**计算机网络**

**课程设计报告**

**课设题目： 基于单播的局域网聊天程序**

**专 业： 软件工程**

**班 级: XXXXXXX**

**学 号: XXXXXXXXX**

**姓 名： XXXXXX**

**成 绩：**

**完成日期: 2023.12.19- 2023.12.31**

**指导教师: XXXXX**

目录

[一、 课程设计题目与要求 1](#_Toc24955)

[二、 需求分析 1](#_Toc24354)

[三、 整体设计 2](#_Toc18025)

[1． 客户端整体设计 2](#_Toc7181)

[2． 服务器整体设计 3](#_Toc13354)

[四、 详细设计 4](#_Toc11883)

[1． 好友上下线功能 4](#_Toc3790)

[2． 黑名单 4](#_Toc31589)

[3． 隐身 5](#_Toc23246)

[5． 忙碌状态 6](#_Toc8634)

[五、 系统测试 8](#_Toc24431)

[1． 服务器 8](#_Toc26901)

[2． 客户端 9](#_Toc21827)

[六、 源代码（部分） 9](#_Toc16650)

[1． 创建连接 9](#_Toc18862)

[2． Combobox（状态选择） 11](#_Toc29959)

[3． 右键菜单（拉黑） 12](#_Toc26522)

[七、 总结 13](#_Toc9403)

[八、 参考文献 14](#_Toc5819)

1. **课程设计题目与要求**

**题目**：基于单播的局域网聊天程序

**要求：**了解通信协议在计算机网络应用中的重要地位；设计一个完整应用层协议

学会如何根据具体的应用要求，设计出符合要求的应用层协议；掌握如何将设计好的协议清楚、详细地表示出来（即协议的描述方法）；掌握协议的具体实现方法应用可视化编程工具，实现协议所描述的功能，掌握网络应用的设计、开发过程。

**程序功能**

采用单播方式，实现局域网聊天

**基本功能**

登录：输入服务器IP地址，端口，用户名，密码（可选）

用户列表显示：显示在线用户

实时聊天：一对一聊天(注：不是客户端与服务器聊天)，广播

**扩展功能一**

好友上线提示

用户状态设置：在线、忙碌、隐身等

一对多聊天：同时给多个用户发送信息

**扩展功能二**

用户名唯一性检查

用户忙绿状态下自动回复信息

1. **需求分析**

**需求如下**：

1. 上线通知：在有用户上线时，显示在其他用户的界面上显示该用户的上线信息。
2. 群聊功能：可以发送对所有人的消息，服务器会负责向所有人转发。
3. 私聊功能：可以指定消息接收人，服务器会负责向指定人转发。
4. 拉黑功能：可以指定不接收某人的消息，并告知对方已被我方拉黑。
5. 隐身模式：可以让服务器转发本人下线的消息，但是服务器依旧会转发所有人消息给隐身者。
6. 忙碌模式：发送忙碌提醒给全部人，在其他人依旧准备发送消息时，会自动回复提醒忙碌状态。
7. 自动回复：可以设置自动回复，在弹出窗口中填写自动回复语句。
8. **整体设计**
   1. ****客户端整体设计****

客户端是直接面对用户的窗口程序，应该做到简单直接，上手就可以操作。

下面是客户端的整体设计概述：

****登录界面****：只有用户名输入框和一个登录按钮，十分简洁，但是能 能完成登录功能。

****聊天窗口****：实时显示聊天消息，包括群聊和私聊，并且配备一个清除按 钮可以清空消息列表。

****用户列表****：显示当前在线的用户列表，支持选择特定用户进行聊天，拉 黑。

**输入框**：输入想要发送的话语，按下发送键即可直接发送。

**用户连接**：使用socket与服务器进行连接。

****状态设定****：可以设置自己当前的状态，比如在线，隐身，忙碌等，还可 以设置自动回复。

* 1. ****服务器整体设计****

服务器作为中心管理和控制单元，负责处理客户端请求、和转发消息。以下是服务器的整体设计：

****聊天窗口****：实时显示聊天消息，包括群聊和私聊，以功能执行情况显示。

****用户列表****：显示当前在线的用户列表。

****连接信息：**在这里可以设置最大人数和使用的端口信息。**

1. **详细设计**
   1. **好友上下线功能**

在用户的状态发生改变时，客户端都会发送特定消息给服务器，比如离线状态变为在线状态时，客户端会发送“LOGIN”开头的消息，并附上自己的id，这时候服务器会转发给所有在线用户，告诉所有人你已经上线。同样的，在你隐身或者下线的时候，客户端会在隐身或者下线前发送“LOGIN1”或者“DISCONNECTED”开头的消息，在服务器上这两个消息都会被转发为以“DISCONNECTED”开头的消息，让其他用户认为你已下线，区别就是“LOGIN1”也就是隐身状态时不会中断与服务器的连接，并还可以接收群聊消息。

* 1. **黑名单**

首先在拉黑的时候我们需要指定对象，并进行此操作，所以我想到可以把这个功能放到好友列表上，如图1所示。

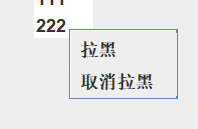


图 4-1 拉黑按键位置

按下这个按钮后，首先会私聊提示对方已经被我们拉黑，然后把对方的放入黑名单（一个在客户端的容器），在接收服务器消息时把发送者id在这个黑名单中比对，如果是黑名单内的人发送的，那么就不显示他的信息。

在按下取消拉黑的时候，也会发送一个私信告知对方，同时去除黑名单中的id，这样就可以重新让此发送者的消息出现在我们的屏幕上了。

改进：这两个按键可以合并为一个按键，每次按压就把状态取反，这样一个按键就实现两个按键的功能，避免右键菜单过于臃肿。

* 1. **隐身**

这个按键放在了选择在线状态的combobox里面（如图2所示），选择这个功能的时候就会发送“LOGIN1”开头的消息给服务器，代表自己已经处于隐身状态，此时服务器会在不中断与本客户端连接的同时，告诉其他用户本客户端已经断开连接，也就是服务器转发消息的时候把开头的消息换为“DISCONNECTWED”让其他客户端直接做本客户端下线的处理，安全性更强。此时，隐身客户端可以接收群聊消息，但是不可以接收私聊消息。

如果想要退出隐身模式，只需要把状态调回在线就行，客户端会发送“RELOGIN”开头的消息，此时服务器会转发“LOGIN”开头的消息，这样就可以重新上线了。

遇到的问题：所有在comboBox里面的功能都会被触发两次，如何解决会在系统测试中提到。

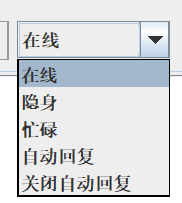


图 4-2 在线模式选择框

* 1. 自动回复

如图2所示自动回复功能也是在combobox中，选中此状态后就会弹出自动回复设定窗口，在这个窗口中可以设定自动回复语句，如图3所示。



图 4-3 自动回复语句设置窗口

点击确定之后，只要是有人私聊我们，我们就会自动回复他我们刚刚设置好的语句。

只要选择取消自动回复之后就可以取消自动回复功能。

改进：在写完这个功能之后我才发现把这个功能放在选择状态的combobox里面是极为错误的，应该把这个功能放在用户列表的右键菜单中，通过一个按键控制，点击一次状态取反，这样设置完自动回复就不会改变在线状态，也更为简洁欸，更符合直觉。另外，应该在接收私聊消息时判断有没有“[自动回复]”或者忙碌状态的开头，这样可以避免两个自动回复或者忙碌状态陷入死循环。

* 1. **忙碌状态**

在combobox中选择忙碌状态会发送“LOGIN2”开头的消息，这个消息会让服务器群发本客户端已忙碌的消息，于此同时，忙碌状态会直接设置忙碌状态专属的自动回复语句，如图4所示。

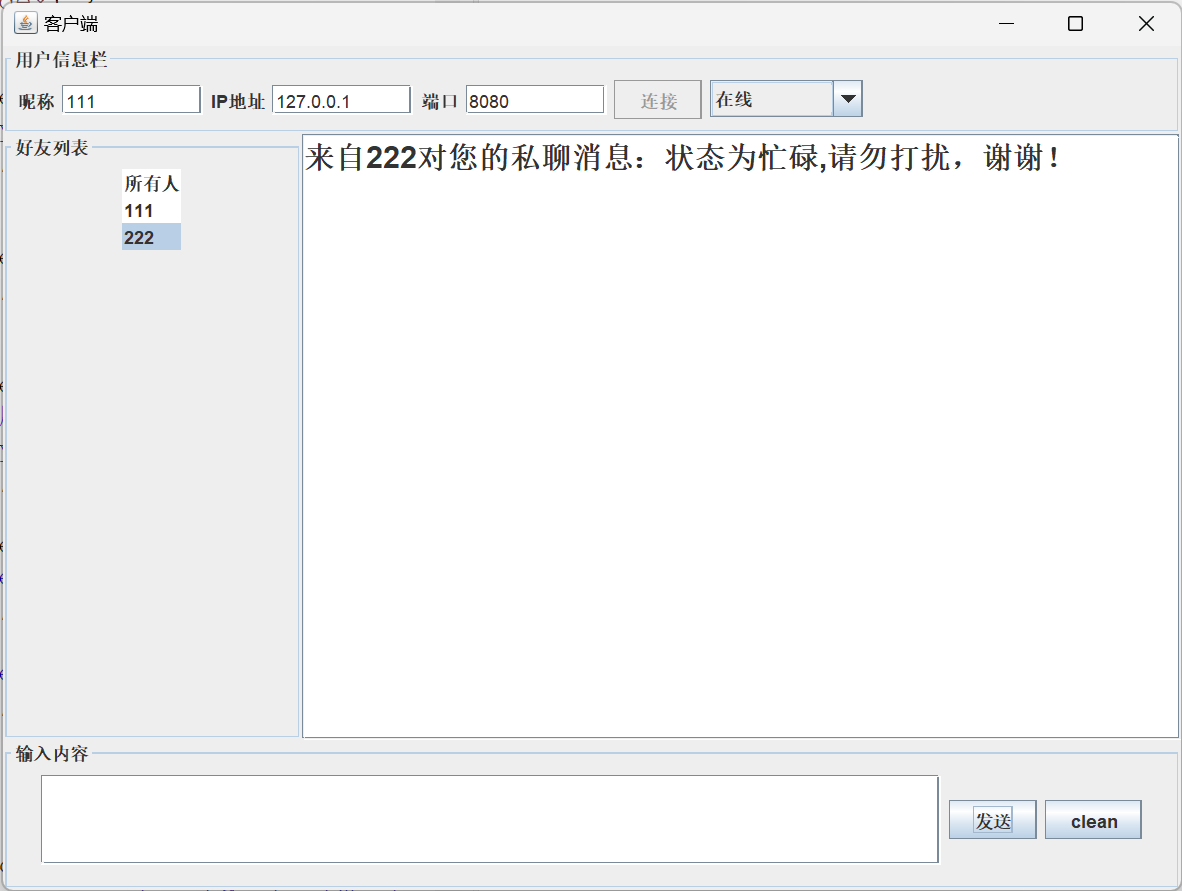


图 4-4 忙碌状态自动回复示例

1. **系统测试**
   1. 服务器

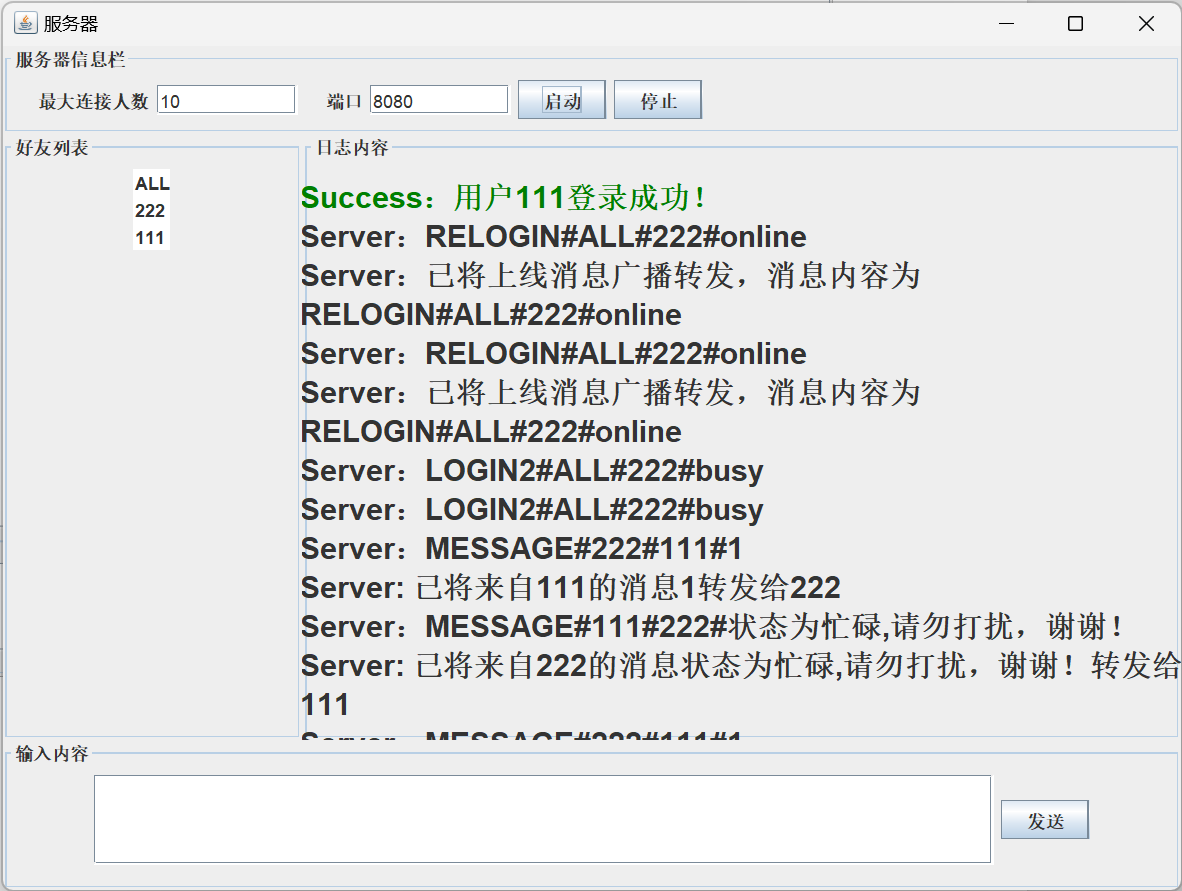


图 5-1 服务器运行测试

可以正常的完成客户端的连接和消息的转发工作。

* 1. 客户端

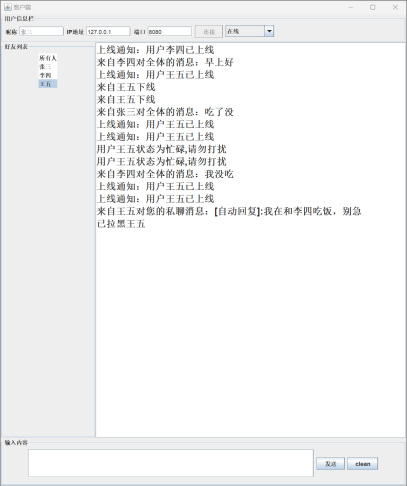
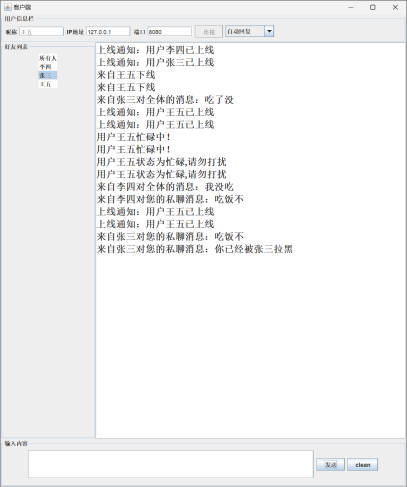
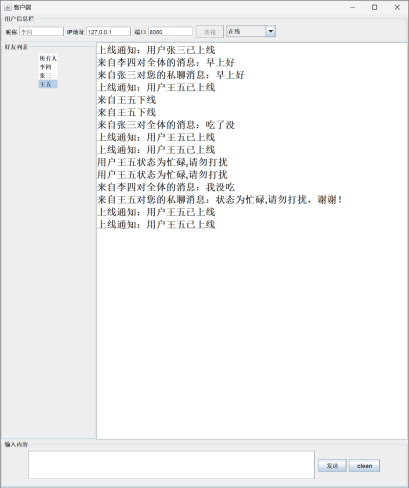


图 5-2 客户端运行测试

如图5所示，客户端功能正常。

前面提到过的combobox所有功能执行两边的问题已经找到了解决办法，只需要加上“if(e.getStateChange() == ItemEvent.SELECTED)”即可，因为combobox改变下拉列表中被选中的项的时候，其实是触发了两次事件：

第一次是上次被选中的项的 State 由 Selected 变为 deSelected ，即取消选择

第二次是本次被选中的项的 State 由 deSelected 变为 Selected ，即新选中，所以，必然的 ItemStateChanged 事件中的代码要被执行两次了。

1. **源代码（部分）**
   1. **创建连接**
      1. **客户端**

**public** **void** ConnectServer() {

// 获取基本信息

String ServerIPAddress = ServerIPAddressText.getText().trim();// 删除头尾空白

**int** ServerPort = Integer.*parseInt*(ServerPortText.getText().trim());

String NickName = NickNameText.getText();

**try** {

// socket相关

socket = **new** Socket(ServerIPAddress, ServerPort);

input = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));

output = **new** PrintStream(socket.getOutputStream());

OnlineClientNickName.addElement("所有人");

OnlineClientNickName.addElement(NickName);

SendMessage(str + NickName);// 向服务器发送本帐号登陆消息

// 为客户端建立线程:

cliendThread = **new** ClientThread();

} **catch** (UnknownHostException e) {

Error("Client：主机地址异常" + e.getMessage());

**return**;

} **catch** (IOException e) {

Error("Client：连接服务器异常" + e.getMessage());

**return**;

}

}

* + 1. **服务器**

**private** **class** ServerThread1 **implements** Runnable{

**public** ServerThread1() {

**new** Thread(**this**).start();

}

@Override

**public** **void** run() {//循环accept下一个socket连接

**while**(**true**) {

**if**(!serverSocket.isClosed()) {

**try** {

Socket socket = serverSocket.accept();//为每个即将连接的客户端新建一个socket

ClientThread clientThread = **new** ClientThread(socket);//每接收到一个服务器请求，就为其新建一个客户线程

String ClientNickName = clientThread.getClientNickName();

clientThreads.put(ClientNickName, clientThread);

Success("用户"+ClientNickName+"登录成功！");

} **catch** (IOException e) {

Error("Server：用户登录失败！"+e.getMessage());

}

}**else** {

Error("Server:未知错误！");

}

}

}

}

* 1. **Combobox（状态选择）**

lib.addItemListener(**new** ItemListener() { //点击状态

@Override

**public** **void** itemStateChanged(ItemEvent e) {

// **TODO** Auto-generated method stub

**int** index = lib.getSelectedIndex();

**switch**(index) {

**case** 0:

str = "RELOGIN#";

SendMessage(str + "ALL#" + NickNameText.getText() + "#online");

isBusy = **false**;

**break**;

**case** 1:

str = "LOGIN1#";

SendMessage(str + "ALL#" + NickNameText.getText() + "#invisible");

**break**;

**case** 2:

str = "LOGIN2#";

SendMessage(str + "ALL#" + NickNameText.getText() + "#busy");

Log("用户"+NickNameText.getText()+"忙碌中！");

isBusy = **true**;

**break**;

**case** 3:

AutoReplyFrame();

autoReply = **true**;

**break**;

**case** 4:

autoReply = **false**;

**break**;

}

}

});

* 1. **右键菜单（拉黑）**

OnlineClientList.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

**if**(e.getButton() == MouseEvent.***BUTTON3***) {

// 显示右键菜单

JPopupMenu popupMenu = **new** JPopupMenu();

JMenuItem menuItem1 = **new** JMenuItem("拉黑");

JMenuItem menuItem2 = **new** JMenuItem("取消拉黑");

menuItem1.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

String selectedName = (String) OnlineClientList.getSelectedValue();

Log("已拉黑" + selectedName);

SendMessage("MESSAGE#" + selectedName + "#" + NickNameText.getText() + "#你已经被" + NickNameText.getText() + "拉黑");

banList.addElement(selectedName);

}

});

menuItem2.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

String selectedName = (String) OnlineClientList.getSelectedValue();

Log("已取消拉黑" + selectedName);

SendMessage("MESSAGE#" + selectedName + "#" + NickNameText.getText() + "#你已经被" + NickNameText.getText() + "取消拉黑");

banList.removeElement(selectedName);

}

});

popupMenu.add(menuItem1);

popupMenu.add(menuItem2);

popupMenu.show(OnlineClientList, e.getX(), e.getY());

}

}

});

1. **总结**

通过本次课设，我学习到了socket创建连接的方式，以及多线程的应用。因为客户端和服务器程序有图形界面，所以也加深了我对swing的理解。

在设计课设的时候也显露出了很多问题，因为临近考试，时间紧张，很多功能就是想到哪写到哪，没有整体的规划，并且在很多地方没有深入思考，反而采取了最简单的实现方式，比如拉黑和取消拉黑分为两个按钮，没有让他们成为一个按钮，也没有让他们互斥；再比如说自动回复放在了状态选择的combobox中，不仅奇怪，而且操作麻烦。这也是我需要改正的，

总而言之，通过这次课设，我学习到了很多知识，也发现了我很多的不足。

1. **参考文献**

【1】计算机网络（第七版）

【2】Java面向对象程序设计（第四版）