# 第16天【网络编程】

## 主要内容

1. 网络基本概念
2. TCP编程
3. UDP编程

## 学习目标

|  |  |
| --- | --- |
| 知识点 | 要求 |
| 计算机网络 | 了解 |
| ISO/OSI模型和TCP/IP协议栈 | 了解 |
| TCP与UDP区别和联系 | 理解 |
| IP地址和端口号 | 掌握 |
| URL统一资源定位符 | 掌握 |
| Socket套接字 | 理解 |
| TCP编程-一次单向通信 | 了解 |
| TCP编程-一次双向通信 | 了解 |
| TCP编程-对象传输 | 了解 |
| TCP编程-多线程通信 | 掌握 |
| UDP编程-一次单向通信 | 了解 |
| UDP编程-一次双向通信 | 了解 |
| UDP编程-多次双向 | 了解 |

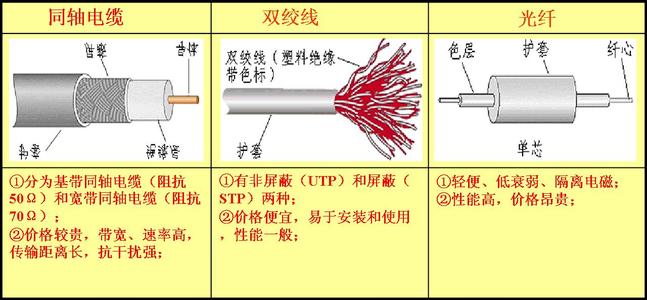
## 一、网络基本概念

### 1.1计算机网络

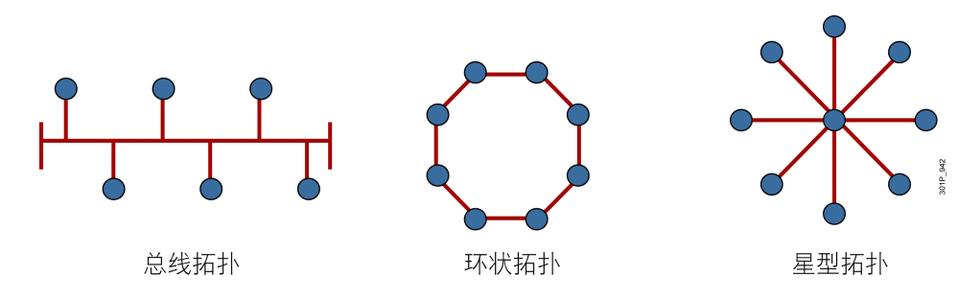
* **计算机网络定义**

计算机网络是指将地理位置不同的具有独立功能的多台计算机及其外部设备，通过通信线路连接起来，在网络操作系统，网络管理软件及网络通信协议的管理和协调下，实现资源共享和信息传递的计算机系统。

* **计算机网络定义分析**
  + 主干：计算机网络是计算机系统
  + 网络功能：资源共享 信息传递
  + 网络组成：
    - 网络硬件：计算机 外部设备 通信线路 （连接）
    - 网络软件：网络操作系统 网络管理软件 网络通信协议（管理协调）
* **计算机网络分类1（按照规模）**
  + 局域网LAN
  + 城域网MAN
  + 广域网WAN
* **计算机网络分类2（传输介质）**
  + 同轴电缆网络：和有线电视网的电缆相似
  + 双绞线网络：8根网线，不同颜色，两两相绞
  + 光纤网络：快 容量大 光信号，玻璃介质
  + 卫星网络

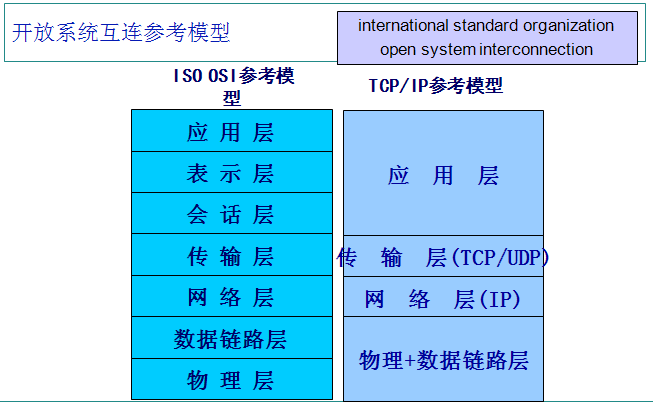


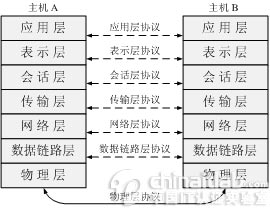
* **计算机网络分类3（拓扑结构）**
  + 星形网络
  + 总线网络
  + 环状网络



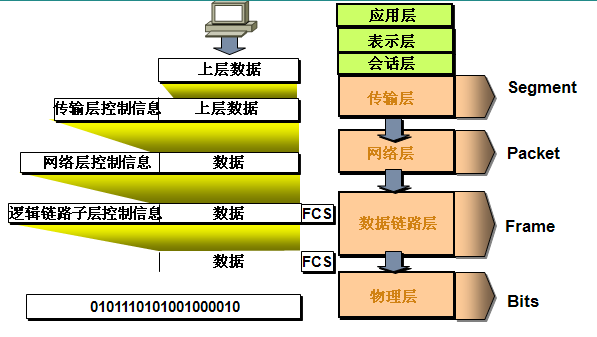
### 1.2网络通信协议

* **网络通信协议**
  + 计算机网络中实现通信必须有一些约定即通信协议，对速率、传输代码、代码结构、传输控制步骤、出错控制等制定标准
  + 好比公路交通规则，学生守则
* **问题：网络协议太复杂**
  + 计算机网络通信涉及内容很多，比如指定源地址和目标地址，加密解密，压缩解压缩，差错控制，流量控制，路由控制，如何实现如此复杂的网络协议呢？
* **解决方案：分层**
  + 由于结点间联系复杂，制定协议时，把复杂成份分解成 一些简单的成份，再将它们复合起来。最常用的复合方式是层次方式，即同层间可以通信、上一层可以调用下一层，而与再下一层不发生关系。
* **网络通信协议的分层**
  + 名义上标准：ISO/OSI参考模型
  + 事实上标准： TCP/IP协议栈（Internet使用的协议）

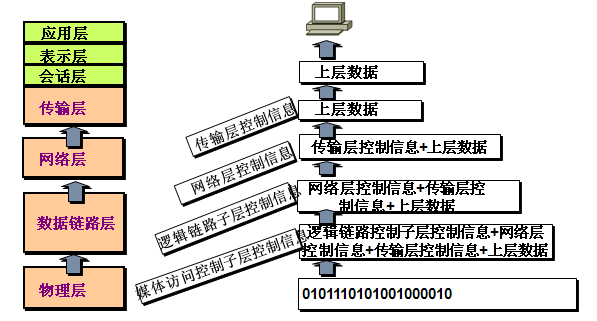




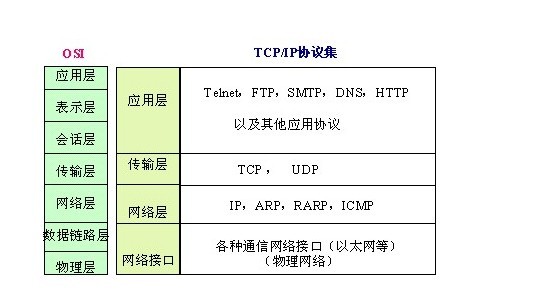
* 数据封装与拆封
  + 1、封装（发送数据）



* + 2、拆封（接收数据）

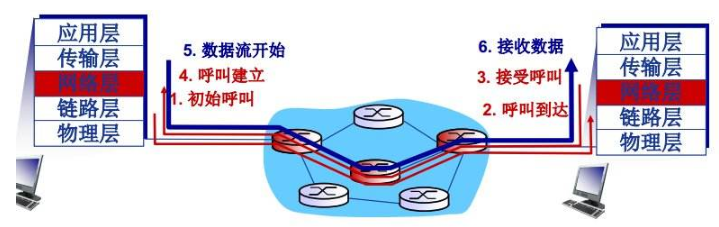


* TCP/IP协议栈
  + 网络层主要协议IP协议
  + 传输层主要协议TCP和UDP

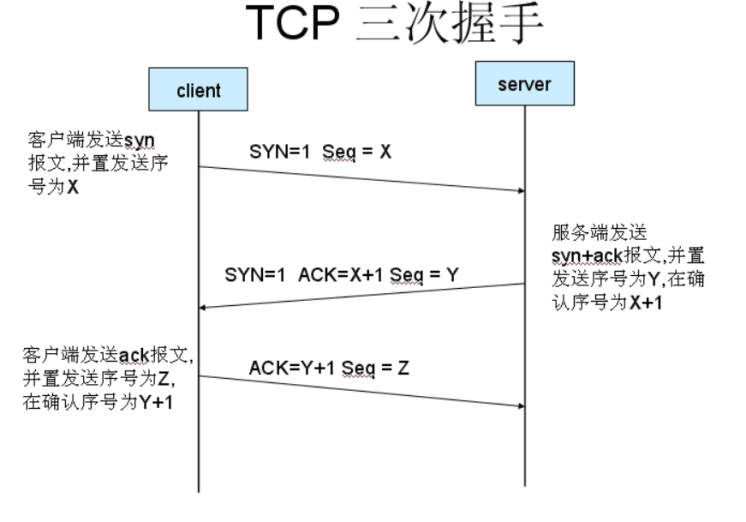


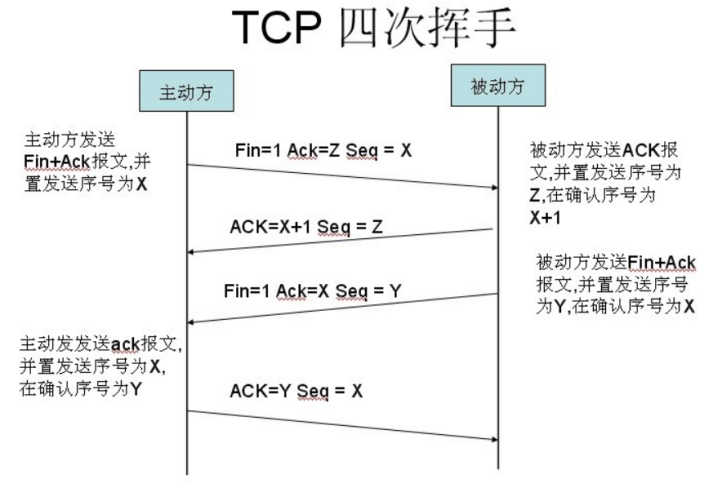
### 1.3 TCP协议和UDP协议

* TCP(transfer control protocol)
  + 一种面向连接（连接导向）的、可靠的、基于字节流的运输层（Transport layer）通信协议 。

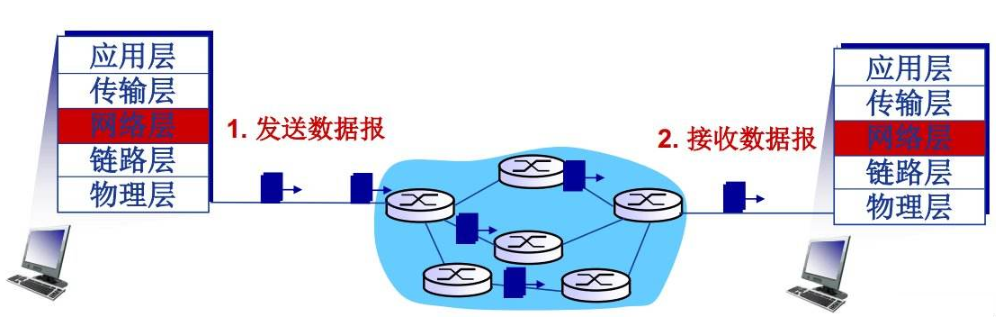


* 特点
  + 面向连接
  + 点到点的通信
  + 高可靠性:三次握手
  + 占用系统资源多、效率低
* 生活案例
  + 打电话 QQ 邮件发送
* 应用案例：
  + HTTP：www服务器 浏览网页 FTP：文件传输 上传下载
  + TELNET：远程登录 SMTP:发送邮件





* UDP(User DatagramProtocol )
  + 一种无连接传输层协议，提供面向事务的简单不可靠信息传送服务



* 特点:
  + 非面向连接，传输不可靠，可能丢失
  + 发送不管对方是否准备好，接收方收到也不确认
  + 可以广播发送
  + 非常简单的协议，开销小
* 生活案例：
  + 发送短信 发电报
* 应用案例：
  + DNS：将用户认识的域名解析为计算机认识的ip地址
  + SNMP：使网络管理员能够管理网络效能，发现并解决网络问题以及规划网络增长。通过 SNMP 接收随机消息（及事件报告）网络管理系统获知网络出现问题

1、连接方面区别

TCP面向连接（如打电话要先拨号建立连接）。

UDP是无连接的，即发送数据之前不需要建立连接。

2、安全方面的区别

TCP提供可靠的服务，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达。

UDP尽最大努力交付，即不保证可靠交付。

3、传输效率的区别

TCP传输效率相对较低。

UDP传输效率高，适用于对高速传输和实时性有较高的通信或广播通信。

4、连接对象数量的区别

TCP连接只能是点到点、一对一的。

UDP支持一对一，一对多，多对一和多对多的交互通信。

|  |
| --- |
| **生活案例：采用TCP和UDP方式去欢乐谷游玩**  方案1：TCP协议  先定好路线，进行分组，依次出发，先发先到  方案2：UDP协议  告诉目的地，线路不确定，进行分组，分头行动，最终到达目的，无法保证先发先到 |

### 1.4 IP地址和端口

* IP地址
  + 用来标志网络中的一个通信实体的地址。通信实体可以是计算机，路由器等。
* IP地址分类
  + IPV4：32位地址，以点分十进制表示，如192.168.0.1
  + IPV6：128位（16个字节）写成8个16位的无符号整数，每个整数用四个十六进制位表示，数之间用冒号（：）分开，如：3ffe:3201:1401:1280:c8ff:fe4d:db39:1984
* 特殊的IP地址
  + 127.0.0.1 本机地址
  + 192.168.0.0--192.168.255.255私有地址，属于非注册地址，专门为组织机构内部使用。

端口:port

* + IP地址用来标志一台计算机，但是一台计算机上可能提供多种应用程序，使用端口来区分这些应用程序。
  + 端口是虚拟的概念，并不是说在主机上真的有若干个端口。通过端口，可以在一个主机上运行多个网络应用程序。
  + 端口范围0---65535,16位整数
* 理解IP和端口的关系
  + IP地址好比每个人的地址（门牌号），端口好比是房间号。必须同时指定IP地址和端口号才能够正确的发送数据
  + IP地址好比为电话号码，而端口号就好比为分机号。
* 端口分类
  + 公认端口 0—1023 比如80端口分配给WWW，21端口分配给FTP
  + 注册端口 1024—49151 分配给用户进程或应用程序
  + 动态/私有端口 49152—65535
* IP和端口API
  + InetAddress类 封装计算机的ip地址 ,没有端口
  + InetSocketAddress 包含端口，用于socket通信的

#### 【示例1】InetAddress类的使用

|  |
| --- |
| **public class** TestInetAddress {  **public static void** main(String[] args) **throws** UnknownHostException {  *//认识本机的IP地址* InetAddress ia = InetAddress.*getLocalHost*();  System.***out***.println(ia);  System.***out***.println(ia.getHostName());  System.***out***.println(ia.getHostAddress());  *//System.out.println(ia.getAddress());* **byte**[] buf = ia.getAddress();  System.***out***.println(buf.**length**);  System.***out***.println(Arrays.*toString*(buf)); InetAddress ia2 = InetAddress.*getByName*(**"www.bjsxt.com"**);  System.***out***.println(ia2);  System.***out***.println(ia2.getHostAddress());  System.***out***.println(Arrays.*toString*(ia2.getAddress()));  } } |

#### 【示例2】InetSocketAddress 类的使用

|  |
| --- |
| **public class** TestInetSocketAddress {  **public static void** main(String[] args) **throws** UnknownHostException { // InetSocketAddress isa=new InetSocketAddress("www.baidu.com", 1234);  InetAddress addr = InetAddress.getByName(**"www.baidu.com"**);  **int** port = 4321;  InetSocketAddress isa = **new** InetSocketAddress(addr, port);  System.**out**.println(isa);  System.**out**.println(isa.getAddress());  System.**out**.println(isa.getPort());  } } |

### 1.5 URL统一资源定位符

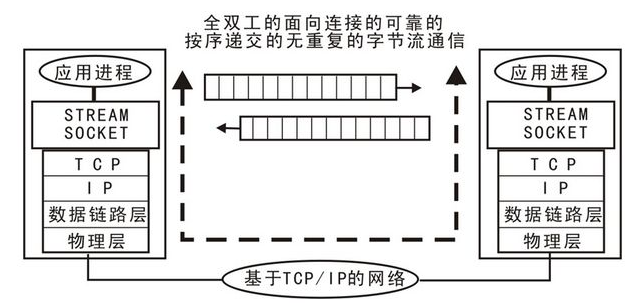
* URL（Uniform Resource Locator）
  + 统一资源定位符,由4部分组成：协议 、存放资源的主机域名（IP）、端口号和资源文件名。
* URL是指向互联网“资源”的指针。
  + 资源可以是简单的文件或目录，也可以是对更为复杂的对象的引用，例如对数据库或搜索引擎的查询。

#### 【示例3】URL类的使用

|  |
| --- |
| **public class** TestURL {   **public static void** main(String[] args) **throws** IOException {  //URL url = new  //URL("https://www.bjsxt.com:90/news/11068.html?name=zhsan&pwd=lisi");  URL url = **new** URL(**"https://www.bjsxt.com:90/news/11068.html#cloths"**);  System.**out**.println(url.getProtocol());  System.**out**.println(url.getHost());  System.**out**.println(url.getPort());  System.**out**.println(url.getDefaultPort());  System.**out**.println(url.getPath());  System.**out**.println(url.getQuery());  System.**out**.println(url.getRef());  System.**out**.println(url.getAuthority());  System.**out**.println(url.getContent());  System.**out**.println(url.getFile());  } } |

### 1.6 Socket套接字

* 套接字Socket
  + 我们开发的网络应用程序位于应用层，TCP和UDP属于传输层协议，在应用层如何使用传输层的服务呢？在应用层和传输层之间，则是使用套接字来进行分离。
  + 套接字就像传输层为应用层开的一个小口，应用程序通过这个小口向远程发送数据，或接收远程发来数据；而这个小口以内，也就是数据进入这个口之后，或者数据从这个口出来之前，是不知道也不需要知道的，也不会关心它如何传输，这属于网络其它层次的工作。



* Socket实际是传输层供给应用层的编程接口。传输层则在网络层的基础上提供进程到进程问的逻辑通道，而应用层的进程则利用传输层向另一台主机的某一进程通信。Socket就是应用层与传输层之间的桥梁
* 使用Socket编程可以开发客户机和服务器应用程序，可以在本地网络上进行通信，也可通过Internet在全球范围内通信。
* 生活案例1：
  + 如果你想写封邮件发给远方的朋友，如何写信、将信打包，属于应用层。信怎么写，怎么打包完全由我们做主；
  + 而当我们将信投入邮筒时，邮筒的那个口就是套接字，在进入套接字之后，就是传输层、网络层等（邮局、公路交管或者航线等）其它层次的工作了。我们不会去关心信是如何从西安发往北京的，我们只知道写好了投入邮筒。
* 生活案例2：
  + 如果你想发货给国外，你只要把货物放入集装箱，然后交给码头就可以了。发送什么货物，货物如何打包，完全有你做主。
  + 码头就是套接字，剩下的事情就交给港口和货运公司处理就行了，具体细节我们无需了解。

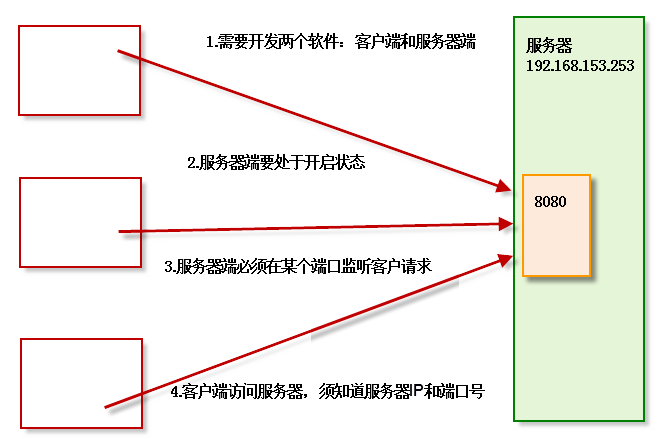
### 本节作业

1. 介绍TCP/IP协议栈
2. TCP协议和UDP协议的比较
3. IP地址和端口的作用、分类、关系
4. IP地址、端口port和URL的联系

## 二、TCP编程

### 2.1 TCP编程-一次单向通信

功能：实现类似QQ、微信、邮箱、商城的网络登录功能，可多个用户同时登录。为便于理解，进行功能分解迭代，分为一次单向通信、一次双向通信、传输对象、引入多线程来实现。



#### 【示例4】网络登录-一次单向通信

|  |
| --- |
| **public class** LoginServer {  **public static void** main(String[] args) **throws** IOException {  //1.创建一个ServerSocket，负责监听客户端请求  ServerSocket serverSocket = **new** ServerSocket(8800);  //2.使用ServerSocket进行监听  Socket socket = serverSocket.accept();//请求未到，在此阻塞  //3.处理用户的请求  InputStream is = socket.getInputStream();  DataInputStream dis = **new** DataInputStream(is);  String result = dis.readUTF();  System.**out**.println(**"这里是服务器端："**+result);  //4.关闭资源  dis.close();  } }  **public class** LoginClient {  **public static void** main(String[] args) **throws** IOException {  //1.创建一个Socket，指明服务器端的IP地址和端口号  Socket socket = **new** Socket(InetAddress.getByName(**"192.168.153.253"**),8800);  //2.向服务器端发起一个新的请求  OutputStream os = socket.getOutputStream();  DataOutputStream dos = **new** DataOutputStream(os);  dos.writeUTF(**"username=bjsxt&password=bjsxt"**);  //3.关闭资源  dos.close();  } } |

### 2.2 TCP编程-一次双向通信

#### 【示例5】网络登录-一次双向通信

|  |
| --- |
| **public class** LoginServer {  **public static void** main(String[] args) **throws** IOException {  //1.创建一个ServerSocket，负责监听客户端请求  ServerSocket serverSocket = **new** ServerSocket(8800);  //2.使用ServerSocket进行监听  Socket socket = serverSocket.accept();//请求未到，在此阻塞  //3.处理用户的请求  InputStream is = socket.getInputStream();  DataInputStream dis = **new** DataInputStream(is);  String result = dis.readUTF();  System.**out**.println(**"这里是服务器端："**+result);  //4.给出客户端响应  OutputStream os = socket.getOutputStream();  DataOutputStream dos = **new** DataOutputStream(os);  dos.writeUTF(**"您好，登录成功，欢迎您"**);  //5.关闭资源  dis.close();  dos.close();  } }  **public class** LoginClient {  **public static void** main(String[] args) **throws** IOException {  //1.创建一个Socket，指明服务器端的IP地址和端口号  Socket socket = **new**  Socket(InetAddress.getByName(**"192.168.153.253"**),8800);  //2.向服务器端发起一个新的请求  OutputStream os = socket.getOutputStream();  DataOutputStream dos = **new** DataOutputStream(os);  dos.writeUTF(**"username=bjsxt&password=bjsxt"**);  //3.接收来自服务器端的响应  InputStream is = socket.getInputStream();  DataInputStream dis = **new** DataInputStream(is);  String result = dis.readUTF();  System.**out**.println(**"这里是客户端："**+result);  //3.关闭资源  dis.close();  dos.close();  } } |

### 2.3 TCP编程-传输对象

* 在一次双向通信基础上进步优化，在客户端输入用户名和密码，并封装到User对象中。如何在TCP编程中传输对象呢？

#### 【示例6】网络登录-传输对象

|  |
| --- |
| **public class** User **implements** Serializable {  **private** String **userId**;  **private** String **password**;  **}**  **public class** LoginServer {  **public static void** main(String[] args)  **throws** IOException, ClassNotFoundException {  *//1.创建一个ServerSocket，负责监听客户端请求* ServerSocket serverSocket = **new** ServerSocket(8800);  *//2.使用ServerSocket进行监听* Socket socket = serverSocket.accept();*//请求未到，在此阻塞  //3.处理用户的请求* InputStream is = socket.getInputStream();  ObjectInputStream ois = **new** ObjectInputStream(is);  User user = (User)ois.readObject();  System.***out***.println(**"这里是服务器端："**+user);  *//4.给出客户端响应* OutputStream os = socket.getOutputStream();  DataOutputStream dos = **new** DataOutputStream(os);  **if**(user.getUserId().indexOf(**"sxt"**)>=0){  dos.writeUTF(**"您好，登录成功，欢迎您"**);  }**else**{  dos.writeUTF(**"您好，登录失败，请重新输入"**);  }  *//4.关闭资源* ois.close();  dos.close(); } }  **public class** LoginClient {  **public static void** main(String[] args) **throws** IOException {  *//1.创建一个Socket，指明服务器端的IP地址和端口号* Socket socket =  **new** Socket(InetAddress.*getByName*(**"192.168.153.253"**),8800);  *//从键盘输入用户名和密码，封装到User对象中* Scanner input = **new** Scanner(System.***in***);  System.***out***.println(**"请输入用户名："**);  String userId = input.next();  System.***out***.println(**"请输入密码："**);  String password = input.next();  User user = **new** User(userId,password);  *//2.向服务器端发起一个新的请求* OutputStream os = socket.getOutputStream();  ObjectOutputStream oos = **new** ObjectOutputStream(os);  oos.writeObject(user);  *//3.接收来自服务器端的响应* InputStream is = socket.getInputStream();  DataInputStream dis = **new** DataInputStream(is);  String result = dis.readUTF();  System.***out***.println(**"这里是客户端："**+result);  *//3.关闭资源* dis.close();  oos.close();  } } |

* 使用TCP实现网络登录功能总结
  + 服务器创建ServerSocket，在指定端口监听并并处理请求；
  + ServletSocket通过accept() 接收用户请求并返回对应的Socket，否则一种处于监听等待状态，线程也被阻塞
  + 客户端创建Socket，需要指定服务器的ip和端口号，向服务器发送和接收响应
  + 客户端发送数据需输出流(写)，客户端获取反馈数据需输入流(读)
  + 服务端反馈数据需输出流(写)，服务端获取请求数据需输入流(读)
  + 一旦使用ServerSocket和Socket建立了网络连接后，网络通信和普通IO流操作并没有太大区别
  + 网络通信输出流建议使用DataOutputStream和ObjectOutputStream,与平台无关，输入流相应使用DataIntputStream和ObjectInputStream
  + 如果是字符串通信也可以使用BufferedReader和PrintWriter，简单方便

### 2.4 TCP编程-引入多线程

* 实现应用中，服务器端一直处于运行状态，并且会出现多个用户同时登录的情况，需要服务器端进行处理。
* 引入多线程，服务器端收到请求后，开辟一个新的线程，实现对用户请求的处理。

#### 【示例7】网络登录-引入多线程

|  |
| --- |
| **public class** LoginThread **extends** Thread {  **private** Socket **socket**;  **public** LoginThread(Socket socket) {  **this**.**socket** = socket;  }  @Override  **public void** run() {  **try**(ObjectInputStream ois = **new** ObjectInputStream(**socket**.getInputStream());  DataOutputStream dos = **new** DataOutputStream(**socket**.getOutputStream());  ) {  //1.获取客户端的请求数据并输出  User user = (User) ois.readObject();  System.***out***.println(**"info="** + user);  //2.给出客户端响应  **if** (**"bjsxt"**.equals(user.getUserId()) && user.getPassword().length() >= 6) {  dos.writeUTF(**"登录成功，欢迎您！！"**);  } **else** {  dos.writeUTF(**"登录失败，请再次输入"**);  }  } **catch** (IOException e) {  e.printStackTrace();  } **catch** (ClassNotFoundException e) {  e.printStackTrace();  } **catch** (Exception e) {  e.printStackTrace();  }  } }  **public class** LoginServer {  **public static void** main(String[] args)  **throws** IOException, ClassNotFoundException {  *//1.创建一个ServerSocket，负责监听客户端请求* ServerSocket serverSocket = **new** ServerSocket(8800);  **int** i = 0;  **while**(**true**){  *//2.使用ServerSocket进行监听* Socket socket = serverSocket.accept();*//请求未到，在此阻塞  //3.创建一个新的线程来处理请求* Thread thread = **new** LoginThread(socket);  thread.start();  *//4.输出请求的次数，客户端的ip地址* String ip = socket.getInetAddress().getHostAddress();  System.***out***.println(**"这是第"**+(++i)+**"次访问,对方的IP地址是："**+ip);  }  } } |

### 本节作业

1. 使用TCP编程实现网络登录（一次双向通信）
2. 引入多线程实现网络登录（处理多个客户端同时访问）
3. 总结TCP编程的步骤和注意事项

## 三、UDP编程

* 需求：完成在线咨询功能：
  + 客户和咨询师在线交流
* 分析
  + 使用基于UDP协议的Socket网络编程实现
  + 不需要利用IO流实现数据的传输
  + 每个数据发送单元被统一封装成数据包的方式，发送方将数据包发送到网络中，数据包在网络中去寻找他的目的地。
* UDP基本概念
  + DatagramSocket：用于发送或接收数据包
  + DatagramPacket：数据包

### 3.1 UDP编程-一次单向通信

#### 【示例8】在线客服-一次单向通信

|  |
| --- |
| **public class** AskServer {  **public static void** main(String[] args) **throws** IOException {  *//1.创建一个Socket，指明接收数据的端口* DatagramSocket socket = **new** DatagramSocket(8888);  *//2.接收数据并输出* **byte** [] buf = **new byte**[1024];  DatagramPacket packet = **new** DatagramPacket( buf, buf.**length**);  socket.receive(packet);  *//System.out.println(Arrays.toString(packet.getData()));* System.***out***.println(**new** String(packet.getData(),0,packet.getLength()));  System.***out***.println(packet.getLength());*//数据长度* System.***out***.println(packet.getAddress());*//客户端的ip* System.***out***.println(packet.getPort());*//客户端接收数据的端口  //3.关闭Socket* socket.close();  } }  **public class** AskClient {  **public static void** main(String[] args) **throws** IOException {  *//1.创建一个Socket，【可以不指明接收数据的端口，会默认分配】* DatagramSocket socket = **new** DatagramSocket();  *//2.封装数据并发送* String str = **"亲，在吗？"**;  **byte** [] buf = str.getBytes();  InetAddress address = InetAddress.*getByName*(**"192.168.153.253"**);  **int** port = 8888;  DatagramPacket packet = **new** DatagramPacket(buf, buf.**length**,address,port);  socket.send(packet);  *//3.关闭Socket* socket.close();  } } |

### 3.2 UDP编程-一次双向通信

#### 【示例9】在线客服-一次双向通信

|  |
| --- |
| **public class** AskServer {  **public static void** main(String[] args) **throws** IOException {  *//1.创建一个Socket，指明接收数据的端口* DatagramSocket socket = **new** DatagramSocket(8888);  *//2.接收数据并输出* **byte** [] buf = **new byte**[1024];  DatagramPacket packet=**new** DatagramPacket( buf, buf.**length**);  socket.receive(packet);  System.***out***.println(**new** String(packet.getData(),  0,packet.getLength()));  *// 3.发送响应信息给客户端* String str = **"亲，在的"**; **byte** [] buf2 = str.getBytes();  InetAddress address = packet.getAddress();  **int** port = packet.getPort();  DatagramPacket packet2 = **new** DatagramPacket(buf2,buf2.**length**,address,port);  socket.send(packet2);  *//4.关闭Socket* socket.close();  } }  **public class** AskClient {  **public static void** main(String[] args) **throws** IOException {  *//1.创建一个Socket，【可以不指明接收数据的端口，会默认分配】* DatagramSocket socket = **new** DatagramSocket();  *//2.封装数据并发送* String str = **"亲，在吗？"**;  **byte** [] buf = str.getBytes();  InetAddress address = InetAddress.*getByName*(**"192.168.153.253"**);  **int** port = 8888;  DatagramPacket packet =**new** DatagramPacket(buf, buf.**length**,address,port);  socket.send(packet);  *//3.接收服务器的响应信息* **byte** [] buf2 = **new byte**[1024];  DatagramPacket packet2 = **new** DatagramPacket(buf2,buf2.**length**);  socket.receive(packet2);  System.***out***.println(**new** String(packet2.getData(), 0,packet2.getLength()));  *//4.关闭Socket* socket.close();  } } |

### 3.3 UDP编程-多次双向通信

#### 【示例10】在线客服-多次双向通信

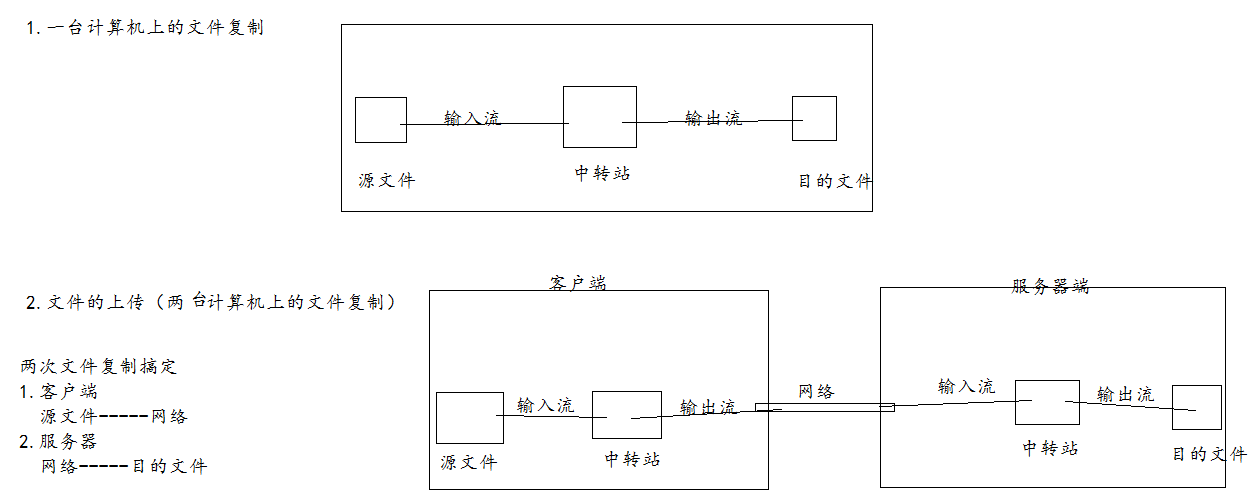
|  |
| --- |
| **public class** AskServer {  **public static void** main(String[] args) **throws** IOException {  *//1.创建一个Socket，指明接收数据的端口* DatagramSocket socket = **new** DatagramSocket(8888);  Scanner input = **new** Scanner(System.***in***);  **while**(**true**){  *//2.接收数据并输出* **byte** [] buf = **new byte**[1024];  DatagramPacket packet = **new** DatagramPacket( buf, buf.**length**);  socket.receive(packet);  *//System.out.println(Arrays.toString(packet.getData()));* String result = **new** String(packet.getData(),0,packet.getLength());  **if**(**"bye"**.equals(result)){  **break**;  }  System.***out***.println(result);  *// 3.发送响应信息给客户端* String str = input.nextLine();  **byte** [] buf2 = str.getBytes();  InetAddress address = packet.getAddress();  **int** port = packet.getPort();  DatagramPacket packet2 =**new** DatagramPacket(buf2,buf2.**length**,address,port);  socket.send(packet2);  }  *//4.关闭Socket* socket.close();  } }  **public class** AskClient {  **public static void** main(String[] args) **throws** IOException {  *//1.创建一个Socket，【可以不指明接收数据的端口，会默认分配】* DatagramSocket socket = **new** DatagramSocket();  Scanner input = **new** Scanner(System.***in***);  **while**(**true**){  *//2.封装数据并发送* String str = input.nextLine(); *// Java 大数据 前端 python* **byte** [] buf = str.getBytes();  InetAddress address = InetAddress.*getByName*(**"192.168.153.253"**);  **int** port = 8888;  DatagramPacket packet =**new** DatagramPacket(buf, buf.**length**,address,port);  socket.send(packet);  **if**(**"bye"**.equals(str)){  **break**;  }  *//3.接收服务器的响应信息* **byte** [] buf2 = **new byte**[1024];  DatagramPacket packet2 = **new** DatagramPacket(buf2,buf2.**length**);  socket.receive(packet2);  System.***out***.println(**new** String(packet2.getData(), 0,packet2.getLength()));  }  *//4.关闭Socket* socket.close();  } } |

### 本节作业

1. 说明DatagramSocket和DatagramPacket的作用
2. 使用UDP实现在线客服功能
3. 使用TCP编程实现文件上传功能

思路：进行两次文件复制

客户端将文件从本地复制到网络；服务端将文件从网络复制到本地；



#### 【示例11】文件上传-客户端

|  |
| --- |
| **public class** UploadClient {  **public static void** main(String[] args) **throws** Exception {  // 创建一个Socket，指明访问哪个服务器的哪个端口  Socket socket = **new** Socket(InetAddress.getLocalHost(), 8888);  // 使用Socket向服务器端上传文件  // 1.创建一个输入流和输出流  BufferedInputStream bis = **null**;  BufferedOutputStream bos = **null**;  bis = **new** BufferedInputStream(**new** FileInputStream(**new** File(**"d:/readme.txt"**)));  bos = **new** BufferedOutputStream(socket.getOutputStream());  // 2.使用输入流和输出流完成文件复制  // 2.1准备一个中转站（水杯）  **byte**[] buf = **new byte**[1024];  **int** len = bis.read(buf);  **while** (len != -1) {  bos.write(buf, 0, len);  len = bis.read(buf);  }  // 3.关闭输入流和输出流 省略  } } |

#### 【示例12】文件上传-服务器端

|  |
| --- |
| **public class** UploadServer {  **public static void** main(String[] args) **throws** Exception{  //创建一个ServerSocket，可以在某个端口监听  ServerSocket serverSocket = **new** ServerSocket(8888);   //使用创建的ServerSocket在指定的端口监听  Socket socket = serverSocket.accept();   //接收来自客户端的文件流，并完成将文件保持到本地   // 1.创建一个输入流和输出流  BufferedInputStream bis = **null**;  BufferedOutputStream bos = **null**;  bis = **new** BufferedInputStream(socket.getInputStream());//!!!!  bos = **new** BufferedOutputStream(**new** FileOutputStream(**"d:/readme2.txt"**));  // 2.使用输入流和输出流完成文件复制  // 2.1准备一个中转站（水杯）  **byte**[] buf = **new byte**[1024];  **int** len = bis.read(buf);  **while** (len != -1) {  bos.write(buf, 0, len);  len = bis.read(buf);  }  // 3.关闭输入流和输出流 省略  } } |