摘　要

随着互联网技术的蓬勃发展，网络聊天逐渐常态化。它使得联系亲朋好友不再有空间与时间的限制。本设计拟在实现远程通信，为好友或是家庭成员之间实现实时聊功能。

本设计为一款基于Windows平台，使用Java语言进行开发的小程序。它以NetBeans为开发编程工具，并使用JavaFx中的Scene Builder进行搭建场景及实现可视化界面。图形用户界面使得用户易于操作、可读性高。

本设计功能如下：一、两客户端用户私人聊天。二、两客户端用户间文件的传输。三、多人群聊。

本设计分为两部分：服务器与客户端。除文件传输外，客户端的其他交互均通过与客户端的交互实现。以示服务器监听之效。

关键词：小程序；聊天；网络社交；网络通信；UDP 传输

3、 系统分析与设计

3.1 系统界面设计及功能实现

3.1.1 服务器与客户端的启动

图 1　服务器界面

首先，是服务器界面（图 1），操作方法如下：

一、运行文件，进入客户端系统，填入服务器用户设定的端口号，并点击启动。（本设计以服务器端口号 6000 为例），信息窗口显示服务器处于接收信息状态。

图 2 服务器界面点击“启动”后

图 3　　客户端界面

其次是客户端界面（图 3），操作方法如下：

二、运行文件，进入客户端系统，在“服务器 IP”填入服务器所在设备网络

属性中 IPv4 的 IP 地址；在“服务器端口”填入之前服务器设定的端口号；在“本地端口”填入客户端用户自定义的端口号并点击“启动”按钮。

三、此时，服务器与客户端均进入启动状态，等待接收消息，如下图 4、5 所示。

提示：务必先运行服务器，否则会导致客户端激活信息无法被服务器接收。

图 4 服务器激活

图 5 客户端激活

3.1.2 服务器与两个客户端的交互——私人聊天（功能 1）

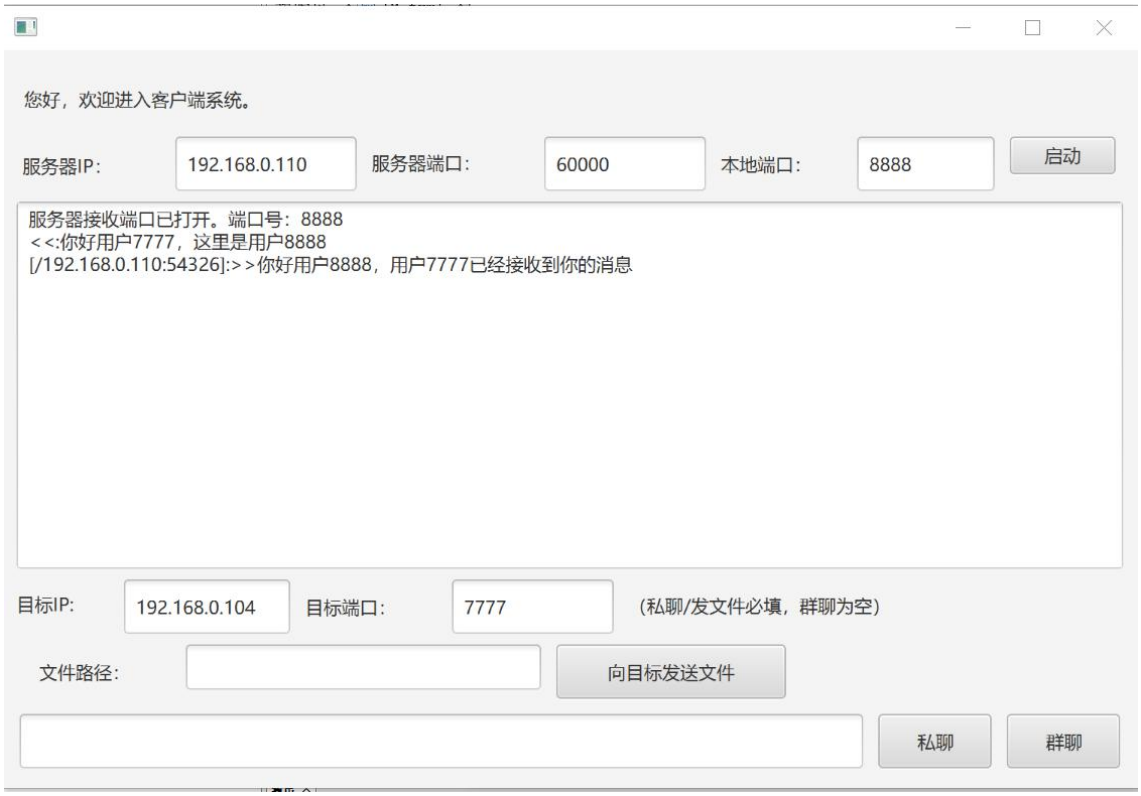


图 6 客户端用户 8888 向客户端用户 7777 私人发送消息

私人聊天操作简单易行。

首先，需确认双方已开启客户端并激活。

其次，在最下方输入栏内输入信息；输入目标客户端的 IP 及端口（方法及定义方式同激活启动时）；最后点击“私聊”按钮，即完成向目标客户端的私人信息发送。

下面的图 7 展示了用户 7777 接收到私人信息并回复消息的过程。

图 8 则展示了在两用户消息交流时，服务器的中转站功能，好比“卫星”的功能，接收用户 A 的信息通过目标 IP 和端口转向用户 B。

服务器信息栏会显示从用户 A 处接收到的并在成功向用户 B 转发后再次在信息栏中提示。


您好，欢迎进入客户端系统。
服务器IP:
192.168.0.110
服务器端口:
60000
本地端口:
7777
启动
服务器接收端口已打开。端口号：7777
[/192.168.0.110:51273]:>>你好用户7777，这里是用户8888
<<:你好用户8888，用户7777已经接收到你的消息
目标IP:
192.168.0.104
目标端口:
8888
(私聊/发文件必填，群聊为空)
文件路径:
向目标发送文件
私聊
群聊

图 7


您好，欢迎进入服务器系统。
客户端IP:
客户端端口:
服务器端口:
60000
启动
服务器接收端口已打开。端口号：60000
[/192.168.0.104:55345]:>>客户端请求激活。
激活成功！客户端IP: /192.168.0.104 端口：7777已保存地址到服务器。
[/192.168.0.104:55644]:>>客户端请求激活。
激活成功！客户端IP: /192.168.0.104 端口：8888已保存地址到服务器。
[/192.168.0.104:57471]:>>4#你好用户7777，这里是用户8888
服务器已经向目标发送：4#你好用户7777，这里是用户8888
[/192.168.0.104:57472]:>>4#你好用户8888，用户7777已经接收到你的消息
服务器已经向目标发送：4#你好用户8888，用户7777已经接收到你的消息
发送

图 8

3.1.3 服务器与多个客户端的交互——多人聊天（功能 2）

说明：群聊功能无需填写目标 IP 与端口，只需在激活后在最下方信息栏填入信息，

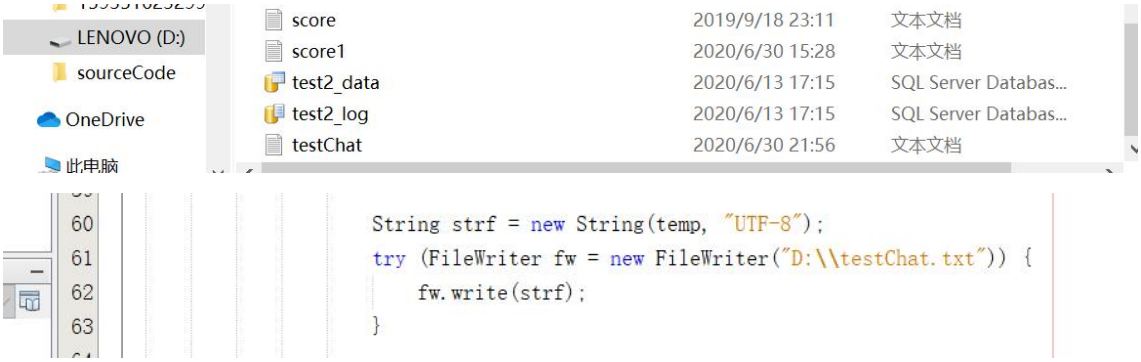
再点击群聊按钮即可。

上图分别模拟和展示了三个客户端在线时群聊的过程的服务器在接收到信息后对在线用户的转发过程。

3.1.4 两个客户端的交互——私人传输文件（功能 3）

上图中展示了两个客户端之间（不经过服务器）的文件传输过程，暂时只支持文本文件的传输。

注意：本系统默认将接收到的文件以默认的名称存在下图所示的位置，如果要改变，需修改最后一图的代码地址。



4、 系统实现

4.1 系统功能 1 的实现

私人聊天的实现主要依赖于标识符 flag：1#的使用，整体思路如下：

1. 客户端将目标 IP 与目标端口即信息内容打包放入一个数据报内。客户端实现代码如下图。

```
public class DataHandler {

    public static DatagramPacket toPacket(String sendMsg, InetAddress remoteAddress, int remotePort, InetAddress destAddress, int destPort) {
      //标识符1#，告诉服务器为私包，里面还整合了IP及端口数据。
        String buffer1, buffer2, buffer3, buffer4;
        buffer1 = "1";
        buffer2 = destAddress + "#";
        buffer3 = destPort + "#";
        buffer4 = buffer1 + buffer2 + buffer3 + sendMsg;    //标识符+IP+端口+信息内容
        byte[] buffer = buffer4.getBytes();

        return new DatagramPacket(buffer, buffer.length, remoteAddress, remotePort);
    }
```

2. 在此处一并介绍服务器接收到数据报并进行解析的思路及代码实现，后文就只介绍客户端的打包方式，不至行文过于繁琐。

Flag：1# （客户端打包专用）通过标识将客户端私人发送的数据报解析并转发给目的客户端。

```
switch (str.charAt(0)) {
    case '1':
        //将客户端私发的报解析并转发给目的客户端
        try {
            socket2 = new DatagramSocket(null);
        } catch (SocketException ex) {
            Logger.getLogger(ReceiverThread.class.getName()).log(Level.SEVERE, null, ex);
        }   try {
            socket2.send(DataHandler.destIncluded(temp));
            socket2.close();

        } catch (IOException ex) {
            Logger.getLogger(ReceiverThread.class.getName()).log(Level.SEVERE, null, ex);
        }   controller.appendMessage("服务器已经向目标发送：" + receivedMessage);
        break;
```

Flag：2# （客户端打包专用）用来标识使客户端“启动”按钮点击时所发送的数据报，此数据报内部无信息内容，仅为客户端本地端口号。服务器得到后保存下来，以供多人聊天时服务使用。

```
case '2':
    //将启动时客户端的ip和端口存入客户端的list
    ipList.add(dp.getAddress());
    portList.add(DataHandler.portIncluded(temp));
    controller.appendMessage("激活成功！客户端IP：" +dp.getAddress()+"  端口："+DataHandler.portIncluded(temp)+"已保存地址到服务器。");
    break;

case'3':
    //为群发
    for (int i = 0; i <= ipList.size()-1; i++) {
        try {
            socket2 = new DatagramSocket(null);
        } catch (SocketException ex) {
            Logger.getLogger(ReceiverThread.class.getName()).log(Level.SEVERE, null, ex);
        }
        try {
            socket2.send(DataHandler.groupIncluded(temp,ipList.get(i),portList.get(i)));
        } catch (IOException ex) {
            Logger.getLogger(ReceiverThread.class.getName()).log(Level.SEVERE, null, ex);
        }
        socket2.close();
        controller.appendMessage("服务器已经群发：" + receivedMessage);
    }   break;
```

Flag：3#（客户端打包专用） 用来标识“群发”功能，服务器获取并解析数据报后，将内容信息根据已激活保存的 IP、端口号逐个转发，以达到群发功能。

Flag：4# （服务器打包专用）用来标识非文件信息。客户端得到报并解析得到此标识后，将信息内容显示在消息栏。

```
public static String notFileData(String temp){
    //由客户端需要区分数据报是普通报或文件报，由于文件不方便在前加falg标识符进行处理，故将所有给客户端发送的普通报中加flag标识符4#，客户端有对应的toString处理方式
    //会导致一个弊端，服务器接受的客户消息前多出4#这个flag标识符，暂无方法改善
    return "4"+"#"+temp;
}
}
```

无 Flag：由于文件传输涉及乱码等格式问题，不宜使用标识符，故无标识符的情况默认设置为接收到的数据报为文件的情况。整体代码实现如下图。

```
            default:
                //无flag则表示接收的是文件

                String strf = new String(temp,"UTF-8");
                FileWriter fw=new FileWriter("E:\\test.txt");
                fw.write(strf);
                fw.close();

                controller.appendMessage("接收成功。");
        }
```

4.2 系统功能 2 的实现

多人聊天的客户端打包方式如下图所示。整体思路为，客户端向服务器提出群发要求，服务器识别后将消息逐条发送。

```
    public static DatagramPacket toPacket(String sendMsg, InetAddress remoteAddress, int remotePort) {
     //标识符3#，告诉服务器此报需群发
        String str;
        str = "3" + "#" + sendMsg;
        byte[] buffer = str.getBytes();

        return new DatagramPacket(buffer, buffer.length, remoteAddress, remotePort);
    }
```

4.3 系统功能 3 的实现

客户端两用户或客户端与服务器之间可以实现文件传输，由于功能限制，开发时间较短，只支持以 UTF-8 格式保存的文本文件的传输（ansi 格式中文字符会乱码），已通过测试，这种方式不会造成乱码。

实现收发功能的代码如下两张图所示。

```
    @FXML
    void OnPSendClickedFile(ActionEvent event) throws FileNotFoundException, IOException {
    //客户端私发文件
        char[] c = new char[500];
        try (FileReader fr = new FileReader(getPath());) {
            int num = fr.read(c);
            String str = new String(c, 0, num);

            byte[] file = str.getBytes("UTF-8");
            DatagramPacket dpf = new DatagramPacket(file, file.length, getDestIP(), getDestPort());
            socket = new DatagramSocket(null);
            socket.send(dpf);
            socket.close();
            appendMessage("已经向目标发送文件。");

        }

    }
```

```
    public static String toString(DatagramPacket p) throws UnknownHostException {
        //客户端送来的非文件报（即普通报）都会加flag标识符4#，故需要处理
        byte[] temp = p.getData();
        String str = new String(temp);

        if (str.charAt(0) == '4') {
            StringTokenizer token = new StringTokenizer(str, "#");

            String flag = token.nextToken();
            String msg = token.nextToken();
            return msg;
        } else {
            return "已收到文件";
        }

    }

}
```

测试文件将会一起放在压缩文件中，读者可进行测试。

4.4 系统功能实现总结

在上述的介绍中，本程序整体思路与运行方式已阐明。还有其他细节由于篇幅所限，不及叙述，源代码内有详细的备注，可解读者疑惑。之间关联，运行便知，便不再赘述。

5、 总结

5. 1 系统存在的不足与心得

系统不足之处甚多，挑几处急需优化的阐述。

其一：暂时还没有做到对发出聊天讯息的用户的命名功能。这造成的影响是，在聊天中（尤其是群聊时）无法知晓究竟时哪位用户发出的讯息，必须由发出聊天者自行阐明身份。

其二：之前也与老师进行过交流，对于两人私人聊天是否应该经过服务器中转有过讨论。的确，客户端之间本就可以很轻易的发送消息，只要将服务器 IP 和

端口改成目标 IP 及端口即可，这样确实方便很多，也无需设置众多的标识符和众多的解析方法，因为将“目标 IP 地址+目标端口+信息内容”放在同一个报中转发并解析不是一件轻松的事情。笔者介于通过服务器的方法已经开发了大半，不愿放弃而去用更简单的方法，故才用此复杂的方式，如果早些与老师交流，定采取前面简单的方式，也不会在这方面花费如此时间精力。

其三：是标识符的设立。笔者一开始设立的初衷是为了标识不同信息的报，以达到采取特定的方法对待特定报的目的。现在看来，大概是运气较好，文件或是信息未有与标识符相同的情况存在，所以才没有误识。若本身消息的开头含有 1#或是 2#等此类与标识符相同的字符，应会造成误识别问题。出于笔者还没找到更好的标识办法，就先用此方法代替。

其四：是分隔符的问题。类似于第三点，笔者采用的是 Tokenizer 的方法，利用“#”进行分割，若消息中出现此字符，则此字符后面的消息会遗失。对此，还未找出较好的处理方式。

其五：是文件处理太过单调，只有 txt 文件，且格式必须为 UTF-8 的编码方式，这是由于笔者对于文件处理还不够熟练，尤其对于乱码一块处理不得当。之后后想进一步开发图片或音频文件的传输功能。

以上，便是不足之处，最后再来一谈笔者在开发这个简易聊天小程序的感受。

程序虽简单，却是实在耗费时间与精力。仅仅实现了三个功能，却耗费了整整五天来研究学习。由于还未学习计算机网络等相关知识，能达到此功能实属不易，无奈还是有无法解决的技术难题，能力有限，还望读者莫要太过苛求。

开发过程中遇到的难题很多，也与老师及学长有过交流。举几个例子，首先是两个客户端以通过服务器的方式实现私人聊天这个功能，耗时最久，达三天。问题主要出在转发功能的实现上，此问题解决后，获益良多，后两个功能亦迎刃而解。其次的难题是数组的清空，第一次误以为赋值给报便可达到清空数组的目的，后发现程序依旧实现不了，请教学长，经点拨反应过来本质是指针的改变，而非实际数组的清空。最后则是乱码问题，需采用 UTF-8 的编码方式才可解决，最终实现了文件的传输。

开发之路实属不易，但是经过这次的项目研究，对编程及网络技术有有了更深的理解，确有成长，当然也要感谢老师和学长的辛勤答疑。