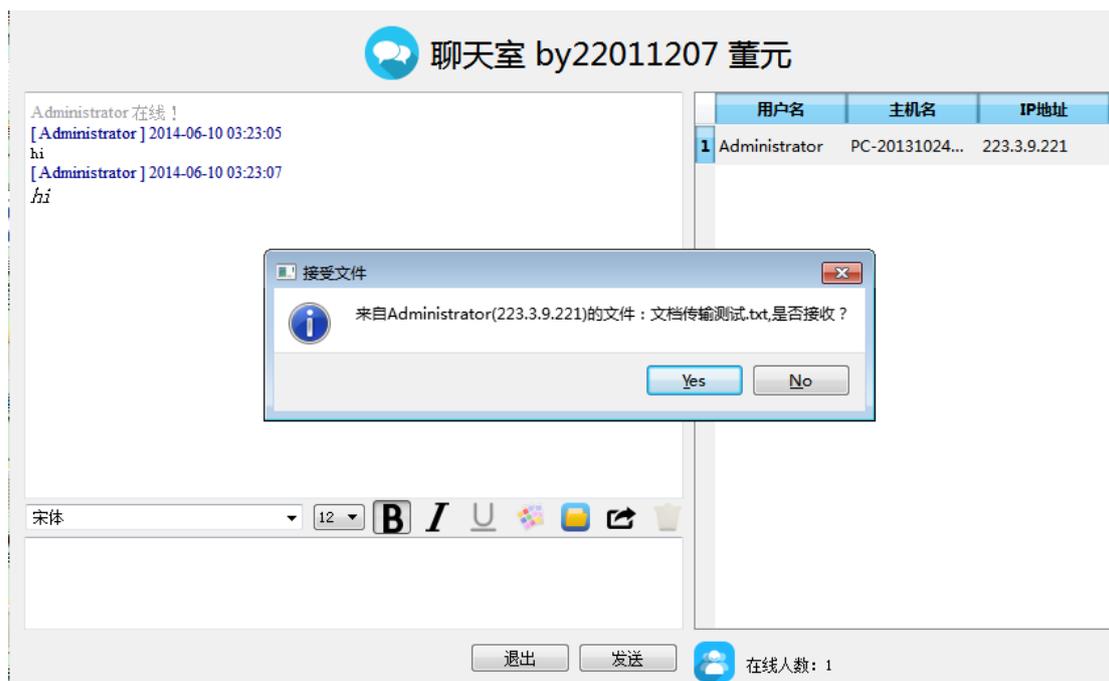


3.传输文件

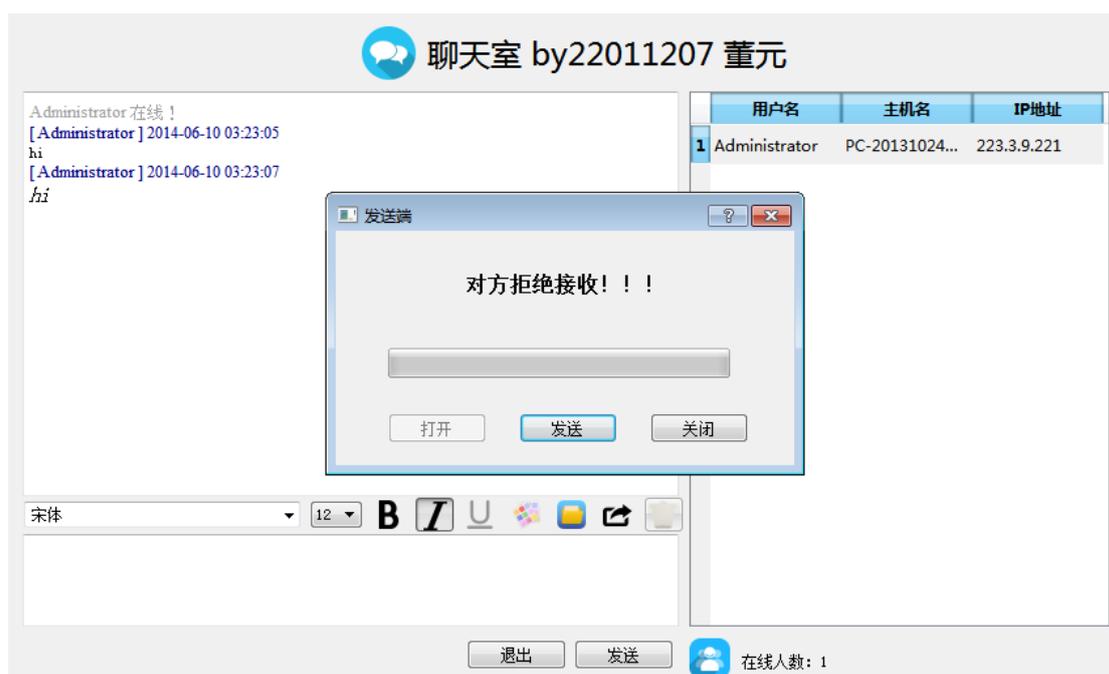
点击工具栏上的文档图标，可以选择要传输的文档。作为测试，选择传输“文档传输测试.txt”文件如下。



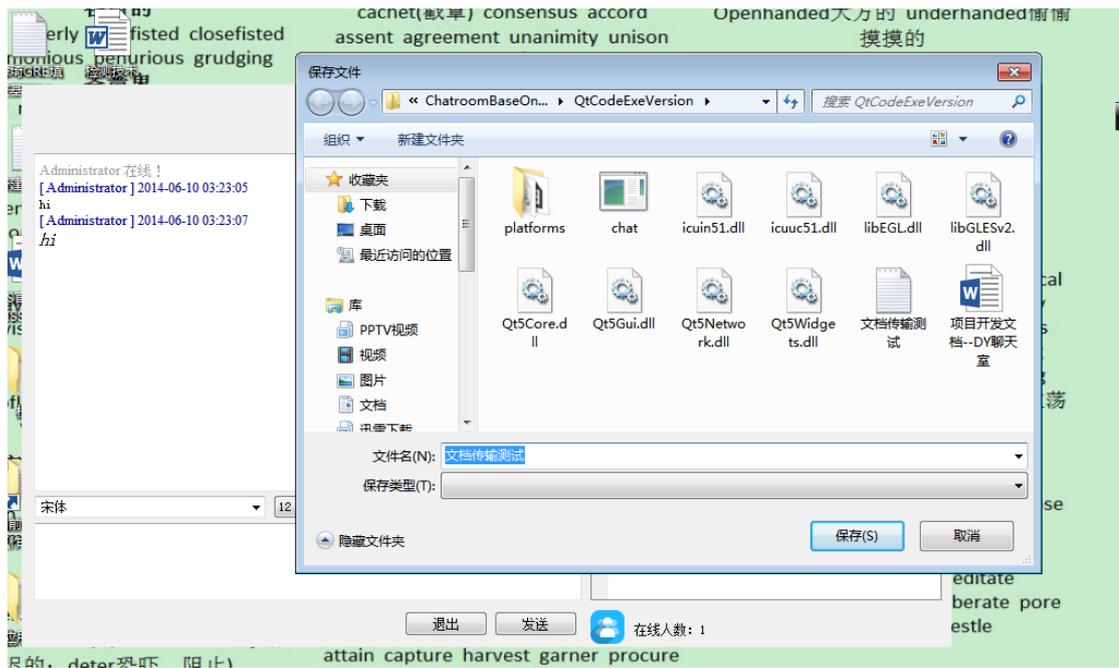
点击发送后，接受方会收到询问是否接受的提示窗口，由于测试时使用的是一台机器，故发送程序也会收到这样的询问。以下是接收方收到的询问信息：



若拒绝，则发送方会收到如下提示：



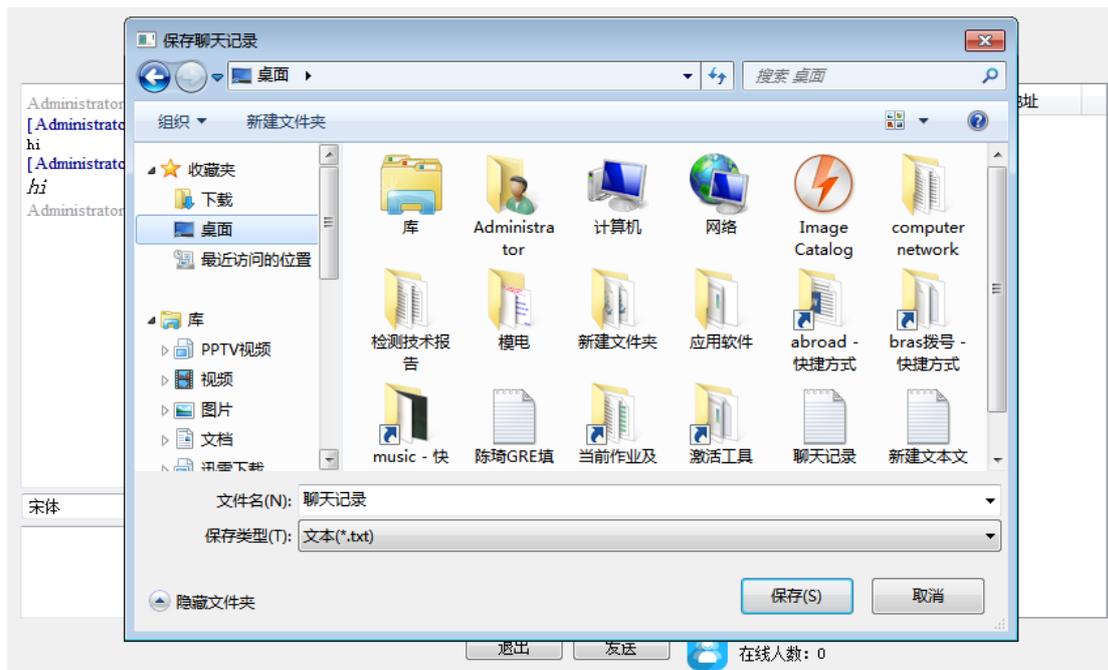
若接收，则可以选择相应的文件保存路径将其存储下来。



4. 聊天记录导出

点击工具栏最右方的聊天记录导出按钮可以将程序运行至今的聊天记录 txt 文件导出到自己想要的路径。





六、心得与体会

通过本次试验，我学习了如何实现双机通信编程方法和原理，并了解了在双机互联构建的局域网环境下如何进行数据交换方面知识。这个实验，很灵活的将分层的网络，数据流转，封装传输等过程展现在我们眼前，对今后的学习实践都有极大的帮助。

另外通过学习我们进一步知道了 TCP 是一种运输层协议，IP 是一种网络层协议 TCP/IP 协议族是一组不同的协议组合在一起构成的协议族。TCP/IP 应用程序工作模型是：使用 TCP/IP 协议的网络，其协议核心内容在层次结构的低三层，即网络接口层、IP 层和传输层，而这三层的功能一般是由操作系统的内核来实现的。TCP/IP 协议网络的典型应用方式，即客户/服务器模式。网络程序设计其实是使用系统提供的网络协议完成用户程序的功能，即在网络应用程序中使用网络协议提供的服务，而不是让用户去实现网络协议各层的功能。

最后，在编写聊天程序这类有界面的程序时，选择封装较好的库（如 Qt）可以更简单地实现较好的界面效果，在本次试验中，如果用 MFC 来达到相同的效果，要花费的经历比起用 Qt 要多很多。