Java进阶1 第16天

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【应用】网络编程概述**
   1. 【理解】能够阐述IP地址和端口号的概念
   2. 【理解】能够理解网络模型的概念
2. **【应用】UDP协议**
   1. 【应用】能够独立实现UDP的发送端和接收端
   2. 【应用】能够独立实现UDP聊天大厅案例
3. **【应用】TCP协议**
   1. 【应用】 能够独立实现TCP的客户端和服务端
   2. 【应用】 能够独立实现服务端向客户端发送数据
   3. 【应用】 能够独立实现服务端循环读取客户端发送的数据
   4. 【应用】 能够独立实现上传文件案例

# 网络编程概述

## IP地址&InetAddress

IP是每台电脑在互联网上的唯一标识符。

一个IPV4的地址是由四段0—255的数字组成：192.168.0.100。

Java中可以使用InetAddress类表示一个IP地址。

常见方法：

public static InetAddress **getLocalHost**()

throws UnknownHostException 获取本地主机

public static InetAddress **getByName**(String host)

throws UnknownHostException 依据主机名获取主机

public String **getHostName**() 获取主机名称

public String **getHostAddress**() 获取主机字符串形式的IP

其他：

ipconfig用于DOS获取计算机IP设置。

ping用于判断两台计算机连接是否通畅。

127.0.0.1与localhost类似，均代表本机地址。

xxx.xxx.xxx.255 广播地址，即该网段下所有用户均可以被通知到。

### 案例代码一:

|  |
| --- |
| **package** com.igeek\_01;  **import** java.net.InetAddress;  **import** java.net.UnknownHostException;  /\*\*  \* **@ClassName**: IPDemo  \* **@Description**: IP描述  \* **@date** 2018年2月6日 下午1:03:43  \* Company www.igeekhome.com  \*  \* IP地址是一台电脑在网络当中最常用的唯一标识，  \* 也可以使用主机名(计算机名)在网络中找到电脑主机，但是主机名有可能重复，会带来许多不安全的因素，弃用主机名，  \* 主要使用IP地址  \*  \* 由四个网段组成：  \* 其本质为：  \* 4个字节的二进制数  \* 二进制数 11111111 10101010 01010101 11110000  \* 十进制 数字1 数字2 数字3 数字4  \*  \* 由于每段数字均由一个字节翻译成十进制数，所有最大为255  \*  \* Java中的InetAddress代表地址，主要代表IP地址  \*  \* 获取地址对象：  \* public static InetAddress getLocalHost() throws UnknownHostException：  \* 获取本地主机  \* public static InetAddress getByName(String host) throws UnknownHostException：  \* 依据主机名(或者是IP地址字符串)获取主机  \*  \* 通过地址对象获取IP字符串或者主机名称字符串:  \* public String getHostName(): 获取主机名称  \* public String getHostAddress(): 获取主机字符串形式的IP  \*  \*/  **public** **class** IPDemo {  **public** **static** **void** main(String[] args) **throws** UnknownHostException {  //获取本地IP地址  InetAddress localHost = InetAddress.*getLocalHost*();    //获取IP字符串或者主机名称字符串  String hostName = localHost.getHostName();  System.***out***.println(hostName);    String hostAddress = localHost.getHostAddress();  System.***out***.println(hostAddress);    //获取其他主机IP地址  InetAddress byName = InetAddress.*getByName*("SKY-20170629LIV");  System.***out***.println(byName);    InetAddress byName2 = InetAddress.*getByName*("192.168.2.99");  System.***out***.println(byName2);  }  } |

## 端口号

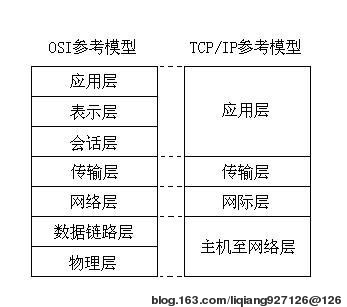
通过IP地址可以连接到指定计算机，但如果想访问目标计算机中的某个应用程序，还需要指定端口号。在计算机中，不同的应用程序是通过端口号区分的。端口号是用两个字节（16位的二进制数）表示的，它的取值范围是0~65535，其中，0~1023之间的端口号用于一些知名的网络服务和应用，用户的普通应用程序需要使用1024以上的端口号，从而避免端口号被另外一个应用或服务所占用。

## 网络模型

网络模型是计算机网络通讯规范，一般符合以下规则：

OSI（Open System Interconnection开放系统互连）参考模型；

TCP/IP参考模型。



## Scoket套接字

在Internet上的主机一般运行了多个服务软件，同时提供几种服务。每种服务都打开一个Socket，并绑定到一个端口上，不同的端口对应于不同的服务。

Socket就是为网络编程提供的一种机制，又叫套接字编程。Socket需要理解以下几点内容：

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；

Socket编程一般使用TCP与UDP两种协议，这里主要介绍TCP。

# UDP协议

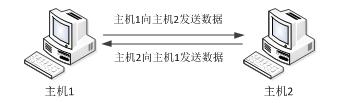
## 概述

UDP是无连接通信协议，即在数据传输时，数据的发送端和接收端不建立逻辑连接。简单来说，当一台计算机向另外一台计算机发送数据时，发送端不会确认接收端是否存在，就会发出数据，同样接收端在收到数据时，也不会向发送端反馈是否收到数据。

由于使用UDP协议消耗资源小，通信效率高，所以通常都会用于音频、视频和普通数据的传输，例如视频会议都使用UDP协议，因为这种情况即使偶尔丢失一两个数据包，也不会对接收结果产生太大影响。

但是在使用UDP协议传送数据时，由于UDP的面向无连接性，不能保证数据的完整性，因此在传输重要数据时不建议使用UDP协议。

UDP的交换过程如下图所示。



## UDP协议实现

发送和接收均可以使用的Socket类：

DatagramSocket

数据打包的相关类：

DatagramPacket

使用UDP发送或接收数据的步骤：

1、建立发送端，接收端；

2、创建数据；

3、建立数据包；

4、调用Socket的发送、接收方法；

5、关闭Socket。

需求：教师的电脑的一个程序发送数据，一个程序接收数据，使用的教师本机的IP。

### 发送端

#### 案例代码二:

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.InetAddress;  **import** java.net.SocketException;  **import** java.net.UnknownHostException;  /\*\*  \* **@ClassName**: UDPSend  \* **@Description**: UDP的发送端  \* **@date** 2018年2月6日 下午4:08:39  \* Company www.igeekhome.com  \*  \* 发送和接收均可以使用的Socket类：  \* DatagramSocket  \*  \* 数据打包的相关类：  \* DatagramPacket  \*  \* 使用UDP发送数据的步骤：  \* 1、建立发送端，空参构造；  \* 2、创建数据；  \* 3、建立数据包；  \* public DatagramPacket(byte[] buf, 数据打包成的字节数组  int length, 数据的长度  InetAddress address, 目标程序IP地址  int port) 目标程序端口  \* 4、调用Socket的发送；  \* public void send(DatagramPacket p) 发送数据报包  \* 5、关闭Socket。  \*  \* ps：发送端与接收端是两个对立的运行程序。  \*/  **public** **class** UDPSend {  **public** **static** **void** main(String[] args) **throws** IOException {  //1、建立发送端  DatagramSocket ds = **new** DatagramSocket();    //2、创建数据  String s = "i love java";    //3、建立数据包  **byte**[] data = s.getBytes();  **int** length = data.length;  InetAddress address = InetAddress.*getByName*("192.168.2.99");  **int** port = 12345;  DatagramPacket dp = **new** DatagramPacket(data , length, address, port );    //4、调用Socket的发送  ds.send(dp);    //5、关闭Socket  ds.close();  }  } |

### 接收端

#### 案例代码三:

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.InetAddress;  **import** java.net.SocketException;  /\*\*  \* **@ClassName**: UDPReceive  \* **@Description**: UDP的接收端  \* **@date** 2018年2月6日 下午4:31:45  \* Company www.igeekhome.com  \*  \* 发送和接收均可以使用的Socket类：  \* DatagramSocket  \*  \* 数据打包的相关类：  \* DatagramPacket  \*  \* 使用UDP接收数据的步骤：  \* 1、建立接收端；  \* public DatagramSocket(int port): 指定程序端口号，方便发送端寻找  \* 2、建立数据包，用于接收数据；  \* public DatagramPacket(byte[] buf, int length)  \* 3、调用Socket的接收方法；  \* 4、解析数据 ；  \* public byte[] getData(): 获取数据报包中的数据  \* public int getLength()： 获取数据报包中的数据长度  \* public InetAddress getAddress(): 获取发送方的IP  \* public int getPort()： 获取发送方的端口号  \* 5、关闭Socket。  \*  \* ps：发送端与接收端是两个对立的运行程序。  \*/  **public** **class** UDPReceive {  **public** **static** **void** main(String[] args) **throws** IOException {  //1、建立接收端  DatagramSocket ds = **new** DatagramSocket(12345);    //2、建立数据包，用于接收数据  **byte**[] buf = **new** **byte**[1024];  DatagramPacket dp = **new** DatagramPacket(buf, buf.length);    //3、调用Socket的接收方法  ds.receive(dp);    //4、解析数据  InetAddress sendAddress = dp.getAddress();  String sendIP = sendAddress.getHostAddress();  **int** sendPort = dp.getPort();    **byte**[] data = dp.getData();  **int** length = dp.getLength();    //将字节数组转成字符串  String content = **new** String(data, 0, length);    System.***out***.println("收到来自"+sendIP+"的，端口号为"+sendPort+"的数据："+content);    //5、关闭Socket  ds.close();    }  } |

## UDP案例

需求：

发送端：不断的向接收端发送数据

接收端：不断的接收从发送端发送过来的数据

### 案例代码是四:

|  |
| --- |
| **package** com.igeek\_03;  **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.InetAddress;  **import** java.net.SocketException;  **import** java.net.UnknownHostException;  **import** java.util.Scanner;  /\*\*  \* **@ClassName**: UDPSend  \* **@Description**: UDP的发送端  \* **@date** 2018年2月6日 下午4:08:39  \* Company www.igeekhome.com  \*  \* 反复向接收端发送数据  \*/  **public** **class** UDPSend {  **public** **static** **void** main(String[] args) **throws** IOException {  //1、建立发送端  DatagramSocket ds = **new** DatagramSocket(54321);    //反复发送数据  **while**(**true**){  //2、创建数据  Scanner scanner = **new** Scanner(System.***in***);  System.***out***.print("请输入发送的内容：");  String s = scanner.nextLine();    //3、建立数据包  **byte**[] data = s.getBytes();  **int** length = data.length;  InetAddress address = InetAddress.*getByName*("192.168.2.99");  **int** port = 12345;  DatagramPacket dp = **new** DatagramPacket(data , length, address, port );    //4、调用Socket的发送  ds.send(dp);  }    }  } |

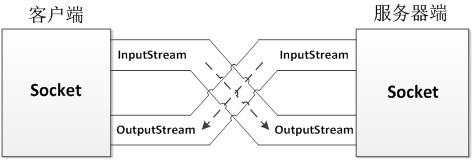
|  |
| --- |
| **package** com.igeek\_03;  **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.InetAddress;  **import** java.net.SocketException;  /\*\*  \* **@ClassName**: UDPReceive  \* **@Description**: UDP的接收端  \* **@date** 2018年2月6日 下午4:31:45  \* Company www.igeekhome.com  \*  \* 反复接收从发送端发送过来的数据  \*/  **public** **class** UDPReceive {  **public** **static** **void** main(String[] args) **throws** IOException {  //1、建立接收端  DatagramSocket ds = **new** DatagramSocket(12345);    //反复接收数据  **while**(**true**){  //2、建立数据包，用于接收数据  **byte**[] buf = **new** **byte**[1024];  DatagramPacket dp = **new** DatagramPacket(buf, buf.length);    //3、调用Socket的接收方法  ds.receive(dp);    //4、解析数据  InetAddress sendAddress = dp.getAddress();  String sendIP = sendAddress.getHostAddress();  **int** sendPort = dp.getPort();    **byte**[] data = dp.getData();  **int** length = dp.getLength();    //将字节数组转成字符串  String content = **new** String(data, 0, length);    System.***out***.println("收到来自"+sendIP+"的，端口号为"+sendPort+"的数据："+content);  }    }  } |

# TCP协议

## 概述:

TCP协议通过三次握手协议将客户端与服务器端连接，两端使用各自的Socket对象。Socket对象中包含了IO流，供数据传输。

即：TCP协议在客户端与服务器端通过Socket组成了IO通道，数据在IO通道中传输，客户端传输的数据成队列方式写出，服务器端接收，同样服务器端传输的数据成队列方式写出，客户端接收。



## TCP协议实现

Socket客户端和ServerSocket服务器端：

1、建立客户端和服务器端；

2、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；

（如果是服务器端，则需要添加一步操作：通过Socket服务获取Socket，再获取其中的IO流）

3、关闭socket。

注意点：

1、服务器端开启后等待客户端访问，可以不关闭；

2、一个服务端可以对应多个客户端；

3、不同客户端间通信可以通过服务器端中转信息。

### TCP客户端

#### 案例代码五:

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.IOException;  **import** java.io.OutputStream;  **import** java.net.Socket;  **import** java.net.UnknownHostException;  /\*\*  \* **@ClassName**: Client  \* **@Description**: 客户端  \* **@date** 2018年2月6日 下午7:31:14  \* Company www.igeekhome.com  \*  \* 客户端发送信息，服务端接收信息  \*  \* Socket客户端：  \* 1、建立客户端；  \* public Socket(String host, int port)： 指定服务器IP与端口  \* 2、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  \* public OutputStream getOutputStream()： 输出数据  \* public InputStream getInputStream()： 输入数据  \* 3、关闭socket。  \*/  **public** **class** Client {  **public** **static** **void** main(String[] args) **throws** UnknownHostException, IOException {  //1、建立客户端  Socket socket = **new** Socket("192.168.2.99", 55555);    //2、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输  OutputStream out = socket.getOutputStream();    //输出数据  out.write(97);    //3、关闭socket  socket.close();  }  } |

### TCP服务端

#### 案例代码六:

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.net.ServerSocket;  **import** java.net.Socket;  /\*\*  \* **@ClassName**: Server  \* **@Description**: 服务器端  \* **@date** 2018年2月6日 下午7:39:07  \* Company www.igeekhome.com  \*  \* 客户端发送信息，服务端接收信息  \*  \* ServerSocket服务器端：  \* 1、建立服务器端；  \* public ServerSocket(int port)： 指定服务器端端口，方便客户端访问  \* 2、由于是服务器端，则需要添加一步操作：通过Socket服务获取Socket，再获取其中的IO流  \* public Socket accept()： 接收一个socket对象  \* 3、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  \* public OutputStream getOutputStream()： 输出数据  \* public InputStream getInputStream()： 输入数据  \* 4、关闭socket。  \*/  **public** **class** Server {  **public** **static** **void** main(String[] args) **throws** IOException {  //1、建立服务器端；  ServerSocket serverSocket = **new** ServerSocket(55555);    //2、由于是服务器端，则需要添加一步操作：通过Socket服务获取Socket，再获取其中的IO流  Socket socket = serverSocket.accept();    //3、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  InputStream in = socket.getInputStream();    **int** read = in.read();    System.***out***.println(read);  System.***out***.println((**char**)read);    //4、关闭socket。  socket.close();  serverSocket.close();  }  } |

## 服务端向客户端发送数据

### 案例代码七:

|  |
| --- |
| **package** com.igeek\_03;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **import** java.net.Socket;  **import** java.net.UnknownHostException;  /\*\*  \* **@ClassName**: Client  \* **@Description**: 客户端  \* **@date** 2018年2月6日 下午7:52:58  \* Company www.igeekhome.com  \*  \* 服务器响应数据  \*  \* 客户端只需要按照顺序，在输出数据后，再获取输入流，输入数据即可  \*/  **public** **class** Client {  **public** **static** **void** main(String[] args) **throws** UnknownHostException, IOException {  //1、建立客户端；  Socket socket = **new** Socket("192.168.2.99", 55555);    //2、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  OutputStream out = socket.getOutputStream();    //输出数据  out.write(97);    //获取输入流，输入响应数据  InputStream in = socket.getInputStream();  **int** read = in.read();    System.***out***.println(read);  System.***out***.println((**char**)read);    //3、关闭socket。  socket.close();  }  } |

|  |
| --- |
| **package** com.igeek\_03;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **import** java.net.ServerSocket;  **import** java.net.Socket;  /\*\*  \* **@ClassName**: Server  \* **@Description**: 服务器端  \* **@date** 2018年2月6日 下午8:00:43  \* Company www.igeekhome.com  \*  \* 服务器响应数据  \*  \* 服务器端只需要按照顺序，在完成输入数据后，再获取输出流，输出数据即可  \*/  **public** **class** Server {  **public** **static** **void** main(String[] args) **throws** IOException {  //1、建立服务器端；  ServerSocket serverSocket = **new** ServerSocket(55555);    //2、由于是服务器端，则需要添加一步操作：通过Socket服务获取Socket，再获取其中的IO流  Socket socket = serverSocket.accept();    //3、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  InputStream in = socket.getInputStream();    **int** read = in.read();    System.***out***.println(read);  System.***out***.println((**char**)read);    //获取输出流，输出响应数据  OutputStream out = socket.getOutputStream();  out.write(98);    //4、关闭socket。  socket.close();  serverSocket.close();  }  } |

## 服务端循环读取客户端发送的数据

### 案例代码八:

|  |
| --- |
| **package** com.igeek\_04;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **import** java.net.Socket;  **import** java.net.UnknownHostException;  /\*\*  \* **@ClassName**: Client  \* **@Description**: 客户端  \* **@date** 2018年2月6日 下午7:52:58  \* Company www.igeekhome.com  \*  \* 客户端发送一个字符串i love java信息，接收服务器端返回一个字母o的响应  \*  \* public void shutdownOutput(): 禁止输出流，告知对方  \*/  **public** **class** Client {  **public** **static** **void** main(String[] args) **throws** UnknownHostException, IOException {  //1、建立客户端；  Socket socket = **new** Socket("192.168.2.99", 55555);    //2、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  OutputStream out = socket.getOutputStream();    //输出数据  String content = "i love java";  out.write(content.getBytes());    //告知服务器，不再发送数据了  socket.shutdownOutput();    //获取输入流，输入响应数据  InputStream in = socket.getInputStream();  **int** read = in.read();    System.***out***.println(read);  System.***out***.println((**char**)read);    //3、关闭socket。  socket.close();  }  } |

|  |
| --- |
| **package** com.igeek\_04;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **import** java.net.ServerSocket;  **import** java.net.Socket;  /\*\*  \* **@ClassName**: Server  \* **@Description**: 服务器端  \* **@date** 2018年2月6日 下午8:00:43  \* Company www.igeekhome.com  \*  \* 客户端发送一个字符串i love java信息，接收服务器端返回一个字母o的响应  \*/  **public** **class** Server {  **public** **static** **void** main(String[] args) **throws** IOException {  //1、建立服务器端；  ServerSocket serverSocket = **new** ServerSocket(55555);    //2、由于是服务器端，则需要添加一步操作：通过Socket服务获取Socket，再获取其中的IO流  Socket socket = serverSocket.accept();    //3、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  InputStream in = socket.getInputStream();    //反复使用一个字节接收  **int** read = -1;    **while**((read=in.read())!=-1){  System.***out***.print((**char**)read);  }    //获取输出流，输出响应数据  OutputStream out = socket.getOutputStream();  String content ="o";  out.write(content.getBytes());    //4、关闭socket。  socket.close();  serverSocket.close();  }  } |

## 上传文件案例

需求：将文件上传到指定IP的主机（这里的客户端和服务器都使用同一个机器）。

### 案例代码九:

|  |
| --- |
| **package** com.igeek\_05;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **import** java.net.Socket;  **import** java.net.UnknownHostException;  /\*\*  \* **@ClassName**: Client  \* **@Description**: 客户端  \* **@date** 2018年2月6日 下午7:52:58  \* Company www.igeekhome.com  \*  \* 文件上传，客户端发送的数据是一个文件当中的数据  \*  \* 读取文件内容，发送给服务器端  \*/  **public** **class** Client {  **public** **static** **void** main(String[] args) **throws** UnknownHostException, IOException {  //1、建立客户端；  Socket socket = **new** Socket("192.168.2.99", 55555);    //2、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  OutputStream out = socket.getOutputStream();    //获取资源文件输入流对象  FileInputStream fis = **new** FileInputStream("ASCII码表完整版.doc");    //读取文件内容  **int** b=-1;    **while**((b=fis.read())!=-1){  //输出数据给服务器  out.write(b);  }    //关闭资源文件输入流  fis.close();    //告知服务器，不再发送数据了  socket.shutdownOutput();    //获取输入流，输入响应数据  InputStream in = socket.getInputStream();  **int** read = in.read();    System.***out***.println(read);  System.***out***.println((**char**)read);    //3、关闭socket。  socket.close();  }  } |

|  |
| --- |
| **package** com.igeek\_05;  **import** java.io.FileOutputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **import** java.net.ServerSocket;  **import** java.net.Socket;  /\*\*  \* **@ClassName**: Server  \* **@Description**: 服务器端  \* **@date** 2018年2月6日 下午8:00:43  \* Company www.igeekhome.com  \*  \* 文件上传，服务器端接收文件中的数据，向服务器端的一个文件中写出  \*  \* 接收文件内容，写出到某个文件  \*/  **public** **class** Server {  **public** **static** **void** main(String[] args) **throws** IOException {  //1、建立服务器端；  ServerSocket serverSocket = **new** ServerSocket(55555);    //2、由于是服务器端，则需要添加一步操作：通过Socket服务获取Socket，再获取其中的IO流  Socket socket = serverSocket.accept();    //3、建立连接后，通过Socket中的IO流(Socket流)进行数据的传输；  InputStream in = socket.getInputStream();    //获取上传文件在服务器端的那个文件输出流  FileOutputStream fos = **new** FileOutputStream("ASCII码表完整版copy.doc");    //反复使用一个字节接收  **int** b = -1;    **while**((b=in.read())!=-1){  //每上传一个字节，就写到服务器端这个文件中一个字节  //最终这个文件就接收了所有上传的数据，即文件上传  fos.write(b);  }    //关闭文件流对象  fos.close();    //获取输出流，输出响应数据  OutputStream out = socket.getOutputStream();  String content ="o";  out.write(content.getBytes());    //4、关闭socket。  socket.close();  serverSocket.close();  }  } |

重点和总结

1、网络编程概述

2、使用UDP协议完成数据通信

3、DatagramSocket和DatagramPacket的使用

4、使用TCP协议完成数据通信

5、套接字Socket和ServerSocket的使用