



**课程设计报告**

**课程名称**  **计算机网络**

**学生学院 计算机学院**

**专业班级 网络工程四班**

**指导教师 梁路**

**学生姓名 高子淇**

**学 号 3119005369**

2021年 7 月 1日

1. 任务书
2. 程序开发的基础知识
3. 开发、运行环境说明
4. 设计思路
5. 程序流程图
6. 关键数据结构
7. 关键性的代码
8. 程序运行结果
9. 开发过程中遇到的问题及解决办法
10. 程序中待解决的问题及改进方向。
11. 任务书

|  |  |
| --- | --- |
| 设计题目 | **基于P2P的局域网即时通信系统** |
| 已知技术参数和设计要求 | 1．掌握P2P原理。  2．实现一个图形用户界面局域网内的消息系统。  3．功能：建立一个局域网内的简单的P2P消息系统，程序既是服务器又是客户，自拟服务器端口号并开启服务。  2.1用户注册及对等方列表的获取：对等方A启动后，用户设置自己的信息（用户名，所在组）；扫描网段中在线的对等方（服务器端口打开），向所有在线对等方的服务端口发送消息，接收方接收到消息后，把对等方A加入到自己的用户列表中，并发应答消息；对等方A把回应消息的其它对等方加入用户列表。双方交换的消息格式自己根据需要定义，至少包括用户名、IP地址等。  2.2发送消息和文件：用户在列表中选择用户，与用户建立TCP连接，发送聊天消息；若是发送文件，则封装在UDP报文中实现。另外，尽可能实现文件的发送进度条和断点续传的功能。  2.3用户主动或被动离线：设计合理的方式实现用户离线时，各对等方的用户列表的状态更新。  3．用户界面：界面上包括对等方列表；消息显示列表；消息输入框；文件传输进程显示及操作按钮或菜单。 |
| 设计内容与步骤 | 1．学习Socket和TCP的基本原理和通信机制；  2．功能设计和界面设计  3．服务器功能的设计和实现  4．客户功能的设计和实现  5．完成课程设计报告 |
| 设计工作计划与进度安排 | 1.P2P和 Socket原理 4小时  2.程序设计基础知识准备 10小时  3.功能设计和界面设计 6小时  4.软件系统的功能的实现 14小时  5.软件测试及完善 4小时  6.课程设计报告 4小时 |

1. 程序开发的基础知识

TCP/IP协议，UDP协议，基于java的socket开发，GUI的实现，UDP的datagramsocket的使用，TCP的socket和serversocket的使用。

1. 开发、运行环境说明

1.开发工具：Eclipse

2.开发环境： JDK11.0.2

2.运行环境：windows10

四、设计思路

首先我们要考虑如何实现群聊，可以通过UDP广播实现群聊，UDP广播会接收到ip地址，在实现一对一聊天的时候就可以使用指定ip地址建立TCP连接。这是通信的基本思路。

然后是设计交互界面GUI，我设计了一个消息窗口，一个发送窗口，一个在线列表，一个输入用户名的文本窗口。按钮我们设计了刷新谈话区和刷新在线列表的按钮，来实现清理群聊窗口和实现对用户主动离线和被动离线的状态更新。因为我的在线列表使用的是一个List进行存储的，所以状态更新我们设计的是直接清空整个列表，因为UDP广播是不断的，所以它会自动再次获取在线好友列表，那些离线的好友就接收不到了。这里我们有两种实现方法，一种是通过刷新好友列表进行重新更新，一种是定时清空好友列表进行更新。为了让用户有主动权，我们选择使用触发“刷新在线用户”按钮来实现列表的状态更新。

在一对一聊天设计中，我设计为双击在线用户开启TCP连接，发送消息对等方接收到通知后也双击打开对方的窗口，就可以接收到信息了。在这一界面也可以实现接收传输文件的功能。

图示

描述已自动生成五、程序流程图

六、关键数据结构

主类

public class PPfriend extends JFrame {

public Mywindow myw ;

public ReceiveToOne recevietoone;

SocketStatus socketStatus=new SocketStatus();

public Dialog qq=null;

String ips=null;

public String sendIp=null;

static public volatile String name=null;

//本地的ip

public String localip=null;

boolean send=false;

boolean isstart=false;

int num=0;

boolean isbegin=false;

String sendfilepath=null;

//接收文件的路径

String savefilepath=null;

String recivename=null;

File files;

//一对一发送的信息

String sendtooneMSG=null;

//一对一接收的消息

String receivetooneMSG=null;

//所有在线用户的ip

HashSet listhastset=new HashSet();

boolean packFrame = false;

List listset = new ArrayList();

ReceiveToOne receivetoone=new ReceiveToOne(this);

//RecevieSMG recevieSMG=new RecevieSMG(this);

SendSMG sendSMG=new SendSMG(this);

SendIp sendip=new SendIp(this);

//Register getname=new Register(qq,this);

boolean issendfile=false;

SendFile sendFile ;

Receive receive;

// SendThread sendThread

boolean flag=true;

// }

public String getName() {

return name;

}

public PPfriend() throws SocketException, UnknownHostException{

myw=new Mywindow();

this.inputips(this);

this.ips=getip();

this.actionPerformed(this);

this.refresh(this);

if(flag){

//发送自己的ip

//接收所有在线用户的ip

ReceiveIp receiveIp=new ReceiveIp(this);

receiveIp.start();

}

receivetoone=new ReceiveToOne(this);

receivetoone.start();

flag=false;

if (packFrame) {

//调整此窗口的大小，以适合其子组件的首选大小和布局

myw.pack();

} else {

//验证此容器及其所有子组件。

myw.validate();

}

// Center the window

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Dimension frameSize = myw.getSize();

if (frameSize.height > screenSize.height) {

frameSize.height = screenSize.height;

}

if (frameSize.width > screenSize.width) {

frameSize.width = screenSize.width;

}

myw.setLocation((screenSize.width - frameSize.width) / 2,

(screenSize.height - frameSize.height) / 2);

myw.setVisible(true);

}

MywindowGUI数据结构：

class Apanel extends Panel{

TextArea chat;

APanel2 aPanel2;

public Apanel(){

//创建个只读文本

chat=new TextArea(40,43);

chat.setEditable(false);

aPanel2=new APanel2();

//布置容器的边框布局

setLayout(new BorderLayout());

add("West",chat);

add("East",aPanel2);

}

}

class APanel2 extends Panel{

java.awt.List list;

APanel3 aPanel3;

APanel2(){

//List 组件为用户提供了一个可滚动的单选文本项列表,

list=new List(25,false);

aPanel3=new APanel3();

setLayout(new BorderLayout());

add("Center",list);

add("North",aPanel3);

add("East",new Label());

add("South",new Label("双击一个人的昵称可以进行交谈"));

}

}

class APanel3 extends Panel{

JTextArea msg;

Button button1;

APanel3(){

msg=new JTextArea(1,10);

button1=new Button("提交");

setLayout(new FlowLayout(FlowLayout.LEFT));

add(new JLabel("输入用户名"));

add(msg);

add(button1);

}

}

class BPanel extends Panel{

JTextArea msg;

Button button1,button2,button3,button4,button5;

public BPanel(){

msg=new JTextArea(5,30);

button1=new Button("发送");

button2=new Button("刷新谈话区");

button3=new Button("刷新在线用户");

Dimension preferredSize = new Dimension(100,80);

button1.setPreferredSize(preferredSize );

// button2.addActionListener(new Myw\_reflash\_msg\_adapter());

setLayout(new FlowLayout(FlowLayout.LEFT));

add(new JLabel("<html>发<br>送<br>的<br>话<html>"));

add(msg);

add(button1);

add(button2);

add(button3);

}

}

public class Mywindow extends JFrame{

//PPfriend friend=new PPfriend();

Label person;

Apanel pa;

BPanel pb;

public Mywindow(){

super();

person=new Label("聊天记录");

pa=new Apanel();

pb=new BPanel();

setLayout(new BorderLayout());

setBackground(Color.YELLOW);

setSize(650, 550);

setLocation(550, 200);

setResizable(true);

add("North",person);

add("Center",pa);

add("South",pb);

ImageIcon imageIcon=new ImageIcon("./src/1.png");

this.setIconImage(imageIcon.getImage());

//此类存在的目的是方便创建侦听器对象

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e){

System.exit(1);

}

});

this.setTitle("p2p");

setVisible(true);

}

}

class Myw\_send\_adapter implements ActionListener {

SocketStatus status=new SocketStatus();

// @Override

public void actionPerformed(ActionEvent e) {

System.out.println("fasonfaoononon.......");

status.setStatus(SocketStatus.RECIVE\_ALL\_MSG);

}

}

class Myw\_reflash\_msg\_adapter implements ActionListener {

SocketStatus status=new SocketStatus();

// @Override

public void actionPerformed(ActionEvent e) {

status.status=SocketStatus.RECIVE\_ALL\_MSG;

}

}

class Myw\_reflash\_people\_adapter implements ActionListener {

SocketStatus status=new SocketStatus();

// @Override

public void actionPerformed(ActionEvent e) {

status.status=SocketStatus.RECIVE\_IP;

}

}

七、关键性的代码

UDP广播ip：

public class SendIp implements Runnable{

String ip;

PPfriend friend;

public SendIp( PPfriend friend) {

super();

this.friend = friend;

}

public String getFriendName() {

return this.friend.name;

}

public void run(){

try {

ip = InetAddress.getLocalHost().getHostAddress();

friend.localip=ip;

ip=SocketStatus.RECIVE\_IP+":"+ip+":"+this.friend.name;

System.out.println(ip);

} catch (UnknownHostException e) {

// name=SocketStatus.RECIVE\_NAME+":"+friend.name+"";

// TODO Auto-generated catch block

e.printStackTrace();

}

while(true){

try {

//构造数据报包，用来将长度为 length 偏移量为 offset 的包发送到指定主机上的指定端口号

DatagramPacket packet=new DatagramPacket(ip.getBytes(), ip.getBytes().length,InetAddress.getByName(friend.ips) ,3333);

// DatagramPacket packet2=new DatagramPacket(name.getBytes(), name.getBytes().length,InetAddress.getByName(friend.ips) ,3333);

DatagramSocket socket=new DatagramSocket();

socket.send(packet);

//socket.send(packet2);

socket.close();

Thread.sleep(2000);

// System.out.println("send...........");

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

UDP接收端口用户ip：

ReceiveIp.java:

public class ReceiveIp extends Thread{

public ReceiveIp(PPfriend friend) {

super();

this.friend = friend;

}

PPfriend friend;

SocketStatus socketStatus=new SocketStatus();

// Vector vectorip=new Vector();

String msg;

String sendip="";

boolean socket=false;

public void run(){

byte[] inbuf=new byte[50];//某认缓冲的大小

//此类表示用来发送和接收数据报包的套接字

DatagramSocket socket;

try {

socket = new DatagramSocket(3333);

while(true){

try {

//此类表示数据报包。 构造 DatagramPacket，用来接收长度为 length 的数据包

DatagramPacket packet=new DatagramPacket(inbuf, inbuf.length);

synchronized (socket) {

try {

// System.out.println("11111111111111111");

//从此套接字接收数据报包

socket.receive(packet);

// System.out.println("2222222222");

} catch (Exception e) {

e.printStackTrace();

}

}

//接受数据

msg=new String(packet.getData());

for (int i = 0; i < inbuf.length; i++) {

inbuf[i]=(byte)0;

}

sendip=packet.getAddress().toString();

sendip=sendip.substring(1);

// System.out.println("sendip"+sendip);

System.out.println("msg"+msg);

String[] status = msg.split(":");

switch (Integer.parseInt(status[0])) {

case 0x01:

String result = msg;

System.out.println("xxxx"+result);

result=result.substring(2);

if(friend.listset.indexOf(result)==-1){

friend.listset.add(result);

friend.myw.pa.aPanel2.list.add(result);

}//else if(result[x].split(":")[0]!=friend.listset.get(friend.listset.indexOf(result[x])).split(":")[0]) {

break;

case 0x02:

if(msg!=null){

msg=status[1];

String[] result2 = status[2].split("\\s");

System.out.println(result2[0]+sendip+"说"+msg);

//String name = status[1].substring(1);

friend.myw.pa.chat.append("用户："+result2[0]+" "+sendip+"说"+":"+msg);

friend.myw.pa.chat.append("\n");

// ac.play();

}

break;

default:

break;

}

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

} catch (Exception e1) {

e1.printStackTrace();

}

}

}

TCP客户端：

Sendtoone.java:

public class SendToOne extends Thread {

DatagramSocket datagramSocket;

InetAddress ia;

PPfriend friend;

//一对一发送的信息

String sendtooneMSG=null;

//用来判断是发送的信息还是建立连接。。true 代表一对一的消息

boolean isMSG;

public SendToOne(PPfriend friend,boolean isMSG,String sendtooneMSG){

this.friend=friend;

this.isMSG=isMSG;

this.sendtooneMSG=sendtooneMSG;

try {

Thread.sleep(1000);

//DatagramPacket packet=new DatagramPacket(inbuf, inbuf.length);

//this.friend.sendip;

String[] ips=friend.sendIp.split(":");

System.out.println(ips[0]);

ia=InetAddress.getByName(ips[0]);

Socket socket=new Socket(ia,9999);

OutputStream out=socket.getOutputStream();

System.out.println(this.sendtooneMSG);

System.out.println("111111"+friend.sendIp);

byte[] bs=new byte[1024];

if(isMSG && sendtooneMSG!=null){

sendtooneMSG=SocketStatus.RECIVE\_ONE\_TO\_ONE\_MSG+":"+friend.sendIp.split("\\s")[0]+":"+this.sendtooneMSG;

bs=sendtooneMSG.getBytes();

}else{

System.out.println("test link....");

String msg=SocketStatus.RECIVE\_TEST\_MSG+":";

System.out.println("sendtoneMSG"+msg);

bs=msg.getBytes();

}

//构造数据报包，用来将长度为 length 的包发送到指定主机上的指定端口号。

out.write(bs);

out.flush();

out.close();

//从此套接字发送数据报包。

//socket.send(reques);

} catch (Exception e) {

e.printStackTrace();

}

}

public void run(){

}

}

TCP服务端：

Receivetoone.java:

public class ReceiveToOne extends Thread {

DatagramSocket datagramSocket;

public String getip=null;

PPfriend friend;

int num=0;

String msg;

public ReceiveToOne(PPfriend friend){

super();

this.friend=friend;

}

public void run(){

try{

byte[] inbuf=new byte[256];

ServerSocket server=new ServerSocket(9999);

//创建数据报套接字并将其绑定到本地主机上的指定端口。用UDP可以接收广播

//Dialog qq=new Dialog("一对一聊天");

while(true) {

Socket socket=server.accept();

System.out.println("recive to one......");

//数据报包用来实现无连接包投递服务。 构造 DatagramPacket，用来接收长度为 length 的数据包

InputStream in=socket.getInputStream();

byte [] bytes = new byte[1024];

int count = in.read(bytes);

//接受数据

String content = new String(bytes,0,count);

//返回某台机器的 IP 地址，此数据报将要发往该机器或者是从该机器接收到的。

// if(msg!=null){

getip=socket.getInetAddress().getHostAddress();

// }

System.out.println(content);

if(getip!=null && content!=null){

//把缓冲区清空

for (int i = 0; i < inbuf.length; i++) {

inbuf[i]=(byte)0;

}

System.out.println("msg........"+content);

String[] status = content.split(":");

System.out.println(status[0]);

switch (Integer.parseInt(status[0])) {

case 0x03:

friend.receivetooneMSG=status[3];

JOptionPane.showMessageDialog(null, getip+"发消息给您，请双击它的ip以便接收",null,JOptionPane.OK\_OPTION);

if(num==1) {

friend.qq.chat.append("用户："+getip+"说"+":"+status[3]+"\n");

}

num=1;

break;

case 0x04:

// if(friend.qq!=null){

// JOptionPane.showMessageDialog(null, getip+"发文件给您，请按接收按钮接收",null,JOptionPane.OK\_OPTION);

// friend.qq.refile.setEnabled(true);

// }else{

JOptionPane.showMessageDialog(null, getip+"发文件给您，请双击它的ip以便接收",null,JOptionPane.OK\_OPTION);

// }

System.out.println("recive....sucess.....");

break;

default:

break;

}

}

in.close();

}

}

catch (Exception e) {

e.printStackTrace();

}

}

}

发送文件：

Sendfile.java：

import javax.swing.JOptionPane;

public class SendFile extends Thread {

public static final int PORT=4444;

PPfriend friend;

private String filepath=null;

SendToOne sendToOne;

public SendFile(PPfriend friend,String filepath){

this.filepath=filepath;

this.friend=friend;

}

public void doserver(){

int length;

sendToOne=new SendToOne(friend,false,null);

File file=new File(filepath);

Socket socket;

if(filepath!=null){

try{

//通过打开一个到实际文件的连接来创建一个 FileInputStream，该文件通过文件系统中的 File 对象 file 指定

System.out.println("filepath"+filepath);

FileInputStream fos=new FileInputStream(file);

// 创建绑定到特定端口的服务器套接字。

System.out.println("sendfile here........");

ServerSocket s=new ServerSocket(PORT);

System.out.println("sendfile here11111111111");

socket = s.accept();

try {

try{

InputStream netin=socket.getInputStream();

System.out.println("sendfile here.11111111111111111111.");

InputStream in=new DataInputStream(new BufferedInputStream(netin));

OutputStream netout=socket.getOutputStream();

OutputStream doc=new DataOutputStream(new BufferedOutputStream(netout));

// 返回此抽象路径名的绝对路径名形式。

file.getAbsoluteFile();

byte[] buf=new byte[2048];

length=fos.read(buf);

while(length!=(-1)){//是否讀完

doc.write(buf, 0, length);//把問價數據送到緩衝區

doc.flush();//刷新緩衝區把數據寫網客戶端

length=fos.read(buf);//繼續從文件讀取數據

}

fos.close();

doc.close();

}finally{

JOptionPane.showMessageDialog(null, "文件传输成功", null, JOptionPane.OK\_OPTION);

socket.close();}

}finally{

s.close();

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

public void run() {

this.doserver();

}

}

接收文件：

Receive.java：

public class Receive extends Thread {

private String filepath=null;

PPfriend friend=null;

InetAddress addr;

public Receive(PPfriend friend,String savepath){

this.friend=friend;

filepath=savepath;

}

public void doclient(){

int length;

File file=new File(filepath);

//创建一个向指定 File 对象表示的文件中写入数据的文件输出流。

FileOutputStream fileOutputStream;

Socket socket;

try {

fileOutputStream = new FileOutputStream(file);

if(friend.receivetoone.getip!=null){

try {

//在给定主机名的情况下确定主机的 IP 地址

addr=InetAddress.getByName(friend.receivetoone.getip);

try {

//此类实现客户端套接字创建一个流套接字并将其连接到指定 IP 地址的指定端口号

socket = new Socket(addr,4444);

try{

//向文本输出流打印对象的格式化表示形式,刷新输出缓冲区

//BufferedWriter:将文本写入字符输出流，缓冲各个字符，从而提供单个字符、数组和字符串的高效写入

PrintWriter out=new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())),true);

InputStream netin=socket.getInputStream();

InputStream in=new DataInputStream(new BufferedInputStream(netin));

OutputStream netout=socket.getOutputStream();

OutputStream doc=new DataOutputStream(netout);

byte[] buf=new byte[2048];

//从输入流中读取一定数量的字节，并将其存储在缓冲区数组 buf中。

length=in.read(buf);

while(length!=(-1)){//是否读完所有的数据

//将指定 byte 数组中从偏移量 off 开始的 len 个字节写入此文件输出流

fileOutputStream.write(buf, 0, length);

length=in.read(buf);

}

in.close();

fileOutputStream.close();

}finally{

JOptionPane.showMessageDialog(null, "文件传输成功！",null,JOptionPane.OK\_OPTION);

socket.close();

}

} catch (IOException e) {

e.printStackTrace();

}

} catch (UnknownHostException e) {

e.printStackTrace();

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

public void run(){

this.doclient();

}

}

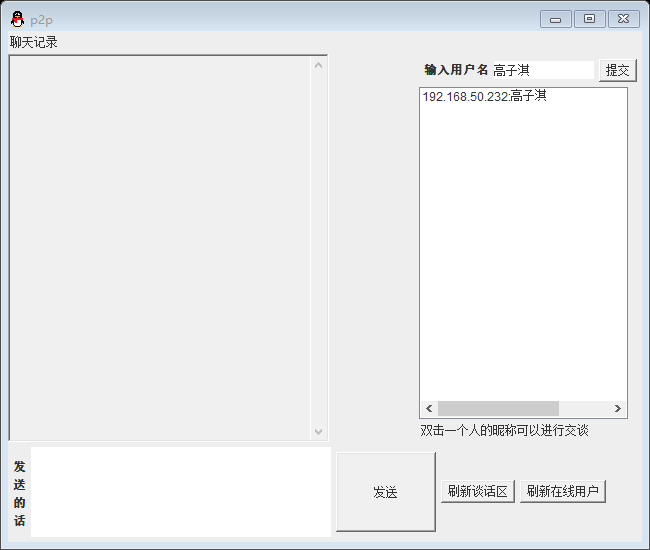
1. 程序运行结果

运行PPfriend.java主类(请一定使用局域网的两台主机以上进行测试)：

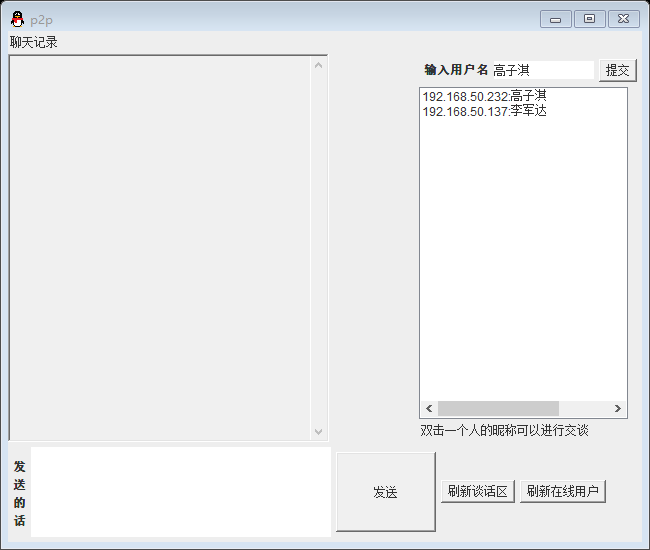
（如果出现中文乱码请将编码改成UTF-8）



输入用户名点击提交：



接收到对等方在线用户列表：



聊天框信息：

图形用户界面, 应用程序, Word

描述已自动生成

一对一聊天：

图形用户界面

描述已自动生成

图形用户界面

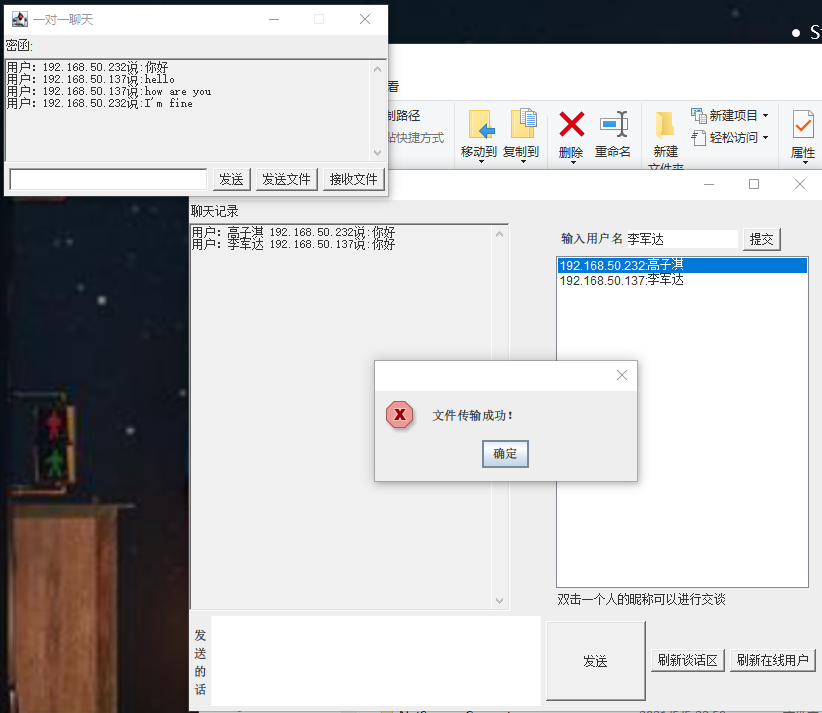
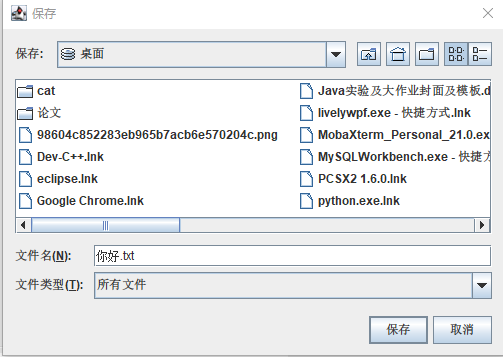
描述已自动生成

发送文件：

图形用户界面, 应用程序

描述已自动生成

图形用户界面

描述已自动生成

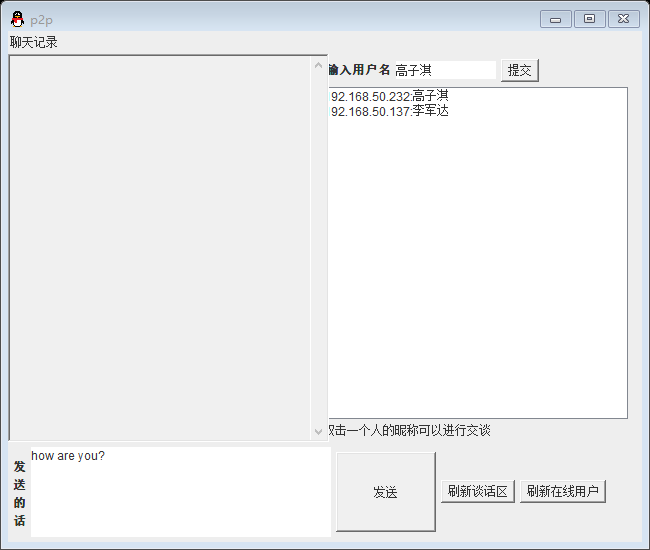
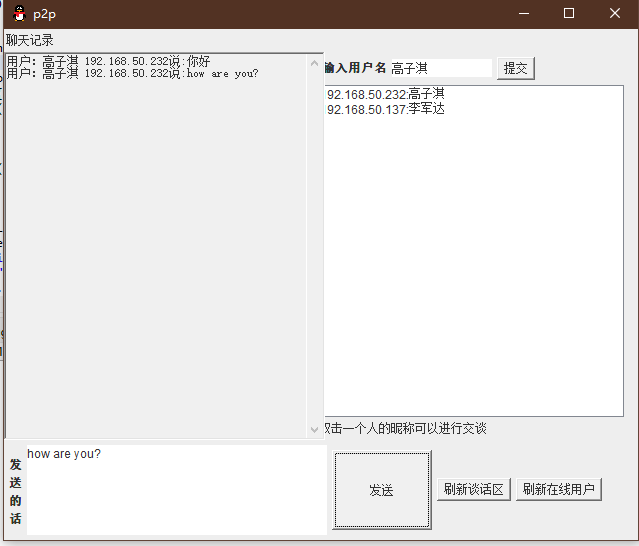
刷新在线用户：

对方已下线，列表只保存在线用户

图形用户界面, 应用程序, Word

描述已自动生成

刷新谈话区：



九、开发过程中遇到的问题及解决办法

先说界面设计，界面设计我的思路就是类似qq微信那样的聊天框弹出，比较麻烦的是每个框的尺寸设计，还有按钮的排列，发送、刷新谈话区和刷新在线用户三个按键排在一起很长，后来把聊天记录框加宽了一点实现的。在用户列表最上面我们加了用户名的输入框和提交按钮。

之后是p2p的通信，我设计一个sendip类一个receiveip类，这里出现一个问题，我们广播ip的时候想要把用户名带上，一直是想怎么让输入在主线程的用户名传到子线程里面去发送，最后我换了个思路，决定只有你输入完提交完用户名之后，再开启sendip这个线程进行广播。这里建立的连接全部用的是java的UDP Datagramsocket实现的。端口号我们设置的是3333。

然后是一对一聊天，一对一这里的问题是最多的，因为首先你要在UDP之外实现一个TCP连接。我们的TCP的serversocket是从打开应用就开始监听了，然后只有在你打开单一用户聊天框，输入聊天消息，按下发送才会开启socket发送，这时候对方的监听就收到了。一开始有个问题是，只能进行一次接受和发送，之后对方都收不到了。我最后发现问题在于，当对方接收到消息时，是没有建立一对一聊天这个窗口的，也就是接收到的信息是没办法打印到聊天框列表的，用户必须自己双击打开对方的聊天窗才能接收到，这个时候就会产生报错，因为没有实例化对象就append的了。最后我们想了两种解决方法，一般来说一个用户接收到消息，都会马上打开对方的聊天框，这时候我们就可以设置一个sleep（）等待用户打开，这样就不会出现问题。还有一种方法是，在线程里的append在第一次接收的时候不打印，我们在打开聊天框这个触发事件里进行第一次的打印，这样就不会报错了，之后的接收打印才走线程里的append。

然后是在线用户的刷新，这里呢，因为在线用户的udp应该是不停止广播的，那很简单，我们只需要知道这个端口还有那些在广播就行了。我们实现有两种方法，一种是设置一个线程，一段时间就把当前用户列表清空，重新从广播端口接收所有在线ip。还有一种是让用户主动去刷新列表，也就是刷新在线用户，点击一下就会清空当前列表重新接收在线ip。我们最后选择的是第二种方法。

十、程序中待解决的问题及改进方向。

1.有一个问题是，由于写了条件语句可以完成一对一的聊天了，但是发起对话的那个人是接受不到第一次发回来的消息的，但之后的都可以，这个问题的话，我的初步思路是将第一次发送和接收全部用空信息取代，就是把这个第一次给浪费掉，正式发送实际上就是第二次发送接收了。类似三次握手的实现方法吧。

2.还有一个分组问题，这个其实好写，只需要把发送ip带上所在组然后receiveip的时候解析组打印到指定组的聊天框就行了，但我觉得没必要分组去实现，还不如再打开个聊天框然后换个端口就是一个分组了。