



南昌大学实验报告

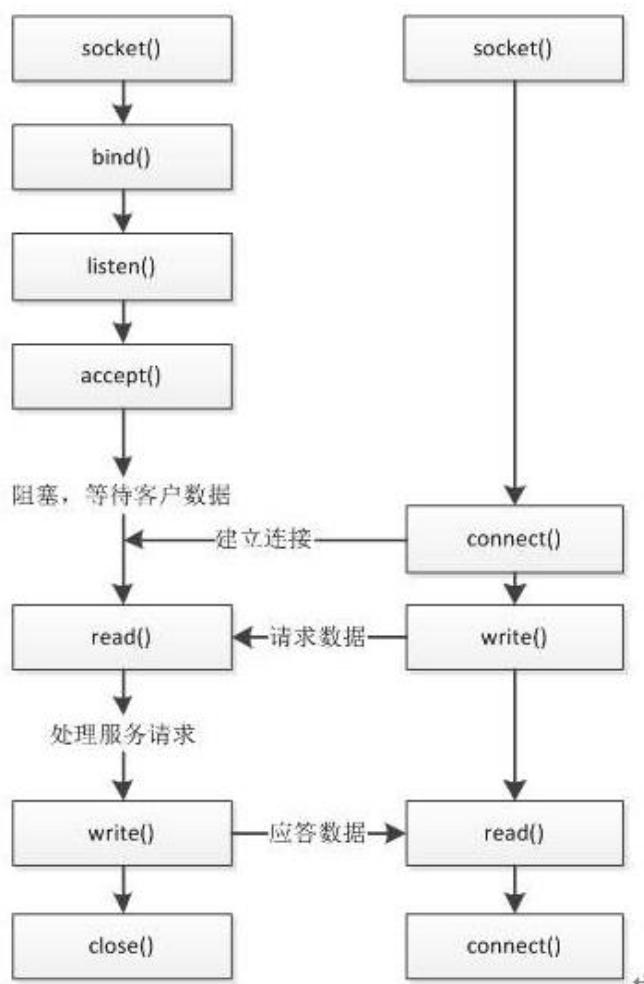
一、实验项目名称

掌握 socket 编程

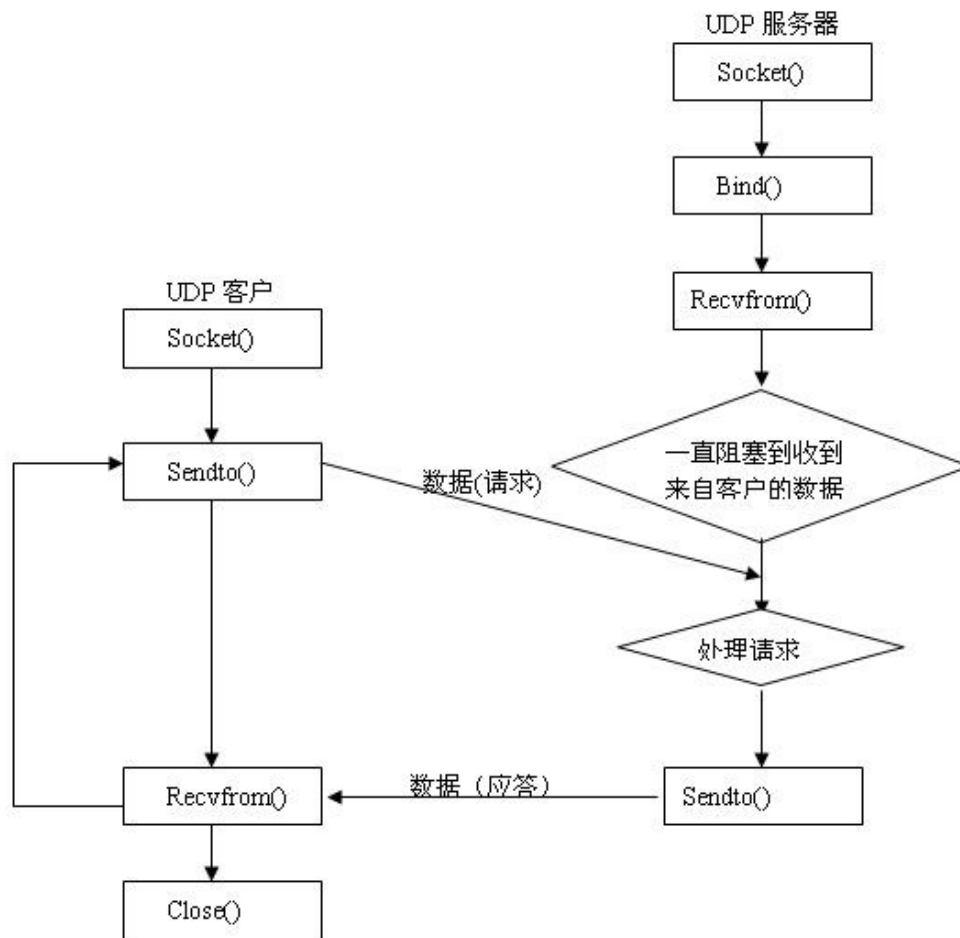
二、实验目的

- 1) 掌握 TCP 和 UDP 的工作原理
- 2) 理解 socket 的概念和应用场合

三、实验基本原理



TCP socket 流程图



UDP socket 流程图

四、主要仪器设备及耗材

IntelliJ IDEA 2022.3.1 设计语言 Java

五、实验步骤 （源代码）

客户端类：

```

1. package com.cyr;
2.
3.
4. import java.io.*;
5. import java .net.*;
6. import java. awt.event.*;
7. import java.awt.*;
8. import javax. swing.*;
9.
10. public class ChatClient implements ActionListener,Runnable{
11.     JTextArea showArea;
12.     JTextField msgText;
  
```

```
13.      JFrame mainJframe;
14.      JButton sentBtn;
15.      JScrollPane JSPane;
16.      JPanel pane;
17.      Container con;
18.      Thread thread=null;
19.      Socket connectToServer;
20.      DataInputStream inFromServer;
21.      DataOutputStream outToServer;
22.
23.      public ChatClient(){
24.          mainJframe=new JFrame("Chat client");
25.          con=mainJframe.getContentPane();
26.          showArea=new JTextArea();
27.          showArea.setEditable(false);
28.          showArea.setLineWrap(true);
29.          JSPane=new JScrollPane(showArea);
30.          msgText=new JTextField();
31.          msgText.setColumns(30);
32.          msgText.addActionListener(this);
33.          sentBtn=new JButton("发送");
34.          sentBtn.addActionListener(this);
35.
36.          pane=new JPanel();
37.          pane.setLayout(new FlowLayout());
38.          pane.add(msgText);
39.          pane.add(sentBtn);
40.
41.          con.add(JSPane, BorderLayout.CENTER);
42.          con.add(pane, BorderLayout.SOUTH);
43.          mainJframe.setSize (500 ,400);
44.          mainJframe.setVisible (true);
45.          mainJframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46.
47.          // 创建套接字连接到服务器
48.          try{
49.              connectToServer=new Socket("localhost",5500);
50.              inFromServer=new DataInputStream(connectToServer.getInputStream());
51.              outToServer=new DataOutputStream(connectToServer.getOutputStream());
52.              showArea.append("连接成功，请说话...\n");
53.
```

```

54.          //创建线程在后台处理对方的消息
55.          thread=new Thread(this);
56.          thread.setPriority(Thread.MIN_PRIORITY);
57.          thread.start();
58.      } catch (UnknownHostException e1){
59.          e1.printStackTrace();
60.      } catch (IOException e1){
61.          showArea.append("抱歉，未能连接到服务器！\n");
62.          msgText.setEditable(false);
63.          sentBtn.setEnabled(false);
64.      }
65.  }
66.
67.  public static void main(String[] args){
68.      //开启客户端
69.      new ChatClient();
70.  }
71.
72.
73.  @Override
74.  //
75.  public void actionPerformed(ActionEvent e){
76.      String s=msgText.getText();
77.      if (s.length()>0){
78.          try{
79.              outToServer.writeUTF(s);
80.              outToServer.flush();
81.              showArea.append("我[客户
端]: "+msgText. getText()+"\n");
82.          } catch (IOException e1){
83.              showArea.append("你的消息:
"+""+msgText.getText()+"未能发送出去！\n");
84.          }
85.      }
86.
87.  }
88.
89.      //本线程负责将服务器传来的消息显示在对话区域
90.      public void run(){
91.          try{
92.              while (true){
93.                  showArea.append("服务端说
->:"+inFromServer.readUTF()+"\n");
94.                  Thread.sleep(1000);

```

```

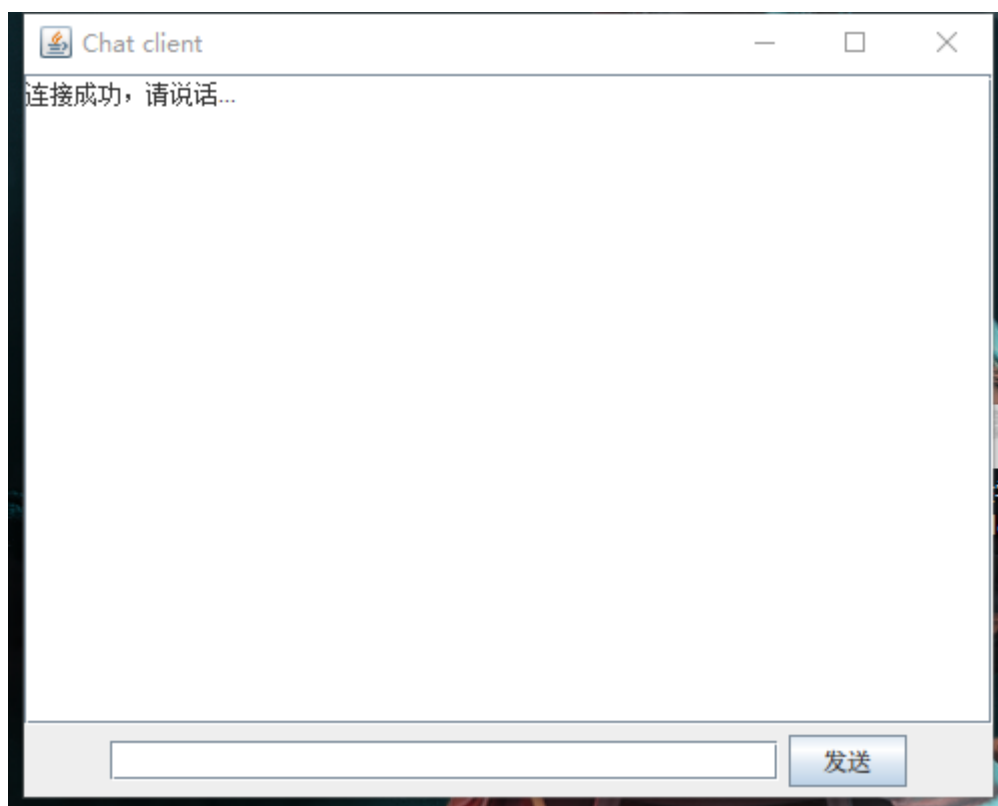
95.         }
96.     } catch (IOException e) {
97.         e.printStackTrace();
98.     } catch (InterruptedException e) {
99.         e.printStackTrace();
100.    }
101. }
102.
103.
104. }

```

结

果

:



服务器类:

```

1.     package com.cyr;
2.     import java.io.*;
3.     import java.net. *;
4.     import java. awt.event. *;
5.     import java.awt.*;
6.     import javax. swing.*;
7.
8.     import static java.lang.Thread.sleep;
9.
10.    public class ChatServer implements ActionListener, Runnable {

```

```
11.
12.     JTextArea showArea;
13.     JTextField msgText;
14.     JFrame mainJframe;
15.     JButton sentBtn;
16.     JScrollPane JSPane;
17.     JPanel pane;
18.     Container con;
19.     Thread thread = null;
20.     ServerSocket serverSocket;
21.     Socket connectToClient;
22.     DataInputStream inFromClient;
23.     DataOutputStream outToClient;
24.
25.     public ChatServer() {
26.         // 绘制界面
27.         mainJframe = new JFrame("Chat server");
28.         con = mainJframe.getContentPane();
29.
30.         // 文本框
31.         showArea = new JTextArea();
32.         showArea.setEditable(false);
33.         showArea.setLineWrap(true);
34.         JSPane = new JScrollPane(showArea);
35.
36.         // 文字区域
37.         msgText = new JTextField();
38.         msgText.setColumns(30);
39.         msgText.addActionListener(this);
40.
41.         // 发送按钮
42.         sentBtn = new JButton("发送");
43.         sentBtn.addActionListener(this);
44.
45.         // 文本框区域
46.         pane = new JPanel();
47.         pane.setLayout(new FlowLayout());
48.         pane.add(msgText);
49.         pane.add(sentBtn);
50.         con.add(JSPane, BorderLayout.CENTER);
51.         con.add(pane, BorderLayout.SOUTH);
52.         mainJframe.setSize(500, 400);
53.         mainJframe.setVisible(true);
```

```

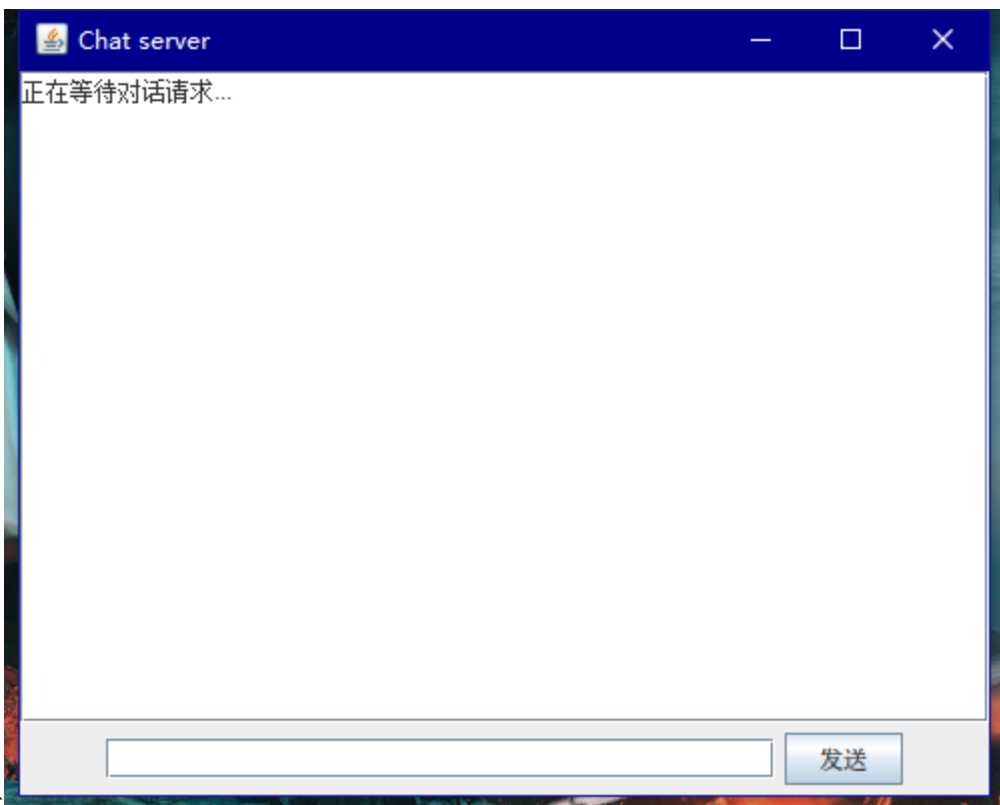
54.         mainJframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
55.         try {
56.             // 创建 socket 套接字
57.             serverSocket = new ServerSocket(5500);
58.             // 打印
59.             showArea.append("正在等待对话请求...\n");
60.             connectToClient = serverSocket.accept();
61.             inFromClient = new DataInputStream(connectToClient
                .getInputStream());
62.             outToClient = new DataOutputStream(connectToClient
                .getOutputStream());
63.             thread = new Thread(this);
64.             thread.setPriority(Thread.MIN_PRIORITY);
65.             thread.start();
66.         } catch (IOException e) {
67.             showArea.append("对不起，不能创建服务器\n");
68.             msgText.setEditable(false);
69.             sentBtn.setEnabled(false);
70.         }
71.     }
72.
73.     public static void main(String[] args){
74.         // 开启服务器
75.         new ChatServer();
76.     }
77.
78.     @Override
79.     // 响应按钮事件并且发送消息给对方
80.     public void actionPerformed(ActionEvent e) {
81.         String s = msgText.getText();
82.         if (s.length() > 0) {
83.             try {
84.                 outToClient.writeUTF(s);
85.                 outToClient.flush();
86.                 showArea.append("我[服务端]说:
87.                 " + msgText.getText() + "\n");
88.                 msgText.setText(null);
89.             } catch (IOException e1) {
90.                 showArea.append("你的消息:
91.                 "" + msgText.getText() + ""未能发出去!\n");
92.             }
93.         }
94.     }

```

```

93.
94.     @Override
95.     // 事件负责将客户机传来的信息显示在对话区域
96.     public void run() {
97.         try{
98.             while (true){
99.                 showArea.append("客户端
->:" + inFromClient.readUTF()+"\n");
100.                 sleep(1000);
101.             }
102.         } catch (IOException e) {
103.             e.printStackTrace();
104.         } catch (InterruptedException e) {
105.             e.printStackTrace();
106.         }
107.     }
108.
109. }

```



结果

六、思考讨论题或体会或对改进实验的建议

本次实验设计了两个类一个是服务器类一个是客户端类。这两个类的重复度很高。基本可以说是 80% 的重复的所以说只要写一个就可以大部分复用。从 gui 到发送其实很简单。只需要加一个文本区域，和一个发送按钮其他只需要把文字用 string 存储发送。

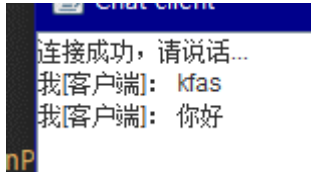


连接成功，请说话...

我[客户端]: kfas

我[客户端]: 你好

结果



连接成功，请说话...

我[客户端]: kfas

我[客户端]: 你好

七、参考资料

[基于 TCP 异步的聊天室程序](#)

[java 网络编程，通过 TCP,Socket 实现多对一的局域网聊天室](#)