网络编程技术程序设计开发文档

周子聪914106840745

阿不都塞米·艾力914106840712

摘    要

近年来随着计算机技术的飞速发展，特别是Internet技术的发展，已经深刻的改变到了人们生活的方方面面。这时过去的种种陈旧的通讯方式，都已经不能满足现代生活的需要。人们希望能够用更低廉的价格，更加方便快捷的方式，来实现即时通讯。这时开发一套基于Internet技术的网络聊天系统变得尤为重要。

* 一、系统需求分析

**1.1需求分析：**

服务器需要同时连接很多个用户，并能提供给这些连接用户所需要的任务处理请求，这就要求服务器能同时处理多个Socket连接。软件需求说明能把软件功能和性能的总体概念描述为具体的软件，从而奠定软件开发基础。该过程将软件计划阶段所确定的软件范围逐步细化到可详细定义的程度并分析出各种不同的软件元素，然后为这些元素找到可行的解决方法。总的来说，是对软件计划阶段建立的软件工作范围的求精和细化。它回答了“系统该做什么”的问题。

**1.1.1系统设计目标**

网络聊天系统主要对象是局域网内的用户，使用交流更简单。设计本系统的

目标如下：

1、一台电脑上能够同时登录多个不同的用户。

2、聊天。所有在线好友发送消息。

**1.1.2系统功能需求**

我们系统功能很简单，连接服务器进行聊天即可。

**1.2.1可行性分析**  
　　 可行性分析首先是对新系统的开发从技术、经济、社会的方面进行分析和研究。在开发系统的过程中，必须要节约开发成本，缩短开发周期。开发该系统成本就要集中在系统的开发和维护上，技术可行性：该系统使用的开发语言是Java，由于Java语言具有跨平台性，很多设计者都会选择它。其次是注意它的社会可行性，网络聊天系统只需要用户的电脑进入局域网就能访问，不必购买其他任何昂贵设备。对系统的组织结构和功能进行分析时，需要将系统调查中有关该业务流程的资料都联系起来作为下一步的分析基础，帮助我们了解业务的具体处理过程，发现和处理系统的调查工作中的欠缺和错误，修改和删除原系统的不合理部分，在新系统基础上优化业务处理流程。

* **二、系统总体设计**

## 2.1.1客户端与服务器的通信

客户端和服务器是基于TCP/IP协议建立连接，并完成数据传输的，其流程图如下图所示。



在Java的基于TCP/IP协议的Socket编程中，服务端使用ServerSocket类开创建，而客户端使用的Socket类。

## 2.1.2客户端与客户端的通信

客户端同客户端的通信是通过服务器转发的形式来实现的，其示意图如下图所示。

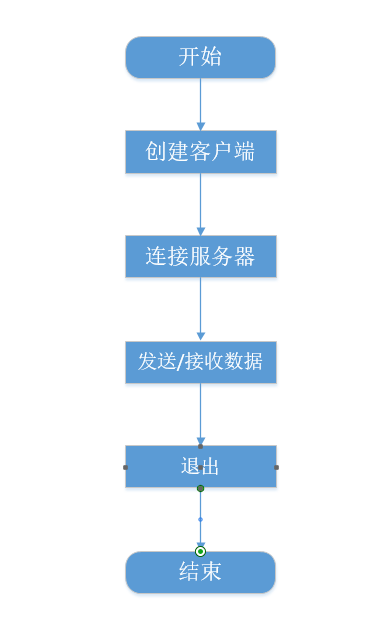
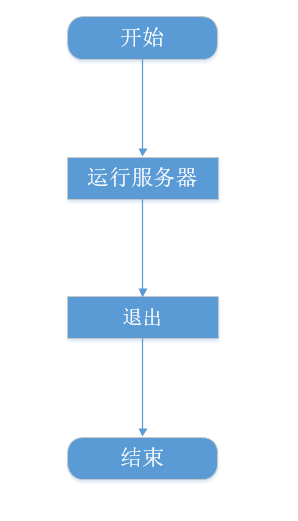
图3.2 通信示意图

需要注意的是，客户端同客户端之间并没有直接的联系，而是通过服务器作为中转站的形式来完成数据传输的。这样可以保证服务器能够对客户端之间的数据进行处理，同时还能保证数据的安全性。

* **三、系统流程图**

根据系统功能画出系统流程图如下：

客户端 服务端

* **四、系统功能模块设计**

## 4.1 服务器端的设计：

现代码如下所示：

**package** chatting;

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.\*;

**public** **class** Server{

**private** **boolean** bStart = **false**;

**private** ServerSocket ss = **null**;

List<Client> clients = **new** ArrayList<Client>();

**private** **int** index = 0;

**public** **void** tcpMonitor() {

**try** {

ss = **new** ServerSocket(8885);

bStart = **true**;

} **catch** (IOException e) {

e.printStackTrace();

}

System.***out***.println("聊天室已建立！");

**try** {

**while** (bStart) {

index++;

Socket s = ss.accept();

Client c = **new** Client(s);

clients.add(c);

System.***out***.println("游客"+s.getPort()+"进入聊天室，当前聊天室人数为：" + index);

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

}

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**try** {

ss.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

**private** **class** Client **implements** Runnable {

DataInputStream dis = **null**;

DataOutputStream dos = **null**;

Socket s = **null**;

**boolean** bStart = **false**;

Client(Socket \_s) {

s = \_s;

**try** {

dis = **new** DataInputStream(s.getInputStream());

dos = **new** DataOutputStream(s.getOutputStream());

} **catch** (IOException e) {

e.printStackTrace();

}

bStart = **true**;

}

**public** **void** sendToEveryClient(String str) {

**try** {

dos.writeUTF(str);

dos.flush();

//System.out.println(str);

} **catch** (IOException e) {

clients.remove(**this**);

}

}

**public** **void** run() {

**try** {

**while** (bStart) {

String str = "游客"+s.getPort()+":"+dis.readUTF();

//System.out.println(str);

**for** (**int** i = 0; i < clients.size(); i++) {

Client c = clients.get(i);

c.sendToEveryClient(str);

}

}

} **catch** (EOFException e) {

clients.remove(**this**);

index--;

System.***out***.println("游客"+s.getPort()+"已离开，当前聊天室人数为：" + (index-1));

} **catch** (SocketException e) {

clients.remove(**this**);

index--;

System.***out***.println("游客"+s.getPort()+"已离开，当前聊天室人数为：" + (index-1));

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**try** {

**if** (s != **null**)

s.close();

**if** (dis != **null**)

dis.close();

**if** (dos != **null**)

dos.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}

**public** **static** **void** main(String[] args) {

Server ts = **new** Server();

ts.tcpMonitor();

}

}

## 4.2 客户端设计

客户端代码如下：

iimport java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

//用户

public class Client extends JFrame{

Socket s = null;

DataOutputStream dos = null;

DataInputStream dis = null;

boolean bConnected = false;

Thread t = new Thread(new RecToServer());

TextArea taContent = new TextArea();

JTextArea tfTxt = new JTextArea(10,5);

JScrollPane sp=new JScrollPane(tfTxt);

JButton send = new JButton("发送");

JButton connect = new JButton("连接");

JButton clear = new JButton("清空");

JPanel p2 = new JPanel();

public void launchFrame() {

p2.add(send);

p2.add(connect);

p2.add(clear);

Container con = this.getContentPane();

con.add(taContent, "North");

con.add(sp,"Center");

con.add(p2, "South");

this.setSize(300, 400);

this.setLocation(400, 400);

this.setTitle("聊天");

tfTxt.setLineWrap(true);

taContent.setEditable(false);

this.setVisible(true);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

connect.addActionListener(new Connect());

send.addActionListener(new SendMsg());

clear.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

taContent.setText("");

}

});

}

public static void main(String[] args) {

Client tc = new Client();

tc.launchFrame();

}

public void connectToServer() {

try {

s = new Socket("219.230.82.65", 8885);

dos = new DataOutputStream(s.getOutputStream());

dis = new DataInputStream(s.getInputStream());

bConnected = true;

} catch (BindException e) {

System.out.println("没有连接到服务器");

} catch (UnknownHostException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void disConnect() {

try {

if (s != null) {

s.close();

}

if (dos != null) {

dos.close();

}

if (dis != null) {

dis.close();

}

System.exit(0);

} catch (IOException e) {

e.printStackTrace();

}

}

private class Connect implements ActionListener {

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand() == "连接") {

//System.out.println("连接成功");

connectToServer();

JOptionPane.showMessageDialog(Client.this,

"连接到服务器", "成功提示", 1);

try {

t.start();

} catch (IllegalThreadStateException ex) {

}

connect.setText("退出");}

else if (e.getActionCommand() == "退出"){

disConnect();

/\*//System.out.println("断开服务器");

JOptionPane.showMessageDialog(Client.this,

"断开服务器", "断开提示", 1);\*/

}

}

}

private class SendMsg implements ActionListener {

public void actionPerformed(ActionEvent e) {

if (connect.getActionCommand() == "连接") {

JOptionPane.showMessageDialog(Client.this,

"没有连接到服务器", "错误提示", 1);

} else {

String str = tfTxt.getText();

tfTxt.setText("");

try {

dos.writeUTF(str);

dos.flush();//清空缓冲区

} catch (SocketException ex) {

JOptionPane.showMessageDialog(Client.this,

"没有连接到服务器", "错误提示", 1);

} catch (IOException ex) {

ex.printStackTrace();

}

}}}

private class RecToServer implements Runnable {

public void run() {

try {

while (bConnected) {

String str = dis.readUTF();

taContent.append(str + "\n");

}

} catch (SocketException e) {

} catch (IOException e) {

e.printStackTrace();

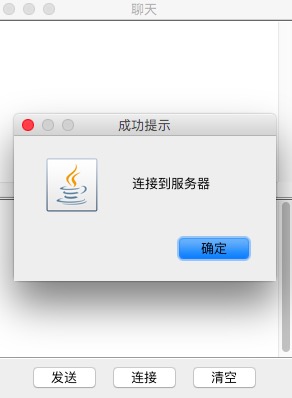
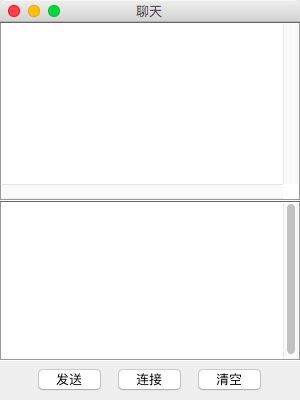
}

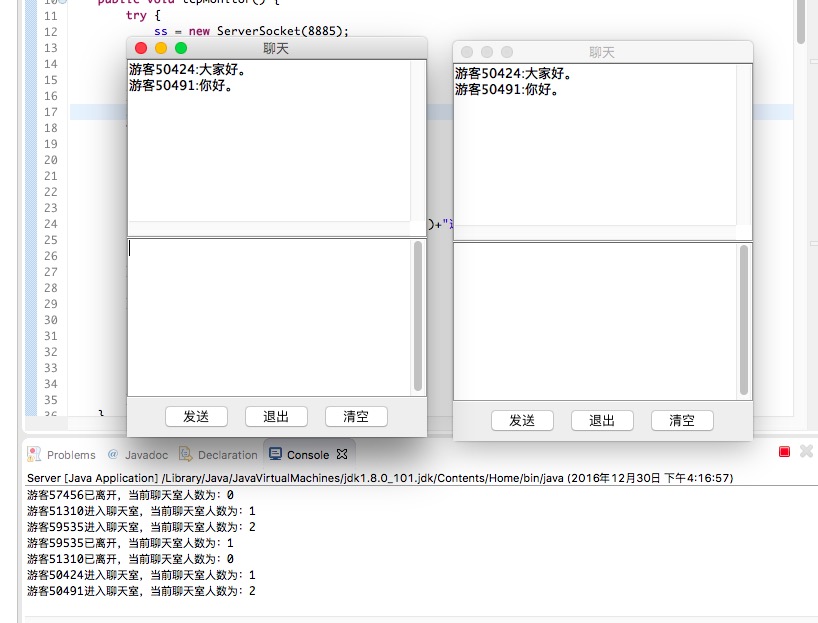
}

}

}

## 4.3 程序运行结果





* **五、总结**

本次实验基于socket通信原理和多线程，开发了一个简单的网络聊天室。通过在本机以及远程计算机的实际操作见证了通信的全过程，了解了网络通信的基本原理以及实现方式。目前基于Internet的应用已经非常多了，这些应用无不在改变人们的传统生活方式。一套网络聊天系统的开发与设计变得尤为重要。

通过完成本次程序我们学会了socket以及多线程技术，对今后开发网络应用起到了良好的开头作用。

通过现场答辩，对比其他组的程序总结出以下不足之处以及过人之处：

首先，界面过于简单，没有经过合理的布局以及对控件的美化；其次，功能过于简单，其他组甚至有图片传输以及文件传输功能。虽然我们组界面、功能都相对简单，但相信我们组是对网络编程这门课程所学的socket通信原理以及多线程操作掌握的相对熟练的一组，对于界面设计不够优美的问题完全可以通过更加强大的开发环境（Android studio或者xcode）对其进行美化（毕竟该项目只采用了最原始的java API，可以发布于任何平台），对于功能简单的问题作出以下解释：近期考试繁多、个人事务繁忙。（说人话）本课程所占学分较少，我们也是本着了解的目的去选修的；另外与我们另外的卓工课程的基于Rest的远程调用实验类似。

通过选修这门课程我们了解了各种基本的网络编程接口，这势必能对将来更好学习和掌握最新最流行的网络编程技术起到作用。