

**计算机网络编程**

**课程设计报告**

**题目：**利用UDP编写一个网络程序，该程序服务向服务器

发一个温度值，客户端显示一条动态温度曲线

**学 院 计算机科学学院**

**专 业 软件工程**

**班 级 软件工程1602**

**姓 名 张金莉**

**学 号 41609050231**

**同组人姓名 曹镇德 任浩 雷帅帅**

**同组人学号 207 215 223**

**指导教师 薛文生**

**2019年1月6日**

## 一、任务概述

**1.1 编写目的**

1）熟悉并学会使用C/S编程模式

2）采用 Java 语言，了解 Socket 的原理，并熟悉使用 Socket 。并利用套接字 Socket 编写出指定题目的交互信息

3）掌握对传输层协议TCP/UDP和客户/服务器数据通信的原理及方式

4）进一步理解和领会计算机网络基本结构、计算机网络基础理论、计算机网络编程技术及Java 语言开发方法，把所学的知识运用到具体的程序设计当中去，编写一个接近实际的应用程序并掌握网络编程技术及应用程序分析、设计、编程和调试的整个过程。

**1.2 项目任务和要求**

采用UDP或者TCP编写一个网络程序，该程序的服务端每隔1秒向客户端发送一个温度值，客户端能够接收该温度值并显示一条动态温度曲线。

**1.3 参考资料**

CSDN网站上的资料：

1. **DatagramSocket的构造器：**  
   DatagramSocket()：创建一个DatagramSocket实例，并将该对象绑定到本机默认IP地址、本机所有可用端口中随机选择的某个端口。   
   DatagramSocket(int prot)：创建一个DatagramSocket实例，并将该对象绑定到本机默认IP地址、指定端口。   
   DatagramSocket(int port, InetAddress laddr)：创建一个DatagramSocket实例，并将该对象绑定到指定IP地址、指定端口。   
   通过上面三个构造器中任意一个构造器即可创建一个DatagramSocket实例，通常在创建服务器时，我们创建指定端口的 DatagramSocket实例——这样保证其他客户端可以将数据发送到该服务器。一旦得到了DatagramSocket实例之后，就可以通过如下两 个方法来接收和发送数据：   
   receive(DatagramPacket p)：从该DatagramSocket中接收数据报。   
   send(DatagramPacket p)：以该DatagramSocket对象向外发送数据报。
2. **DatagramPacket的构造器**：   
   DatagramPacket(byte buf[],int length)：以一个空数组来创建DatagramPacket对象，该对象的作用是接收DatagramSocket中的数据。   
   DatagramPacket(byte buf[], int length, InetAddress addr, int port)：以一个包含数据的数组来创建DatagramPacket对象，创建该DatagramPacket时还指定了IP地址和端口——这就决定了 该数据报的目的。   
   DatagramPacket(byte[] buf, int offset, int length)：以一个空数组来创建DatagramPacket对象，并指定接收到的数据放入buf数组中时从offset开始，最多放length个字节。   
   DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)：创建一个用于发送的DatagramPacket对象，也多指定了一个offset参数。   
   在接收数据前，应该采用上面的第一个或第三个构造器生成一个DatagramPacket对象，给出接收数据的字节数组及其长度。然后调用 DatagramSocket 的方法receive()等待数据报的到来，receive()将一直等待（也就是说会阻塞调用该方法的线程），直到收到一个数据报为止 .

## 二、开发环境

操作系统：window 10

开发环境：IDEA

## 三、项目需求分析

1.项目需求

1）学会使用C/S模式编写一个网络程序，实现一个温度折线图。

2）了解并学会使用Java套接字，即DatagramSocket和DatagramPacket。并且还要了解DatagramSocket和DatagramPacket的应用场景。

3）温度折线图是动态的，实时的。即在服务器端产生的数据要不断地发送给客户端，而客户端也要不断的接收服务器端发送过来的数据。

4）利用JFreeChart是JAVA平台上的一个开放的图表绘制类库。可生成[时序图](https://baike.baidu.com/item/%E6%97%B6%E5%BA%8F%E5%9B%BE/3659178" \t "https://baike.baidu.com/item/jfreechart/_blank)（time series）

## 四、项目设计

**1.总体思路**

1）采用C/S模式编写一个客户端，一个服务端。服务器端利用DatagramSocket创建套接字并把DatagramPacket创建的数据包利用send()方法发送到客户端。

2）客户端创建利用DatagramSocket创建套接字并用DatagramPacket创建一个数据包，在利用receive()方法接收数据包。

3）在客户端利用JFreeChart类提供的方法创建一个时序图。把从服务器端接收到的数据包里的数据转换成对应的温度，插入到时序图中去。

**2.网络编程三要素**

1）IP地址：InetAddress

　 获取任意主机：public static InetAddress getByName(String host)：根据主机名或者IP地址的字符串表示得到IP地址对象

1. 端口号：用于标识进程的逻辑地址，不同进程的标识

· 物理端口 网卡口

· 逻辑端口 我们指的就是逻辑端口

A:每个网络程序都会至少有一个逻辑端口

　B:用于标识进程的逻辑地址，不同进程的标识

　C:有效端口：0~65535，其中0~1024系统使用或保留端口。

3）传输协议：UDP

　UDP：将数据源和目的封装成数据包中，不需要建立连接；每个数据报的大小在限制在64k；因无连接，是不可靠协议；不需要建立连接，速度快。

**3.Socket**

1）Socket套接字：网络上具有唯一标识的IP地址和端口号组合在一起才能构成唯一能识别的标识符套接字。

2）Socket原理机制：通信的两端都有Socket。网络通信其实就是Socket间的通信。数据在两个Socket间通过IO传输。

**4.UDP传输思路设计**

· DatagramSocket与DatagramPacket

· 建立发送端，接收端。

· 建立数据包。

· 调用Socket的发送接收方法。

· 关闭Socket。

· 发送端与接收端是两个独立的运行程序。

1）UDP传输-发送端思路

  A：建立udp的socket服务

  B：将要发送的数据封装成数据包

  C：通过udp的socket服务,将数据包发送出

  D：关闭资源

2）UDP传输-接收端思路

  A：建立udp的socket服务.

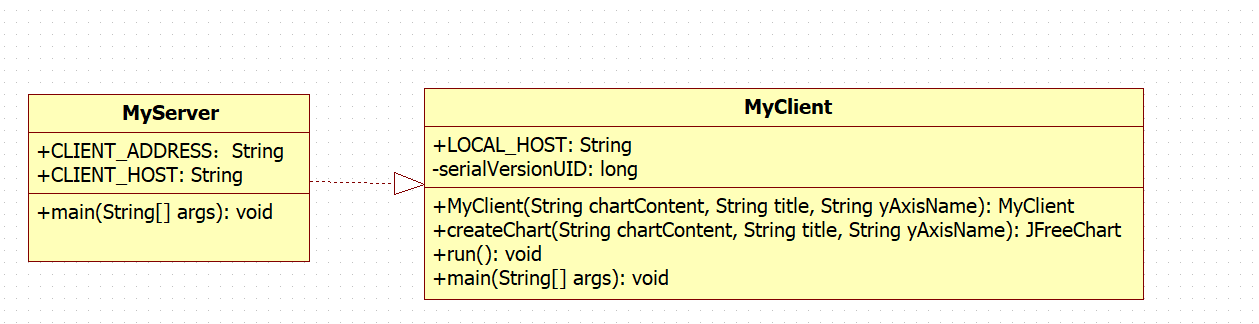
  B：通过receive方法接收数据

  C：将收到的数据存储到数据包对象中

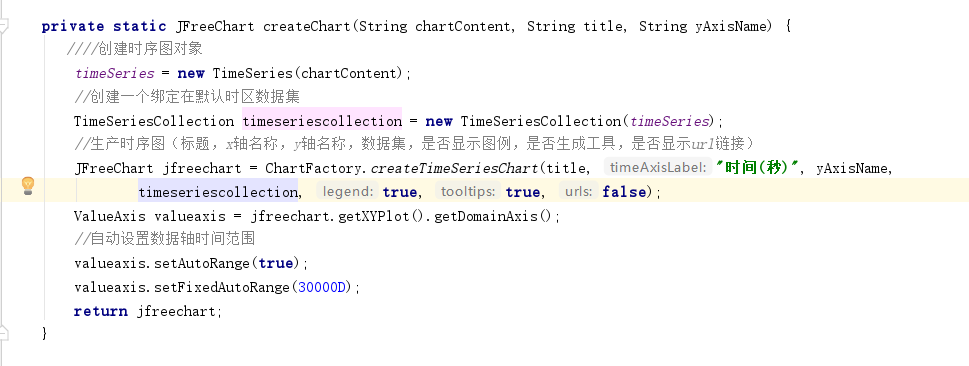
  D：通过数据包对象的功能来完成对接收到数据进行解析.

  E：可以对资源进行关闭

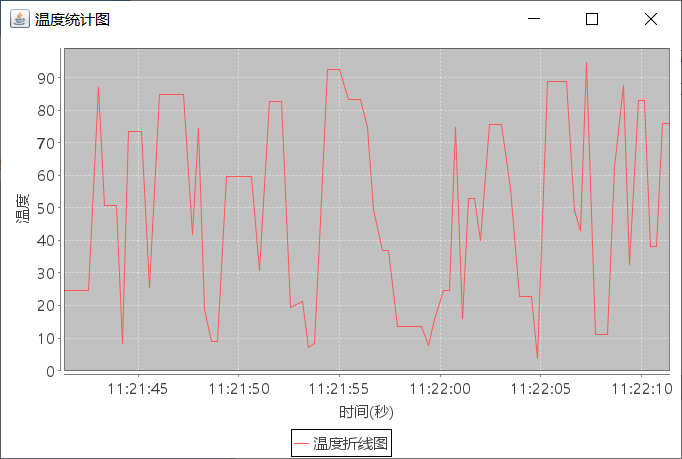
**5.类图**



1. **时序图的实现**



## 五、程序的运行和测试结果



## 六、设计中的问题及心得

在本次实如何验中，我学会了使用UDP协议的DatagramSocket与DatagramPacket的使用。在本次实验中我不知道如何产生连续的温度，所以我在网站上找到了一盒函数Math.random（），用它生成的各种随机数。我们使用这些数字来生成时间序列图表的温度。但是问题又来了，怎么显示温度折线图，我在CSDN上找了好多人写的东西发现JFreeChart是一个免费的100％Java图表库，使开发人员可以轻松地在他们的应用程序中显示专业质量的图表。所以我用这个来做了折线图。虽然最后实现了温度折线图，但是生成的折线图上没有显示每个温度的度数。

### 七、附录：程序清单

**客户端程序**

**import** org.jfree.chart.ChartFactory;  
**import** org.jfree.chart.ChartPanel;  
**import** org.jfree.chart.JFreeChart;  
**import** org.jfree.chart.StandardChartTheme;  
**import** org.jfree.chart.axis.ValueAxis;  
**import** org.jfree.data.time.Millisecond;  
**import** org.jfree.data.time.TimeSeries;  
**import** org.jfree.data.time.TimeSeriesCollection;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.WindowAdapter;  
**import** java.awt.event.WindowEvent;  
**import** java.net.DatagramPacket;  
**import** java.net.DatagramSocket;  
**public class** MyClient **extends** ChartPanel **implements** Runnable {  
  
 */\*\*  
 \* 本地接收数据包的端口号  
 \* 本地接收数据包的端口号  
 \* \*接收 UDP 数据包的步骤如下：  
 \* \*使用 DatagramSocket 创建数据包套接字，并将其绑定到指定的端口。  
 \* \*使用 DatagramPacket 创建字节数组来接收数据包。  
 \* \*使用 receive() 方法接收 UDP 包。  
 \*/* **public static final** String ***LOCAL\_HOST*** = **"12345"**;  
  
 **private static final long *serialVersionUID*** = 1L;  
 *//* **private static** TimeSeries *timeSeries*;  
  
 **public** MyClient(String chartContent, String title, String yAxisName) {  
 **super**(*createChart*(chartContent, title, yAxisName));  
 }  
  
 **private static** JFreeChart createChart(String chartContent, String title, String yAxisName) {  
 *////创建时序图对象  
 timeSeries* = **new** TimeSeries(chartContent);  
 *//创建一个绑定在默认时区数据集* TimeSeriesCollection timeseriescollection = **new** TimeSeriesCollection(*timeSeries*);  
 *//生产时序图（标题，x轴名称，y轴名称，数据集，是否显示图例，是否生成工具，是否显示url链接）* JFreeChart jfreechart = ChartFactory.*createTimeSeriesChart*(title, **"时间(秒)"**, yAxisName, timeseriescollection, **true**, **true**, **false**);  
 ValueAxis valueaxis = jfreechart.getXYPlot().getDomainAxis();  
 *//自动设置数据轴时间范围* valueaxis.setAutoRange(**true**);  
 valueaxis.setFixedAutoRange(30000D);  
 **return** jfreechart;  
 }  
  
 @Override  
 **public void** run() {  
 DatagramSocket ds = **null**;  
 **byte**[] bys = **null**;  
 DatagramPacket dp = **null**;  
 **double** ret = 0;  
 **while** (**true**) {  
 **try** {  
 Thread.*sleep*(300);  
 *//创建数据包套接字，并将其绑定到指定的端口。；  
 //Integer.parseInt 方法的作用, 就是从字符串转换成一个10进制的整数* ds = **new** DatagramSocket(Integer.*parseInt*(***LOCAL\_HOST***));  
 *// 创建一个字节数组* bys = **new byte**[1024];  
 dp = **new** DatagramPacket(bys, bys.**length**);  
  
 *// 接收数据包* ds.receive(dp);  
 *//bys是个字节数组，通过这个字节数组去构建一个String对象  
 //通过这个String解析得到一个小数  
 //ret就是需要传入的温度值，是个小数  
 //Double.parseDouble(String)将字符串转换为双精度* ret = Double.*parseDouble*(**new** String(bys));  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally** {  
 *timeSeries*.add(**new** Millisecond(), ret);  
 ds.close();  
 }  
 }  
 }  
  
 **public static void** main(String[] args) {  
 *// 设置显示样式，避免中文乱码* StandardChartTheme standardChartTheme = **new** StandardChartTheme(**"CN"**);  
 standardChartTheme.setExtraLargeFont(**new** Font(**"微软雅黑"**, Font.***BOLD***, 20));  
 standardChartTheme.setRegularFont(**new** Font(**"微软雅黑"**, Font.***PLAIN***, 15));  
 standardChartTheme.setLargeFont(**new** Font(**"微软雅黑"**, Font.***PLAIN***, 15));  
 ChartFactory.*setChartTheme*(standardChartTheme);  
 *//创建一个新的窗体，名称为“温度统计图”* JFrame frame = **new** JFrame(**"温度统计图"**);  
 MyClient realTimeChart = **new** MyClient(**"温度折线图"**, **""**, **"温度"**);  
 *//向容器中添加组件* frame.getContentPane().add(realTimeChart, **new** BorderLayout().***CENTER***);  
 *//调整窗口大小* frame.pack();  
 *//显示组件* frame.setVisible(**true**);  
 *//初始化温度图的参数以及点击退出按钮的时候结束当前线程* (**new** Thread(realTimeChart)).start();  
 *//在窗口添加一个Windows事件消息，目的是我们关闭窗口的时候可以正常的退出* frame.addWindowListener(**new** WindowAdapter() {  
 @Override  
 **public void** windowClosing(WindowEvent windowevent) {  
 System.*exit*(0);  
 }  
 });  
 }  
}

**服务器端程序：**

**import** java.net.DatagramPacket;  
**import** java.net.DatagramSocket;  
**import** java.net.InetAddress;  
  
*/\*  
 \* /\*\*  
 \* \* 客户服务器端口号  
 \* \*下面是在 Java 中使用 UDP 协议发送数据的步骤。  
 \* \*使用 DatagramSocket() 创建一个数据包套接字。  
 \* \*使用 DatagramPacket() 创建要发送的数据包。  
 \* \*使用 send() 方法发送数据包  
 \*/***public class** MyServer {  
 */\*\*  
 \* 客户服务器IP地址  
 \*/* **public static final** String ***CLIENT\_ADDRESS*** = **"127.0.0.1"**;  
  
 */\*\*  
 \* 客户服务器端口号  
 \*/* **public static final** String ***CLIENT\_HOST*** = **"12345"**;  
 **public static void** main(String[] args) {  
  
 **while**(**true**){  
 **try** {  
 Thread.*sleep*(300);  
 DatagramSocket ds = **new** DatagramSocket();  
 *//生成随机温度* **byte**[] bys = (Math.*random*() \* 100+**""**).getBytes();  
 *//打包数据* DatagramPacket dp = **new** DatagramPacket(bys, bys.**length**, InetAddress.*getByName*(***CLIENT\_ADDRESS***), Integer.*parseInt*(***CLIENT\_HOST***));  
 *//发送数据* ds.send(dp);  
 ds.close();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 }  
}